BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pakar

Sistem pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yang dipikirkan oleh pakar. Pakar yang dimaksud di sini adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam. Sebagai contoh, dokter adalah seorang pakar yang mampu mendiagnosis penyakit yang diderita pasien serta dapat memberikan penatalaksanaan terhadap penyakit tersebut. Tidak semua orang dapat mengambil keputusan mengenai diagnosis dan memberikan penatalaksanaan suatu penyakit. Contoh yang lain, monitor adalah seorang yang punya keahlian dan pengalaman dalam menyelesaikan kerusakan mesin motor/mobil, psikolog adalah orang yang ahli dalam memahami kepribadian seseorang, dan lain-lain (Kusrini ; 2010 : 3).

Sistem pakar yang mencoba memecahkan masalah yang biasanya hanya bisa dipecahkan oleh seorang pakar, dipandang berhasil ketika mampu mengambil keputusan seperti yang dilakukan oleh pakar aslinya baik dari sisi proses pengambilan keputusannya maupun hasil keputusan yang diperoleh.

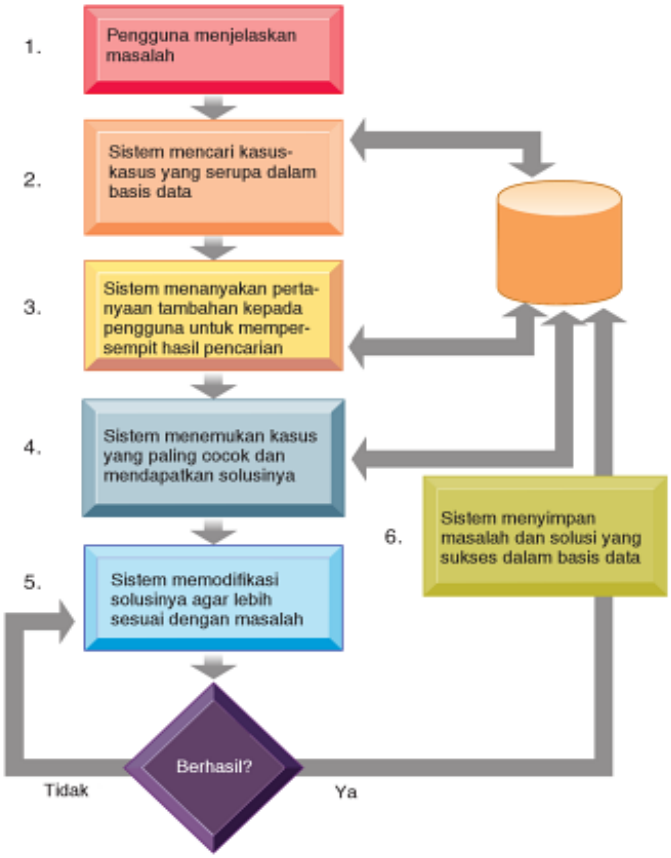
Sebuah sistem pakar memiliki 2 komponen utama yaitu basis pengetahuan dan mesin inferensi. Basis pengetahuan merupakan tempat penyimpanan pengetahuan dalam memori komputer, di mana pengetahuan ini diambil dari pengetahuan kaidah produksi.

Mesin inferensi merupakan otak dari aplikasi sistem pakar. Bagian inilah yang menuntut user untuk memasukkan fakta sehingga diperoleh suatu kesimpulan. Apa yang dilakukan oleh mesin inferensi ini didasarkan pada pengetahuan yang ada dalam basis pengetahuan.

II.2. Sistem Logika Fuzzy

Kebanyakan orang tidak berpikir dengan menggunakan aturan IF THEN yang tradisional. Manusia cenderung mengategorikan hal-hal secara tidak tepat, dengan menggunakan aturan pengambilan keputusan yang mungkin memiliki banyak perbedaan pengertian. Sebagai contoh, seorang laki-laki atau perempuan dapat saja kuat atau cerdas. Sebuah perusahaan dapat digolongkan sebagai besar, sedang, atau kecil. Suatu temperature dapat digolongkan sebagai panas, sejuk, dingin, atau hangat. Kategori-kategori ini merepresentasikan suatu rentang nilai. (Kennec, dkk ; 2010 : 124).

Logika fuzzy (*fuzzy logic*) adalah teknologi berbasis aturan yang dapat merepresentasikan ketidak presisian seperti yang telah disebutkan, dengan menciptakan aturan yang menggunakan nilai subjektif atau nilai yang mendekati. *Logika fuzzy* dapat menjelaskan fenomena atau proses tertentu secara linguistic, kemudian merepresentasikannya dalam sejumlah kecil aturan yang fleksibel. Organisasi dapat menggunakan *logika fuzzy* untuk menciptakan sistem piranti lunak yang menangkap pengetahuan tersirat yang mengandung *ambiguitas linguistic*. (Kennec, dkk ; 2010 : 125). Adapun cara kerja penalaran berbasis kasus dapat dilihat pada gambar II.1.



Gambar II.1. Cara kerja penalaran berbasis kasus
(Sumber : Kennech C, dkk ; 2010 : 126)

Lihat bagaimana *logika fuzzy* merepresentasikan temperature yang bervariasi dalam aplikasi computer untuk mengendalikan temperature ruangan secara otomatis. Istilah-istilah atau dikenal sebagai fungsi keanggotaan didefinisikan secara tidak presisi, sejuk adalah antara 50 derajat dan 70 derajat (*Fahrenheit*), padahal temperature sejuk lebih tepatnya berkisar antara 60 dan 67 derajat. Ingatlah bahwa kategori sejuk bertumpang tindih dengan kategoi dingin atau normal. Untuk mengendalikan lingkungan ruangan menggunakan logika fuzzy, programmer juga perlu mengembangkan definisi ketidakpresisian yang

serupa untuk kelembapan dan faktor-faktor lainnya, seperti misalnya angin dan temperature di luar ruangan.

Aturan-aturan tersebut mungkin mencakup satu pernyataan berikut jika temperature sejuk atau dingin dan kelembapan rendah, sementara angin di luar ruangan tinggi dan temperature luar ruangan rendah, naikkan panas dan kelembapan dalam ruangan. Komputer akan mengombinasikan pembacaan fungsi keanggotan dengan melakukan pembobotan dan, menggunakan semua aturannya, menaikkan atau menurunkan temperature dan kelembapan.

Logika fuzzy menyediakan solusi bagi masalah-masalah yang sulir dipecahkan hanya dengan menggunakan aturan IF THEN. Di jepang, sistem kereta bawah tanah sendai menggunakan control *logika fuzzy* untuk mengatur percepatan kereta dengan mulus sehingga para penumpang yang berdiri tidak perlu perpengangan saat kereta itu berjalan. Mitsubishi Heavy Industries di Tokyo telah berhasil mengurangi pemakaian listrik untuk penyejuk ruangan sampai 20 persen dengan mengimplementasikan program-program control *logika fuzzy*. Perangkat focus otomatis pada kamera hanya mungkin dibuat dengan menggunakan *logika fuzzy*. Pada contoh tersebut, *logika fuzzy* telah memungkinkan perubahan setahap demi setahap pada masukan menghasilkan perubahan yang mulus pada keluaran, dan bukan keluaran yang putus-putus. Hal tersebut membuat *logika fuzzy* sangat bermanfaat bagi konsumen aplikasi elektronik dan rekaya. (Kennec, dkk ; 2010 : 126).

Logika fuzzy juga dapat bermanfaat di bidang manajemen untuk pengambilan keputusan dan control organisasi. Sebuah perusahaan di Wall Street

menciptakan sistem yang memilih perusahaan-perusahaan yang berpotensi untuk diakuisisi dengan menggunakan bahasa yang dimengerti oleh para pedagang saham. Sistem logika fuzzy juga telah dikembangkan untuk mendeteksi kemungkinan kebohongan dalam klaim kesehatan yang diajukan oleh penyedia perawatan kesehatan di Amerika Serikat. (Kennec, dkk ; 2010 : 127).

II.2.1. Logika Pengambilan Keputusan

Logika pengambilan keputusan atau dapat disebut penyimpulan fuzzy (*fuzzy inference*) mengaplikasikan aturan-aturan fuzzy pada masukan fuzzy kemudian mengevaluasi setiap aturan. Prinsip logika fuzzy digunakan untuk mengkombinasi aturan-aturan JIKA-MAKA (IFTHEN) yang terdapat dalam basis aturan kedalam suatu pemetaan dari suatu himpunan *fuzzy input* kesuatu himpunan *fuzzy output*. Logika pengambilan keputusan merupakan langkah kedua dalam pemrosesan logika fuzzy. Terdapat beberapa metode pengambilan keputusan dalam logika fuzzy diantaranya yaitu metode Mamdani. (Sutikno ; 2010 : 2).

Fungsi implikasi yang digunakan pada pengambilan keputusan dengan metode Mamdani dengan menggunakan MIN dan dalam melakukan komposisi dengan menggunakan MAX. Metode komposisi ini sering disebut MAX-MIN. Contoh dalam penggunaan pengambilan keputusan dengan metode Mamdani ditunjukkan pada gambar II.2. Dengan memisalkan fungsi keanggotaan masukan dan keluaran menggunakan fungsi segitiga dan mempunyai 2 aturan fuzzy, yaitu:

a. IF Kesalahan adalah Nol dan Beda kesalahan adalah Positif maka Keluaran adalah Positif.

b. IF Kesalahan adalah Nol dan Beda kesalahan adalah Nol maka Keluaran adalah Nol.

Langkah pertama pengambilan keputusan metode Mamdani adalah melakukan proses fuzzifikasi untuk memetakan data tegas masukan kesalahan dan beda kesalahan kedalam data fuzzy sesuai dengan tipe dan bentuk fungsi keanggotaan. Langkah kedua adalah melakukan proses terhadap kedua data fuzzy tersebut dengan operator AND yang akan mengambil nilai paling minimal dari dua data tersebut. Langkah ketiga dengan implikasi MIN akan memotong fungsi keanggotaan keluaran setelah melalui operator AND sehingga didapatkan daerah fuzzy. Ketiga proses tersebut juga diterapkan pada aturan-aturan fuzzy berikutnya. Setelah semua aturan fuzzy telah dieksekusi, dilakukan proses komposisi dengan metode MAX yaitu solusi himpunan fuzzy diperoleh dengan cara mengambil nilai maksimum aturan, kemudian menggunakannya untuk memodifikasi daerah fuzzy, dan mengaplikasikan ke *output* dengan menggunakan operator OR (*union*). Jika proposisi telah dievaluasi, maka *output* akan berisi suatu himpunan fuzzy yang merefleksikan kontribusi dari tiap-tiap proposisi. Setelah proses implikasi dan komposisi telah dilakukan maka proses selanjutnya adalah proses defuzzifikasi. (Sutikno ; 2010 : 3).

II.2.2. Kendali Logika Fuzzy

Dalam pendekatan kendali berbasis logika fuzzy, masukan, keluaran, dan tanggapan kendali dispesifikasikan sesuai dengan keahlian seorang pakar serta pemodelan matematika terhadap suatu sistem kendali tidak dibutuhkan. Prinsip

dalam mendesain kendali logika fuzzy adalah mengaturatur parameter fungsi keanggotaan dan kaidah atur fuzzy. (Sutikno ; 2010 : 3).

Sebuah pengendali logika fuzzy umumnya dapat dimasukkan dalam sistem pengendali kalang tertutup. Pada Gambar 5 memperlihatkan system kendali kalang tertutup dengan pengendali logika fuzzy dimana E (*error*) dan dE (*delta error*/ perubahan *error*) merupakan masukan pengendali logika fuzzy dan U adalah besaran yang diberikan pada *plant*. Masukan *error* didapatkan dari nilai referensi dikurangi dengan nilai keluaran dari *plant* yang dinotasikan oleh persamaan 5.

$e(k) = r(k) – c(k)$ …… (5)

dimana:

e(k) adalah besar nilai kesalahan diskrit.

r(k) adalah besar nilai referensi diskrit. c(k) adalah besar nilai keluaran *plant* diskrit. Nilai masukan beda kesalahan (perubahan *error* pada sistem kontinyu) didapatkan dari nilai kesalahan sekarang dikurangi dengan nilai kesalahan sebelumnya yang dinotasikan oleh persamaan 6.

$dE(k) = e(k) – e(k-1)$ ……(6)

dimana:

dE(k) adalah besar nilai beda kesalahan diskrit.

e(k) adalah besar nilai kesalahan diskrit.

e(k-1) adalah besar nilai kesalahan diskrit sebelumnya. (Sutikno ; 2010 : 4).

II.3. Data, Informasi, dan Pengetahuan

Data merupakan representasi fakta mengenai suatu objek atau kejadian. (kusrini;2008:4) Misalnya:

1. Data mengenai biodata seseorang

   Nama : Paijo

   Alamat : Sukoharjo

   Jenis Kelamin : Laki-Laki

2. Data mengenai identitas suatu barang

   Nama : Kursi

   Bahan : Kayu Kamper

   Warna : Hijau Tua

3. Data mengenai suatu transaksi penjualan

   Nota : 05

   Tanggal : 1 Januari 2008

   Pembeli : Paijo

   Barang : Kursi, Meja

   Harga : Rp. 200.000,- : Rp. 500.000.-

   Jumlah : 4 : 1

Informasi merupakan data yang sudah diolah sedemikian rupa sehingga sesuai dengan yang dibutuhkan oleh penggunanya, sebagai contoh :

1. Informasi pelanggan yang sering membeli di Toko X (sebagai dasar penentuan pelanggan yang akan diberi bingkisan untuk lebaran).

Tabel II.1. Data Pelanggan

Nama	Alamat	Jenis Kelamin
Paijo	Sukoharjo	Laki-Laki
Imin	Bantul	Laki-Laki
Tentrem	Turi	Perempuan

(Sumber : Kusrini ; 2010 : 4)

2. Informasi mengenai jumlah keuntungan dari penjualan bulan Januari 2007 adalah keuntungan penjualan bulan Januari 2007 : Rp. 2.000.000,-

Pengetahuan merupakan saringan/intisari dari informasi. Pengetahuan ini lebih umum, tetapi mungkin tidak komplet dan lebih fuzzy.

Pengetahuan bisa berisi fakta, informasi, konsep, prosedur, model dan heuristic yang dapat digunakan untuk menyelesaikan suatu masalah. Contoh pengetahuan di antaranya :

1. Orang yang beralamat Sukoharjo suka mebel dengan warna-warni cerah
2. Jika seseorang membeli meja, kemungkinan besar akan membeli kursi juga

Pengetahuan diklasifikasikan menjadi :

1. Pengetahuan procedural (*procedural knowledge*) lebih menekankan pada bagaimana melakukan sesuatu. Contoh :
    a) Pengetahuan tentang bagaimana mencuci dengan menggunakan mesin.
    b) Pengetahuan tentang bagaimana membuat puding.
    c) Pengetahuan tentang bagaimana cara mengobati luka bakar.
2. Pengetahuan deklaratif (*declarative knowledge*) menjawab pertanyaan apakah sesuatu bernilai salah atau benar. Contoh :
    a) Jangan berikan pisau pada anak di bawah umur 3 tahun.
    b) Buah apel berwarna hijau dan berbentuk bulat.

c) Ada asosiasi positif antara merokok dan kanker.

3. Pengetahuan tacit (*tacit knowledge*) pengetahuan yang tidak bisa diungkapkan dengan bahasa

   a) Bagaimana cara mengayuh sepeda

   b) Bagaimana cara berjinjit untuk mencari balet

Pengetahuan bisa dimasukkan secara manual, semi otomatis maupun otomatis. Secara manual pengetahuan dapat diperoleh dari hasil wawancara, pelacakan proses penalaran ataupun observasi. Secara semiotomatis pengetahuan dimasukkan dengan sedikit bantuan dari *knowledge engineer*, tetapi sumbernya masih dari pakar, sedangkan cara otomatis dapat dilakukan dengan minimal input dari *knowledge engineer* maupun dari pakar.

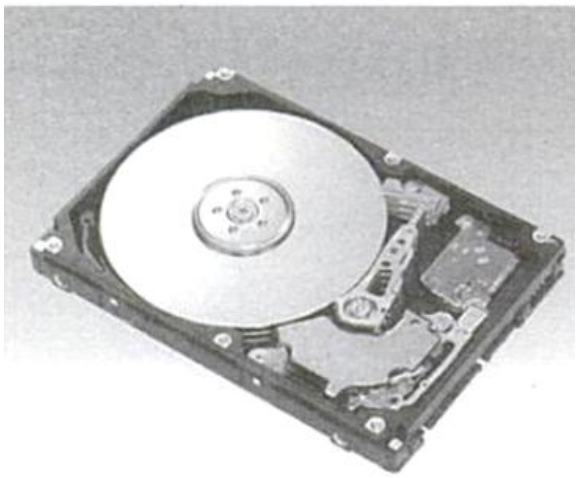
II.4. Hard disk

Hard disk atau biasa disebut juga sebagai hard drive, *fixed disk*, HDD, atau cukup hard disk saja, adalah media yang digunakan untuk menyimpan *file system* dan data dalam komputer. Hard disk terdiri atas tiga bagian utama, yaitu piringan magnetic, bagian mekanis, serta *head* untuk membaca data. Piringan tersebut digunakan untuk menyimpan data, sedangkan bagian mekanis bertugas memutar pringan tersebut (wahana komputer ; 2010 : 1).

Piringan data pada hard disk disebut dengan *platter*. Pada kedua sisi *platter* dilapisi dengan suatu material yang dirancang agar bisa menyimpan informasi secara magnetis. Platter disusun dengan melubangi tengahnya dan disusun pada suatu spindle. *Platter* berputar dengan kecepatan sangat tinggi yang

dikendalikan oleh *spindle* motor yang terhubung pad spindle. Alat elektromagnetik baca tulis khusus yang bernama *head* terpasang pada *slider* dan digunakan untuk menyimpan informasi ke dalam piringan atau membacanya.

Slider terpadang di atas arm, yang kesemuannya terhubung secara mekanis pada suatu kumpulan tunggal dan tersambung pada permukaan piringan melalui suatu alat yang disebut dengan actuator. Selain itu ada juga *logic board* mengatur aktifitas komponen lain dan berkomunikasi dengan PC. Adapun bentuk mekanik sebuah hard disk dapat dilihat pada gambar II.2.



Gambar II.2. Bentuk Hard disk
(Sumber : Wahana Komputer ; 2010 : 2)

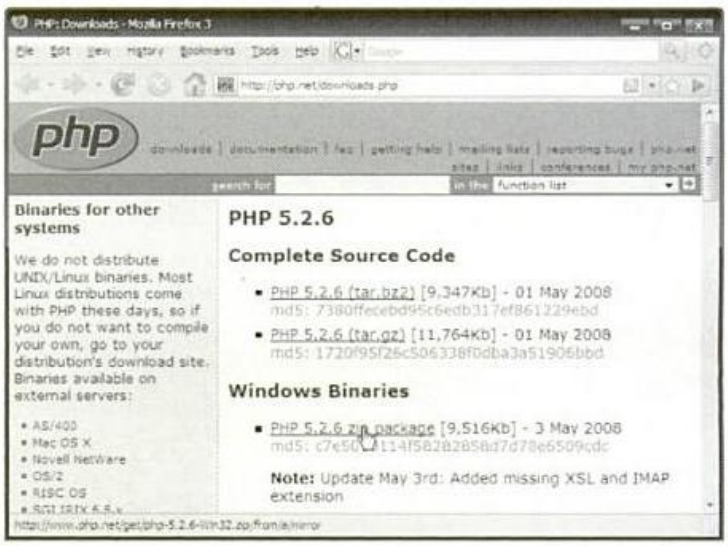
II.5. Desain atau Perancangan Sistem

Desain atau perancangan dalam pengembangan perangkat lunak merupakan upaya untuk mengkontruksi sebuah sistem yang memberikan kepuasan (mungkin informal) akan spesifikasi kebutuhan fungsional memenuhi target, memenuhi kebutuhan secara implisit dan eksplisit dari segi performansi

maupun penggunaan sumber daya, kepuasan batasan pada proses desain dari segi biaya, waktu, dan peranggkat. (Rosa A.S, M. Shalahuddin ; 2011 : 21).

II.6. Mengenal *Server* Environment

Kelebihan PHP yang paling terasa adalah tersedianya PHP parser di banyak platform. Anda bisa menjalankan skrip PHP di banyak *server*, seperti Apache dan IIS dan di banyak sistem operasi. Untuk melihat halaman download PHP dapat dilihat pada Gambar II.3.

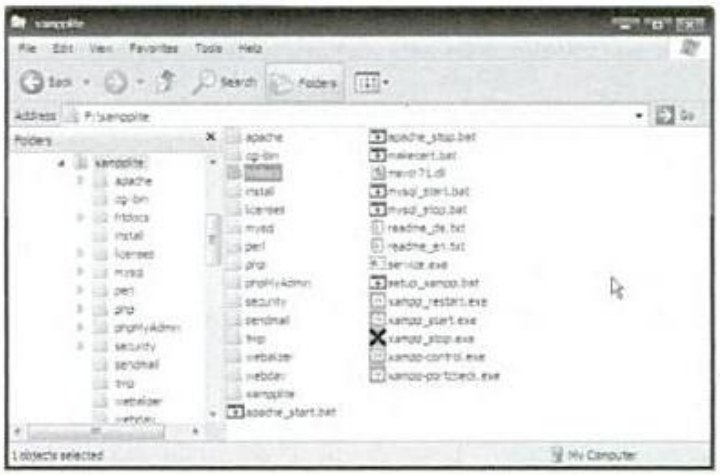


Gambar II.3. Halaman PHP
(Sumber : Ali Zaki ; 2010 : 32)

II.7. Instalasi Server Environment

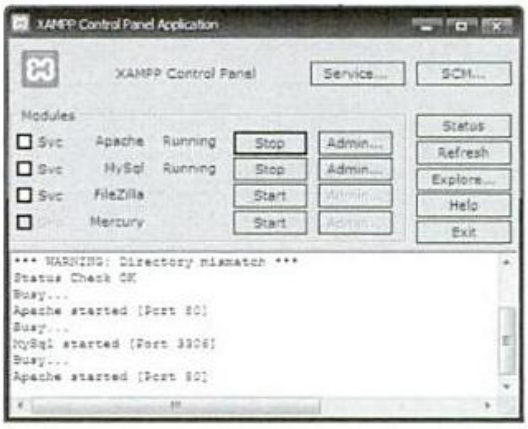
File *xampp Lite* yang sudah di *download* berupa file 7zip *executable*. Eksekusi file tersebut dan kemudian tentukan tempat tujuan ekstraksi dengan mengisikannya di kota *Extract to*.

Xampplite akan langsung terekstrak ke tempat tujuan. Ada banyak file di dalam *folder xampplite*. *Folder* penting adalah htdocs di mana file-file halaman web harus diletakkan di situ. Yang kedua adalah file xampp-control.exe yang berguna untuk mengatur perilaku *server*, seperti mengaktifkan dan menonaktifkan komponen-komponen tertentu. Untuk melihat tempat foler terinstall xampplite dapat dilihat pada gambar II.4.



Gambar II.4. Tempat Penyimpanan xampplite
(Sumber : Ali Zaki ; 2010 : 35)

Agar bisa membuat kode PHP dieksekusi oleh server, Anda perlu mengaktifkan modul Apache dan MySQL dari xampp-control.exe. Klik tombol Start untuk mengaktifkan modul tersebut. Dapat dilihat pada gambar II.5.



Gambar II.5. Mengaktifkan PHP dan MySQL
(Sumber : Ali Zaki ; 2010 : 35)

II.8. Mengenal MySQL

MySQL merupakan *database server open source* yang cukup populer keberadaannya. Dengan berbagai keunggulan yang dimiliki, membuat software database ini banyak digunakan oleh para praktisi untuk membangun suatu project. Adanya fasilitas API (*Application Programming Interface*) yang dimiliki oleh MySQL, memungkinkan bermacam-macam aplikasi komputer yang ditulis dengan berbagai bahasa pemrogramman dapat mengakses basis data MySQL (Wahana Komputer ; 2010 : 2).

II.9. *Database*

Database adalah sebuah struktur yang umumnya terbagi dalam 2 hal, yaitu sebuah *database flat* dan sebuah *database relasional*. *Database* relasional lebih mudah dipahami daripada *database flat* karena database relasional mempunyai bentuk yang sederhana serta mudah dilakukan operasi data. MySQL sendiri

adalah sebuah database relasional. Database yang memiliki struktur relasional terdapat tabel-tabel untuk menyimpan data. Pada setiap tabel terdiri dari kolom dan baris serta sebuah kolom untuk mendefinisikan jenis informasi apa yang harus disimpan (Wahana Komputer ; 2010 : 2).

Mengapa menggunakan database, itu pertanyaan yang akan keluar dari pikiran Anda pada saat pertama kali ingin mempelajari database. Database akan menjadi sangat berguna saat Anda perlu menyimpan informasi yang dikategorikan secara logis. Contoh, jika Anda ingin menyimpan informasi tentang PT. Wahana Komputer dengan database, Anda bisa mengelompokkan berbagai hal dalam bisnis menjadi beberapa tabel.

II.10. UML (*Unified Modelling Language*)

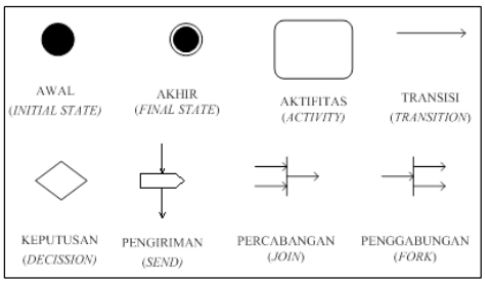
Pemodelan perangkat lunak bekerja dengan cara yang cukup serupa layaknya seorang arsitek atau insinyur teknik sipil yang akan membuat sebuah bangunan / gedung berskala besar. Saat seorang arsitek atau insinyur teknik sipil akan membuat sebuah bangunan / gedung berskala besar, ia biasanya membuat denah-denah atau maket-maket yang menggambarkan bentuk jadi dari bangunan / gedung. Kita sebagai seorang perancang sistem perangkat lunak juga bertindak dengan cara yang serupa, hanya saja yang kita rancang bukan bangunan, melainkan sistem perangkat lunak. Menggambarkan komponen-komponen sistem perangakt lunak dalam bentuk-bentuk geometri tertentu misalmya untuk menggambarkan suatu kelas (class) dalam aplikasi, menggunakan antarkelas (asosiasi), menggunakan garis lurus (Adi Nugroho ; 2009 : 6).

II.10.1. Use Case Diagram

Dalam konteks UML, tahap konseptualisasi dilakukan dengan pembuatan use case diagram yang sesungguhnya merupakan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunanya. Selanjutnya, use case diagram tidak hanya sangat penting pada tahap analisis, tetapi juga sangat penting untuk perancangan (*design*), untuk mencari (mencoba menemukan) kelas-kelas yang terlibat dalam aplikasi, dan untuk melakukan pengujian (testing) (Adi Nugroho ; 2009 : 7).

II.10.2. Diagram *Activity*

Activity diagram menggambarkan urutan aktifitas yang digunakan untuk menjelaskan aktifitas dari sebuah operasi. Pada activity diagram terdapat keadaan aksi yang berisi spesifikasi dari aktifitas tertentu. Diagram ini berisi, pilihan keputusan dan kondisi serta spesifikasi message yang dikirim atau diterima sebagai gambaran dari aksi.



Gambar II.6. Diagram Activity
(Sumber : Tarbudi ; 2011 : 10)

III.10.3. *Sequence*

Diagram interaksi yang menekankan pada waktu pengiriman *message*. *Sequence* diagram menunjukkan sekumpulan objek dan pengiriman serta

penerimaan *message* antar objek. Objek yang umumnya memiliki nama atau instansiasi dari *class*, tapi dapat pula merupakan turunan dari *things* lain, seperti *collaboration*, *component* dan *node*. Diagram ini digunakan untuk mengilustrasikan *dynamic view* dari sistem.

II.10.4. *State*

Sebuah *state* diagram menggambarkan keadaan mesin, transisi, *event* dan *activity*. Diagram ini adalah pelengkap khusus untuk mendeskripsikan sebuah *class* yang menggambarkan *state* dari objek dari *class* dan *event* yang menyebabkan *state* berubah. *Event* tersebt dapat berasal dari objek yang mengirimkan suatu *message* atau dari kondisi yang terpenuhi.

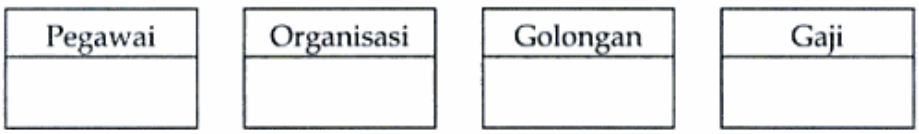
II.11. *Entity Relationship Diagram*

Entity Relationship Diagram (ERD) digunakan untuk mengidentifikasi data yang akan diambil, disimpan, dan dipanggil kembali (*regrieve*) untuk keperluan-keperluan tertentu dalam mendukung kegiatan yang dilakukan oleh organisasi. ERD digunakan untuk mengidentifikasi asal data yang dibutuhkan dan dilaporkan (Marimin, Hendri Tanjung, Haryo Prabowo ; 2010 : 111).

ERD (model data) merupakan alat yang digunakan dalam analisis untuk menggambarkan kebutuhan data dan asumsi-asumsi dalam sistem yang akan dibangun/dikembangkan secara terstruktur dari atas ke bawah. Model data ini juga diatur pada tahapan SDLC dalam mendesain database. Pembuatan ERD membutuhkan pemahaman terhadap sistem dan komponen-komponen yang menyusunnya.

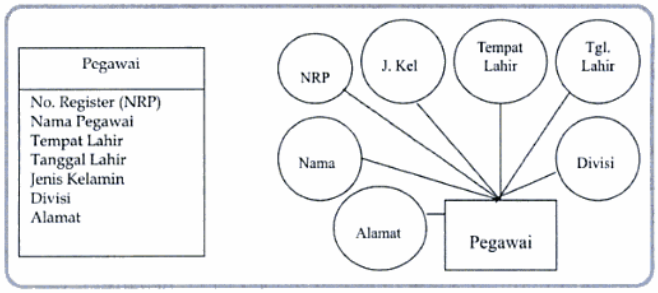
Untuk mempermudah dalam perancangan database, maka digunakan *Entity Relationship Diagram* (ERD). ERD diutamakan untuk permodelan dari desain konseptual. *Entity Relationship Diagaram* menggambarkan struktur dan keterkaitan tabel-tabel data yang menyusun *database* secara detail. ERD merupakan representasi data sebagai entitas, atribut, dan relasi.

Entitas menggambarkan kumpulan dari segala data, misalnya entitas pegawai berisi kumpulan data seluruh pegawai yang bekerja pada suatu organisasi. Entitas biasanya dilambangkan dengan menggunakan kotak segi empat seperti ditunjukkan Gambar II.7.



Gambar II.7. Contoh Pembuatan Entitas
(Sumber : Marimindkk ; 2010 : 112)

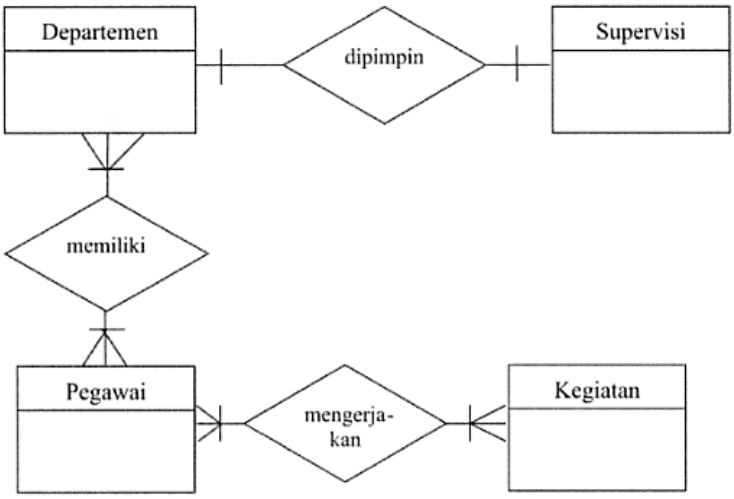
Entitas selanjutnya dijelaskan dengan atribut-atribut yang ada di dalamnya atau sering kali disebut elemen data. Atribut atau elemen data merupakan unit terkecil dari data yang dapat menjalaskan apa yang dimiliki oleh suatu entitas (karakteristik dari entitas). Misalnya entitas pegawai memiliki atribut yang terlihat pada Gambar II.8. Sedangkan relasi menjelaskan keterkaitan di antara dua entitas yang berbeda misalnya pegawai bekerja pada suatu departemen.



Gambar II.8. Contoh Atribut atau Elemen Data Suatu Entitas
(Sumber : Marimin dkk ; 2010 : 112)

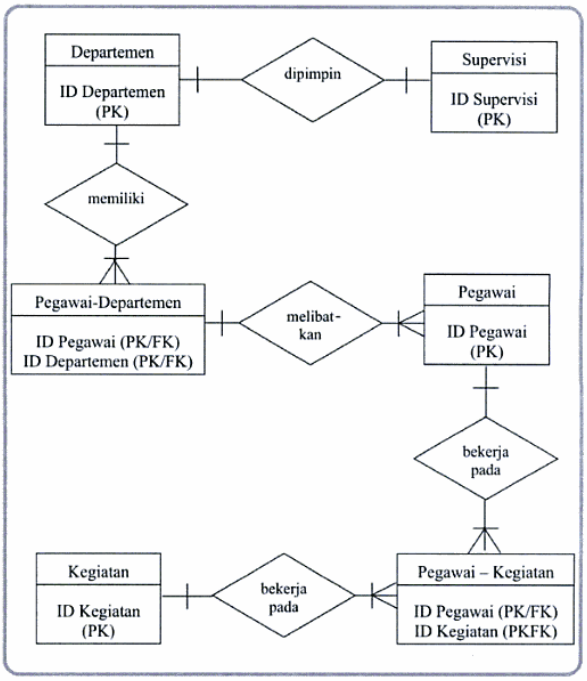
Sifat hubungan atau relasi antarentitas dapat dibedakan menjadi tiga jenis, yaitu hubungan satu ke satu (*one to one relationship*), satu kebanyak (*one to many relationship*) dan banyak ke banyak (*many to many relationship*).

One to one relationship akan terjadi jika setiap entitas dalam suatu himpunan entitas hanya berhubungan dengan satu entitas pada himpunan entitas lain dan sebaliknya. Sebagai contoh, setiap departemen dipimpin oleh seorang supervise dan seorang supervise hanya memimpin pada sebuah departemen, lihat pada Gambar II.9.



Gambar II.9. Contoh One to One dan Many to Many Relationship
(Sumber : Marimin, dkk ; 2010 : 113)

One to many relationship terjadi jika setiap entitas dalam suatu himpunan entitas dapat berhubungan dengan beberapa entitas pada himpunan entitas lain tetapi tidak sebaliknya. Misalnya seorang pegawai hanya bekerja pada sebuah departemen, sedangkan departemen memiliki banyak pegawai, lihat pada Gambar II.10.



Gambar II.10. Contoh Relasi One to One, One to Many dan Many to Many
(Sumber : Marimin, dkk ; 2010 : 114)

Relasi many to many harus dipisahkan dengan cara memberikan entitas tambahan di antara kedua entitas yang ada, sehingga akan menjadikan relasi tersebut menjadi dua relasi *One to Many*. Gambar II.9 memperlihatkan bahwa sebelumnya terdapat relasi *Many to Many* antara departemen dan pegawai serta antara pegawai dan pekerjaan. Berikan dengan adanya relasi *Many to Many*, maka hubungan tersebut harus dipisahkan, sehingga menjadi relasi yang bersifat *one to many.*