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;2008:3).

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. Metode Inferensi

Inferensi merupakan proses untuk menghasilkan informasi dari fakta yang diketahui atau diasumsikan. Inferensi adalah konklusi logis (*logical conclusion*) atau implikasi berdasarkan informasi yang tersedia. Dalam sistem pakar proses inferensi dilakukan dalam suatu modul yang disebut *Inference Engine* (mesin inferensi)

Ketika representasi pengetahuan (RP) pada bagian *knowledge base* telah lengkap, atau paling tidak telah berada pada level yang cukup akurat, maka RP tersebut telah siap digunakan *Inference engine* merupakan modul yang berisi program tentang bagaimana mengendalikan proses *reasoning* (kusrini;2008:8).

Ada dua metode inferensi yang penting dalam sistem pakar yaitu runut maju (*forwarding chaining*) dan runut balik (*backward chaining*).

II.2.1. Runut Maju (*Forward Chaining*)

Runut maju berarti menggunakan himpunan aturan kondisi aksi. Dalam metode ini, data digunakan untuk menentukan aturan mana yang akan dijalankan, kemudian aturan tersebut dijalankan. Mungkin proses menambahkan data ke memori kerja. Proses diulang sampai ditemukan suatu hasil.

Metode inferensi runut maju cocok digunakan untuk menangani masalah pengendalian (*controlling*) dan peramalan (*prognosis*). Untuk memudahkan pemahaman mengenai metode ini, akan diberikan ilustrasi kasus pembuatan sistem pakar sebagai berikut :

Ingin diperoleh konklusi dari daftar konklusi yang ada berdasarkan premis-premis dalam aturan dan fakta yang diberikan oleh user. Berikut ini adalah daftar aturannya :

Aturan 9 :

Jika Premis 1

Dan Premis 2

Dan Premis 3

Maka Konklusi 1

Aturan 10 :

Jika Premis 1

Dan Premis 3

Dan Premis 4

Maka Konklusi 2

Aturan 11 :

Jika Premis 2

Dan Premis 3

Dan Premis 5

Maka Konklusi 3

Aturan 12 :

Jika Premis 1

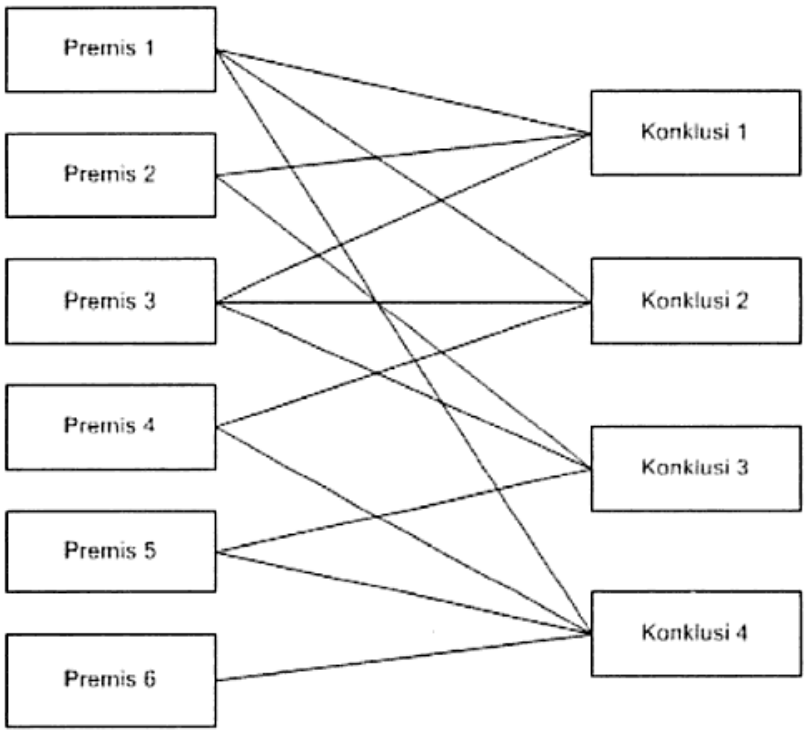
Dan Premis 4

Dan Premis 5

Maka konklusi 4

Jika aturan ini kita gambarkan sebagai sebuah graph yang memetahkan antara premis-premis dan konklusi-konklusi akan tampak seperti Gambar II.1.

Penelusuran maju pada kasus ini adalah untuk mengetahui apakah suatu fakta yang dialami oleh pengguna itu termasuk konklusi 1, konklusi 2, konklusi 3, atau konklusi 4 atau bahkan bukan salah satu dari konklusi tersebut, yang artinya sistem mampu mengambil kesimpulan karena keterbatasan aturan.



Gambar II.1. *Graph* Pengetahuan
(Sumber : Kusrini;2008;10)

Dalam penalaran ini, *user* diminta memasukkan premis-premis yang dialami. Untuk memudahkan pengguna, sistem dapat memunculkan daftar premis yang mungkin sehingga *user* dapat memberikan umpan balik premis mana yang dialami dengan memilih satu atau beberapa dari daftar premis yang tersedia. Berarti daftar premisnya adalah :

Premis1, Premis2, Premis3, Premis4, Premis5, Premis6

Berdasarkan premis-premis yang dipilih, maka sistem akan mencari aturan yang sesuai, sehingga akan diperoleh konklusinya. Seandainya *user* memilih Premis 1, Premis 2 dan Premis 3 maka atura yang terpilih adalah aturan 1 dengan konklusinya adalah Konklusi 1. Seandainya *user* memilih Premis 1 dan Premis 6, amak sistem akan mengarah pada aturan 4 dengan konklusinya adalah konklusinya 4, tetapi karena aturan tersebut premisnya adalah premis 1, premis 4, premis 5 dan premis 6, maka premis-premis yang dipilih oleh *user* tidak cukup untuk mengambil kesimpulan konklusi 4 sebagai konklusi terpilih.

II.2.2. Runut Balik (*Backward Chaining*)

Runut balik merupakan metode penalaran kebalikan dari runut maju. Dalam runut balik penalaran dimulai dengan tujuan kemudian merunut balik ke jalur yang akan mengarahkan ke tujuan tersebut. Runut balik disebut juga sebagai *goal driver reasoning*, merupakan cara yang efisien untuk memecahkan masalah yang dimodelkan sebagai masalah pemilihan terstruktur.

Tujuan inferensi adalah mengambil pilihan terbaik dari banyak kemungkinan. Metode inferensi runut balik ini cocok digunakan untuk memecahkan masalah diagnosis. Dengan menggunakan kasus yang sama pada

proses penalaran runut maju, yang ingin didapatkan pada penalaran ini juga sama yaitu salah satu konklusi dari konklusi 1, konklusi 2, konklusi 3, konklusi 4 atau bahkan tidak dari keempat konklusi tersebut.

Penelusuran didasarkan pada suatu keyakinan bahwa ada kemungkinan konklusi dari daftar konklusi merupakan salah satu tujuan atau konklusi terpilih berdasarkan fakta yang diberikan oleh *user*. Sistem dengan urutan tertentu akan mengambil sebuah konklusi sebagai calon konklusinya. Misal urutannya adalah sesuai dengan urutan konklusi. Awalnya sistem akan mengambil hipotesis bahwa konklusinya adalah konklusi 1. Untuk membuktikan hipotesisnya, Sistem akan mencari premis-premis aturan yang mengandung konklusi 1. Setelah itu sistem akan meminta umpan balik kepada *user* mengenai adalah premis 1, premis 2 dan premis 3, maka sistem akan mencari tahu apakah *user* memilih premis-premis tersebut.

Cara untuk mengambil umpan balik dari *user* bisa dilakukan dengan mencari dari daftar premis yang dipilih oleh *user* atau dengan menanyakan satu per satu premis-premis yang seharusnya dipilih.

Jika ternyata ada premis yang tidak terpilih oleh *user* maka hipotesis terhadap konklusi tersebut gugur, yang artinya fakta yang dimasukkan *user* konklusinya bukan konklusi 1. Oleh karena itu, sistem akan melanjutkan hipotesis ke konklusi berikutnya. Demikian seterusnya sampai ditemukan konklusi yang semua premis dalam aturannya terpilih.

Jika sampai akhir konklusi yang mungkin tidak ada premis yang terpenuhi, maka sistem akan mengambil kesimpulan bahwa konklusinya adalah di luar

pengetahuannya, yang artinya sistem tidak menemukan solusi untuk premis-premis pilihan user.

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;2008:4)

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 :
   - Pengetahuan tentang bagaimana mencuci dengan menggunakan mesin.
   - Pengetahuan tentang bagaimana membuat puding.
   - Pengetahuan tentang bagaimana cara mengobati luka bakar.

2. Pengetahuan deklaratif (*declarative knowledge*) menjawab pertanyaan apakah sesuatu bernilai salah atau benar. Contoh :
   - Jangan berikan pisau pada anak di bawah umur 3 tahun.
   - Buah apel berwarna hijau dan berbentuk bulat.
   - Ada asosiasi positif antara merokok dan kanker.
3. Pengetahuan tacit (*tacit knowledge*) pengetahuan yang tidak bisa diungkapkan dengan bahasa
   - Bagaimana cara mengayuh sepeda
   - 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. Penanganan Data Tidak Lengkap

Teknik penyelesaian yang disebutkan di sini merupakan cara dasar dengan asumsi bahwa untuk mengambil kesimpulan semua fakta harus terpenuhi. Untuk kasus-kasus tertentu, seorang pakar akan tetap dapat mengambil kesimpulan konklusinya meskipun tingkat keyakinannya tidak 100% . Kita ambil contoh masalah yang biasa diputuskan oleh menyimpulkan bahwa seorang pasien akan mengalami suatu penyakit haruslah mengalami suatu gejala, ada kalanya premis

yang harusnya diketahui tidak bisa diketahui karena hal-hal tertentu. Oleh karena itu, dokter akan mengambil kesimpulan pasien mengalami penyakit tersebut dengan derajat kepercayaan tidak 100%.

Besarnya derajat kepercayaan untuk menentukan konklusi dapat ditentukan dengan suatu model rumus. Rumus yang terbaik tentu saja rumus yang dipakai oleh pakarnya. Oleh karena itu seorang *knowledge engineer* harus benar-benar jeli dalam merumuskan cara pikir pakar dari masalah yang sedang ingin dipecahkan dengan sistem pakar. Dalam buku ini yang diberikan adalah metode factor kepastian (*certainty factor*).

II.5. Representasi Pengetahuan

Representasi pengetahuan merupakan metode yang digunakan untuk mengkodekan pengetahauan dalam sebuah sistem pakar. Representasi dimaksudkan untuk menangkap sifat-sifat penting masalah dan membuat informasi itu dapat diakses oleh prosedur pemecahan masalah (kusrini;2008:6). Adapun karakteristik dari metode representasi pengetahuan adalah :

1. Harus bisa deprogram dengan bahasa pemrograman atau dengan *shells* dan hasilnya disimpan dalam memori.
2. Dirancang sedemikian sehingga isinya dapat digunakan untuk proses penalaran.
3. Model representasi pengetahuan merupakan sebuah struktur data yang dapat dimanipulasi oleh mesin inferensi dan pencarian untuk aktivitas pencocokan pola.

Seperti telah disampaikan, bahwa dalam sistem pakar ada beberapa metode representasi pengetahuan. Jika pengetahuan berupa pengetahuan yang bersifat *deklaratif*, maka metode representasi pengetahuan yang cocok adalah jaringan *semantic*, *frame* dan logika predikat. Tetapi jika pengetahuannya berupa pengetahuan *procedural* yang merepresentasikan aksi dan prosedur, maka metode representasi pengetahuan yang cocok adalah kaidah produksi.

II.6. Pengertian Jaringan

Jaringan komputer adalah sistem yang terdiri dari komputer-komputer, serta piranti-piranti yang saling terhubung sebagai satu kesatuan. Dengan dihubungkan pirang-piranti tersebut, alhasil dapat saling berbagi sumber daya antar satu piranti dengan piranti lainnya (Wahana Komputer;2010;2).

Dalam istilah komputer, jaringan merupakan penghubung antara dua komputer atau lebih yang tujuan utamanya adalah berbagi data. Betulkah jaringan komputer itu *hardware* dan *software*? Jawabnya adalah betul. Jaringan komputer adalah gabungan antara *hardware* dan *software*.

Jaringan komputer bisa diklasifikasikan menurut beberapa kategori, nanti akan dijelaskan lebih lanjut di sub bab mengenal jenis jaringan komputer. Karena sebuah sistem, jaringan komputer terdiri atas komponen-komponen, dan perangkat jaringan lainnya yang bekerja bersama-sama untuk mencapai suatu tujuan yang sama. Tujuan dari jaringan komputer adalah :

1. Membagi sumber daya, contohnya berbagi pemakaian printer CPU, memori, harddisk

2. Komunikasi, contohnya suatu elektronik, *instant messaging*, *chatting*.
3. Akses informasi, contohnya *web browsing*.

Agar dapat mencapai tujuan yang sama, setiap bagian dari jaringan kompputer meminta dan memberikan layanan (*service*). Pihak yang meminta layanan disebut *client* dan yang memberikan layanan disebut *server*. Arsitektur ini disebut dengan sistem *client server*, dan digunakan pada hampir seluruh jaringan komputer.

II.6.1. Jenis Jaringan

Jaringan komputer mempunyai berbagai macam tipe yang masing-masing memiliki kelebihan dan kekurangan. Agar lebih jelas, berikut penjabarannya (Wahana Komputer;2010;3).

1. Berdasarkan Ruang Lingkup

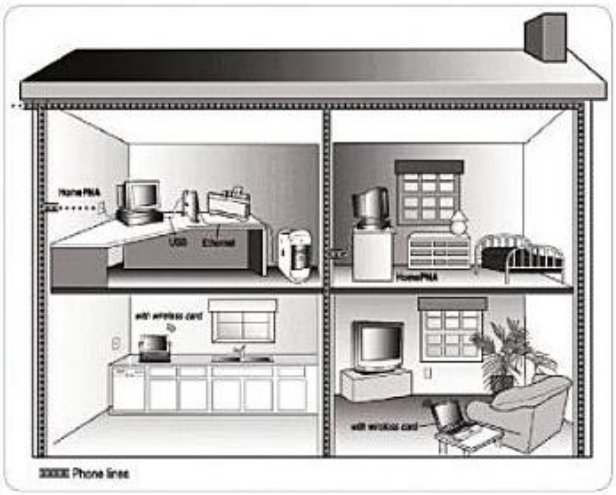
Ada banyak tipe jaringan komputer, itu juga bisa dibedakan berdasarkan beberapa parameter yang berbeda. Parameter pertama adalah berdasarkan ruang lingkup. Ada beberapa tipe jaringan komputer berdasarkan ruang lingkupnya, yaitu :

- *Local Area Network* (LAN)
- *Metropolitan Area Network* (MAN)
- *Wide Area Network* (WAN)

a. LAN (*Local Area Network*)

LAN (*Local Area Network*) adalah sebuah jaringan komputer yang cakupan areanya kecil, seperti di sebuah rumah, kantor, atau sekolah. Karakteristik khusus dari LAN yang membedakan dengan jaringan WAN

adalah *transfer* data yang lebih besar, cakupan atau *geografis* yang lebih sempit dan tidak perlunya jalur komunikasi *leased line*. Teknologi yang dipakai untuk membuat LAN ada beberapa macam, ada *ARCNET* dan *Token Ring*. Namun, yang lazim digunakan sekarang adalah *Ethernet* dan kabel UTP Dapat dilihat pada Gambar II.2.



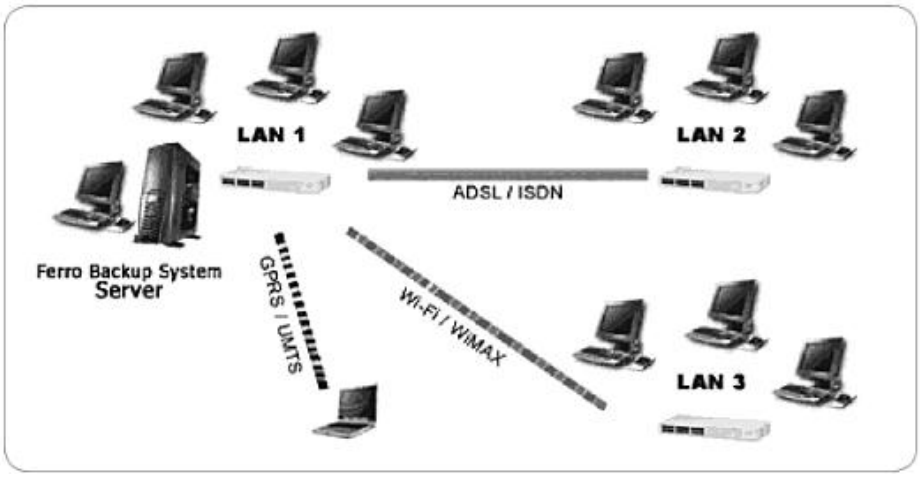
Gambar II.2. Jaringan LAN Didalam Rumah
(Sumber : Wahana Komputer;2010;3)

b. WAN (*Wide Area Network*)

Sebuah *Wide Area Network* (WAN) adalah jaringan komputer yang cakupannya cukup luas, seperti antar regional atau antar negara. Ada beberapa teknik koneksi yang biasanya dipakai untuk membuat *Wide Area Network*, (Wahana Komputer;2010;4) yaitu :

- *Leased line*, koneksi *point to point* antara 2 komputer atau *local area network* (LAN).
- *Circuit switching*, jalur sirkuit yang *dedicated* yang diciptakan antara *end point*. Contohnya adalah koneksi dialup.

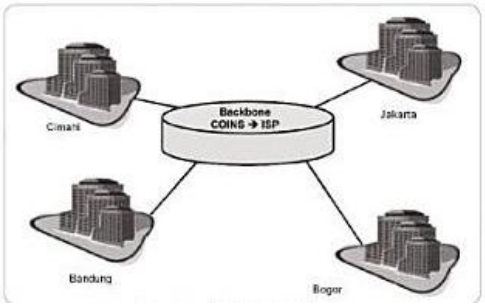
- *Packet switching*, transport paket melalui *point to point* atau *point to multipoint* melalui *carrier internetwork.*
- *Cell relay* mirip dengan *packet switching*, tetapi menggunakan sel dengan panjang yang tatap serta bukan sel yang panjangnya variable. Data dibagi menjadi beberapa sel dengan panjang tetap, kemudian ditransportasikan melalui sirkuit virtual. Dapat dilihat pada Gambar II.3.



Gambar II.3. Jaringan WAN
(Sumber : Wahana Komputer;2010;4)

c. MAN (*Metropolitan Area Network*)

Metropolitan Area Network (MAN) adalah jaringan komputer yang cakupan luasnya mencapai satu atau lebih kota. Sebuah MAN biasanya menghubungkan antara beberapa LAN lokal menggunakan teknologi *backbone*, seperti *fiber optic*, dan menyediakan layanan ke banyak jaringan seperti untuk internet. Dapat dilihat pada Gambar II.4.



Gambar II.4. Jaringan MAN
(Sumber : Wahana Komputer;2010;5)

II.6.2. Jaringan *Client Server*

Jaringan *Client Server* menghubungkan komputer server dengan komputer *client/workstation*. Komputer *Server* adalah komputer yang menyediakan fasilitas bagi komputer-komputer *client/workstation* yang terhubung dalam jaringan. Sedangkan komputer adalah komputer yang menggunakan fasilitas yang disediakan oleh komputer *server*. Komputer *server* pada sebuah jaringan tipe *client server* disebut dengan *dedicated server*, karena komputer yang digunakan hanya sebagai penyedia fasilitas untuk komputer *client/workstation*. Komputer *server* tidak dapat berperan sebagai komputer *client/workstation*.

Keunggulan tipe jaringan *Client Server* adalah :

1. Terdapat Administrator jaringan yang mengelola sistem keamanan dan administrasi jaringan, sehingga sistem keamanan dan administrasi jaringan akan lebih terkontrol.
2. Komputer *server* difungsikan sebagai pusat data, komputer *client* dapat mengakses data yang ada dari komputer client manapun. Apabila terdapat

komputer *client* yang rusak, pengguna masih dapat mengakses data dari komputer *client* yang lain.

3. Pengaksesan data lebih tinggi karena penyediaan dan pengelolaan fasilitas jaringan dilakukan oleh komputer *server*. Dan komputer *server* tidak terbebani dengan tugas lain sebagai *workstation*.
4. Pada tipe jaringan *client server*, sistem *backup* data lebih baik, karena *backup* data dapat dilakukan terpusat di komputer *server*. Apabila data pada komputer *client/workstation* mengalami masalah atau kerusakan masih tersedia *backup* pada komputer *server*.

Kelemahan tipe jaringan *client server* adalah :

1. Biaya mahal, karena membutuhkan komputer yang memiliki kemampuan tinggi yang difungsikan sebagai komputer *server*.
2. Kelancaran jaringan tergantung pada komputer *server*. Bila komputer *server* mengalami gangguan maka jaringan akan terganggu.

II.7. Membangun Jaringan Sederhana Berbasis Kabel

Jaringan kabel atau *wired network* adalah tipe jaringan *konvensional*. Dinamakan *konvensional*, karena tipe jaringan ini ada sejak awal adanya sejarah jaringan komputer.

1. Membuat Desain Jaringan Rumah

Tahap awal untuk membuat jaringan kabel di rumah adalah dengan membuat desain jaringan rumah. Proses perancangan suatu jaringan merupakan langkah yang sangat berpengaruh terhadap kinerja suatu jaringan. Desain jaringan

merupakan fondasi bagi sebuah jaringan. Selain itu, desain yang akan dibuat juga menentukan perangkat keras yang dibutuhkan.

a. Menentukan Jumlah dan Penempatan PC

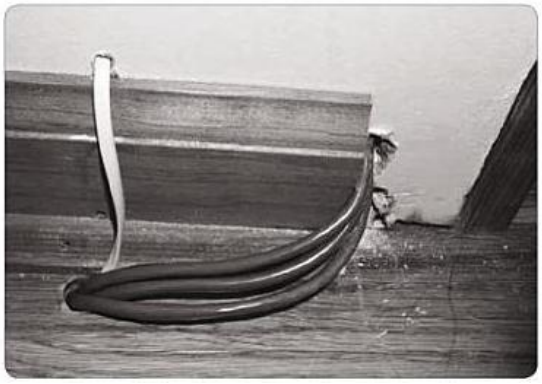
Pertama, tentukan jumlah dan penempatan PC. Jumlah PC menentukan pemilihan *switch* atau hub yang akan digunakan. Karena slot PC di hub atau *switch* ini harus memiliki jumlah yang mampu mengakomodasi jumlah PC ditambah beberapa slot lagi untuk menghubungkan dengan jaringan lain jika diperlukan, seperti ke *modem* atau ke *Access* Point jika ingin disambungkan dengan jaringan *WiFi*. Untuk lebih jelasnya dapat dilihat pada Gambar II.5.

Gambar II.5. Perangkat Switch
(Sumber : Wahana Komputer;2010;24)

Misalnya, Anda ingin menghubungkan 4 komputer, usahakan memilih *switch* atau hub yang punya minimal 8 slot. Selain untuk mengakomodasi komputer, juga untuk menghubungkan piranti-piranti lainnya.

Yang harus dipikirkan juga adalah bagaimana nanti kondisi fisik jaringan rumah Anda. Lazimnya sebuah rumah, akan terdiri dari beberapa ruang. Misalnya, di satu kamar, *user* 1 menggunakan komputer untuk

menyelesaikan pekerjaa, sementara di kamar lain *user* 2 menggunakan komputer untuk bermain internet atau untuk mengerjakan tugas sekolah. Karena jaringan kabel merupakan jaringan fisik, Anda harus mengatur penempatan kabel agar terlihat rapi. Misalnya, Anda bisa melubangi tembok untuk menghubungkan komputer satu dengan komputer lain. Untuk lebih jelasnya dapat dilihat pada Gambar II.6.

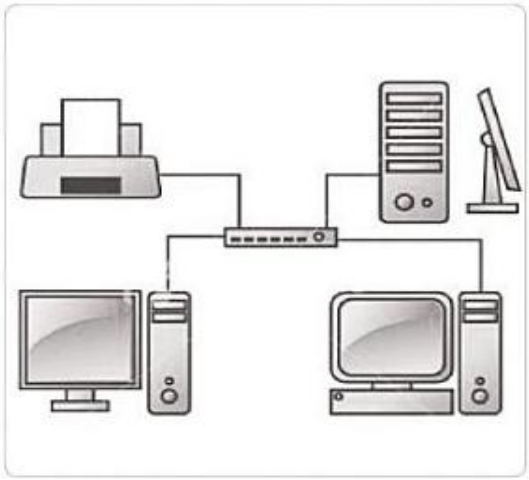


Gambar II.6. Mengebor Dinding Untuk Mendapatkan Lubang
(Sumber : Wahana Komputer;2010;25)

Selanjutnya dinding diberi tutup, untuk lebih jelasnya dapat dilihat pada Gambar II.7.

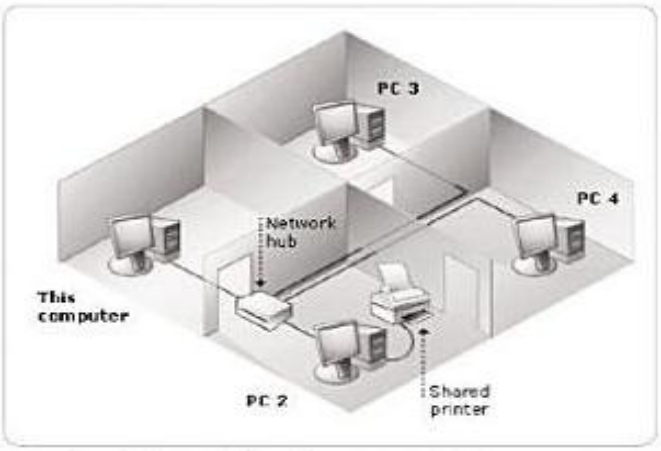
Gambar II.7. Lubang di Dinding Diberi Tutup
(Sumber : Wahana Komputer;2010;25)

Letakkan printer di tempat yang bisa diakses semua orang. Misalnya, di ruang tengah atau di ruang khusus komputer. Karena ini mempermudah penghuni rumah lain untuk mengakses printer. Dapat dilihat pada Gambar II.8.



Gambar II.8. Desain Jaringan Resource Printer
(Sumber : Wahana Komputer;2010;26)

Selanjutnya desain di beberapa ruangan, yang dapat dilihat pada Gambar II.9.



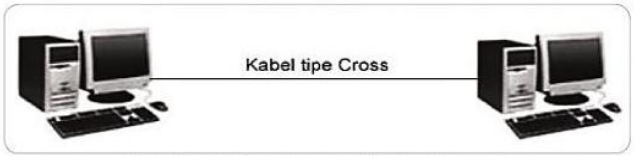
Gambar II.9. Desain Denah Komputer di Beberapa Ruangan
(Sumber : Wahana Komputer;2010;26)

2. Menentukan Tipe dan Topologi Jaringan Yang Digunakan

Topologi jaringan adalah bagian yang menjelaskan hubungan antar komputer yang dibangun berdasarkan kegunaan, keterbatasan *resource* dan keterbatasan biaya. Secara garis besar, topologi bisa dikelompokkan menjadi 2, yaitu jaringan 2 komputer dan lebih dari 2 komputer.

a. Menghubungkan Jaringan Dua Komputer

Jika Anda ingin membuat jaringan dengan 2 komputer dan tidak ada rencana untuk menambahkan komputer lagi ke dalam jaringan di masa depan, cara paling mudah adalah dengan menggunakan satu kabel UTP dan sepasang RJ-45 yang di konfigurasi secara *cross*. Dapat dilihat pada Gambar II.10.



Gambar II.10. Topologi menghubungkan 2 PC Ethernet ke Ethernet
(Sumber : Wahana Komputer;2010;27)

Kemudian Anda dapat langsung menghubungkan antara *slot Ethernet* satu PC dengan *slot Ethernet* PC lainnya tanpa menggunakan *switch* atau *hub*. Dapat dilihat pada Gambar II.11.

Gambar II.11. Kabel UTP dengan 2 Rj-45
(Sumber : Wahana Komputer;2010;27)

II.8. 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.9. Mengenal HTML

HTML memiliki beberapa kode yang memungkinkan *programmer* atau *desainer* mendesain halaman *web* yang tampilannya tidak seperti biasa, Anda akan belajar bagaimana membuat tabel, *frame*, memasukkan *JavaScript*, dan mengenal PHP.

II.9.1.Memahami Tabel di HTML

Tabel penting peranannya dalam halaman *web*, selain untuk menampilkan teks atau gambar dalam format lajur dan kolom, Anda bisa juga menggunakan tabel untuk membantu melayout tampilan halaman.

Tabel merupakan sebuah kotak yang terdiri dari baris (*row*) dan kolom (*column*). Untuk membuat tabel, Anda menggunakan tag <*tabel*> dan menutupnya dengan tag </*table*>. Anda bisa juga menambahkan atribut lain di tag <*table*> pembuka. Misalnya menentukan border, warna, dan sebagainya.

Di dalam tag <*table*> ada beberapa tag lain yang perlu Anda pahami, yaitu :

- Tag <tr>, artinya tag untuk menuliskan baris biasa di tabel, TR singkatan dari *table row*.
- Tag <td>. Artinya tag untuk menuliskan kotak di dalam baris, makanya tag <td> ada di dalam tag <tr>. TD singkatan dari *table* data.
- Tag <th> artinya tag untuk menuliskan kotak biasa seperti <td>, namun untuk *header* tabel. TH singkatan dari *table header*.

Berikut ini merupakan contoh kode yang menunjukkan cara pembuatan tabel. Untuk lebih jelasnya dapat dilihat pada Gambar II.12.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
  "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">


 <head>
   <title>Pembuatan Tabel</title>
 </head>
 <body>
 <h1> Tabel Nilai Siswa</h1>
   <table border="1" bgcolor="pink">
     <tr>
       <th>Nama</th>
       <th>Tempat Tanggal Lahir</th>
       <th>Nama Wali</th>
       <th>Jurusan</th>
       <th>Angkatan</th>
          </tr>
    <tr>
       <td>Arifinsyah Gayo</td>
       <td>Banda Aceh, 24 Januari 1980</td>
       <td>Agus Salim</td>
       <td>Pelayaran Niaga</td>
       <td>2003</td>
      </tr>
     <tr>
       <td>Muh Zaenuri</td>
       <td>Bandung, 10 September 1983</td>
       <td>Asep Suhendar</td>
       <td>Agronomi</td>
       <td>2008</td>
      </tr>
     <tr>
       <td>Nasihun Toha</td>
       <td>Kudus, 30 Agustus 1978</td>
```

Gambar II.12. Kode Pembuatan Tabel
(Sumber : Ali Zaki;2008;20)

II.10. 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.13.

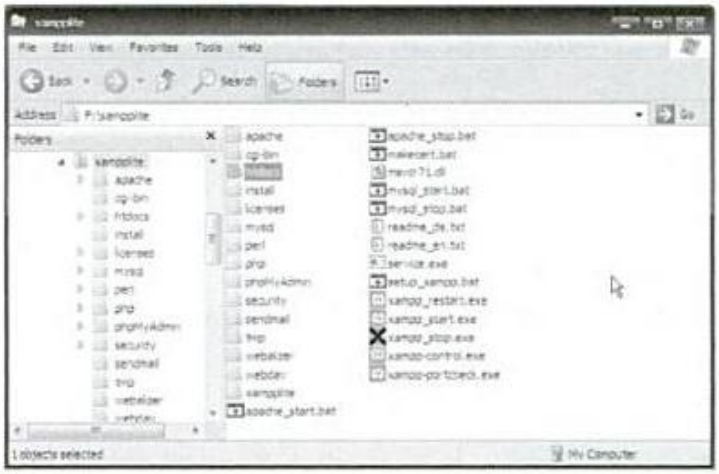
Gambar II.13. Halaman PHP
(Sumber : Ali Zaki;2008;32)

II.10.1. 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*.

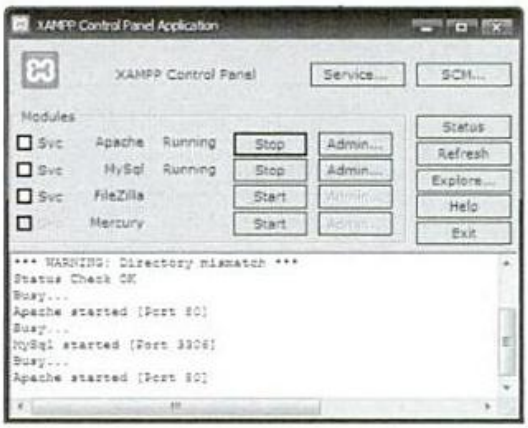
Xampplite akan langsung terekstrak ke tempat tujuan. Ada banyak file di dalam *folder xampplite*. *Folder* penting adalah *htdocs* di mana file-file halaman *web* harus diletakkan di situ. Yang kedua adalah file *xampp-control.exe* yang

berguna untuk mengatur perilaku *server*, seperti mengaktifkan dan menonaktifkan komponen-komponen tertentu. Untuk melihat tempat foler terinstall *xampplite* dapat dilihat pada Gambar II.14.



Gambar II.14. Tempat Penyimpanan xampplite
(Sumber : Ali Zaki;2008;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.15.



Gambar II.15. Mengaktifkan PHP dan *MySQL*
(Sumber : Ali Zaki;2008;35)

II.11. 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.11.1. *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.11.2. *Database Relasional*

Ketika Anda menggunakan *software* sistem manajemen database terkomputerisasi, umumnya menggunakan *database relasional*. Prinsip database relasional adalah informasi dibagi menjadi beberapa data yang terpisah secara logis. Data-data yang terpisah tersebut diletakkan dalam bentuk tabel. Tabel adalah objek dasar yang merupakan jantung dari *databse* relasional. Tabel adalah dasar penyimpanan informasi dan pengambilannya (*retrieval*). Ketika informasi sudah terseimpan di tabel-tabel yang terpisah, Anda nantinya dapat melihat (*view*), mengedit (*edit*), menambah (*add*), dan menghapus (*delete*) informasi dengan berbagai metode. Selain itu, Anda juga bisa mengambil informasi menggunakan *query* dan menampilkan informasi menggunakan *report*.

Keunggulan penyimpanan data menggunakan *database relasional* sangat banyak dibandingkan dengan penyimpanan ke satu tabel ukuran besar dua dimensi (yang disebut *file flat*) seperti di dokumen *Word* atau *spreadsheet Excel*.

Salah satu keunggulan utama *database* adalah berkurangnya *redudansi* data. Konsekuensinya tidak hanya ruang penyimpanan harddisk menjadi berkurang, tetapi kecepatan pemrosesan data juga berkurang. Selain itu, ada kelebihannya lainnya, yaitu :

1. Fleksibiltas

   Jika data berubah, Anda dapat mengupdate nilainya hanya di satu tempat. Dengan demikian, semua *query, form*, dan lainnya akan berubah secara konsekuen dengan nilai yang baru tersebut.

2. *Simple*

   Model penyimpanan tabel yang merupakan dasar dari sistem relasional memang *simple*, dan merupakan metode penyimpanan data yang tak *redundan*. Tiap tabel didesain relasional untuk objek tunggal yang mengandung data, yang konsekuen terhadap aspek tertentu dari database, seperti pegawai, produk atau *order*.

3. *Power*

   Menyimpan data dalam tabel-tabel yang terpisah lebih memudahkan adanya pengelompokkan, *searching*, dan pengambilan data menggunakan banyak cara yang tak terbatas.

4. Kemudahan manajemen

   Dengan tabel yang kecil dan tidak kompleks, informasi jadi mudah dilacak dan diatur.

Misalnya, Anda menggunakan database untuk menyimpan data *order* dari pelanggan. Jika menggunakan tabel tungga, ada banyak *record* yang dibuat untuk tiap *order* yang dilakukan pelanggan. Dengan demikian, tiap kali ada *order*, tiap kali pula sebuah informasi dibuat, walaupun pelanggannya sama. Karena itu, jika suatu saat ada informasi yang berubah misalnya informasi pelanggan, semua *record* yang mengandung informasi pelanggan harus diubah. Ini akan sangat merepotkan. Begitu pula informasi tentang pegawai pemroses *order* juga akan terus berulang (Wahana Komputer;2010:3).

Akan lebih efisien seandainya ada satu tabel untuk pelanggan dan satu tabel untuk pegawai. Kemudian, ada *field identitas* pelanggan yang ditambahkan ke tabel pelanggan dan tabel *order* sehingga ada hubungan atau koneksi antar keduanya. Hubungan ini disebut r*elationalship*. Begitu pula *identitas* pegawai bisa dihubungkan dengan tabel *order* menggunakan *relationship*. Secara lengkap, komponen utama *database* dijelaskan seperti berikut ini.

1. Tabel

Tabel adalah inti konsep database. Tujuannya adalah menyimpan informasi. Tabel satu dengan lainnya bisa dihubungkan. Satu *database* bisa mengandung banyak tabel dalam jumlah tak terbatas. Jumlah *record* dalam tabel juga umumnya tak dibatasi, batasnya hanya pada kapasitas disk yang digunakan. Tiap tabel memiliki beberapa kolom, kolom menentukan nilai-nilai apa yang bisa disimpan, misalnya, tabel yang meyimpan informasi produk menyimpan data seperti nama, harga dan berat produk.

Setiap baris dari tabel bisa mengandung nilai untuk kolom-kolom yang ditentukan di tabel. Selain itu, tiap kolom bisa ditentukan tipe data apa saja yang didukung. Tipe data menentukan jenis data yang bisa disimpan. Tipe data bisa dibatasi seperti jenis numerik atau tanggal, atau lain sebagainya.

2. *Query*

Anda dapat mengambil informasi tertentu yang disimpan di tabel atau *multitabel* menggunakan sebuah *query*. Untuk membuat *query*, Anda harus menentukan parameter-parameter dari informasi yang ingin dicari. Misalnya Anda ingin mengambil data pelanggan yang telah membeli barang tipe X selama 3

bulan yang lalu. Dengan demikian, Anda harus mengisikan *variabel* nama barang dan periode. Informasi yang ditampilkan juga nantinya bisa diurutkan, di *filter* dan diatur cara menampilkannya.

Dengan menjalankan *query*, Anda bisa menampilkan data berupa nilai-nilai yang sesuai dengan kriteria.

II.12. UML (*Unified Modelling Language*)

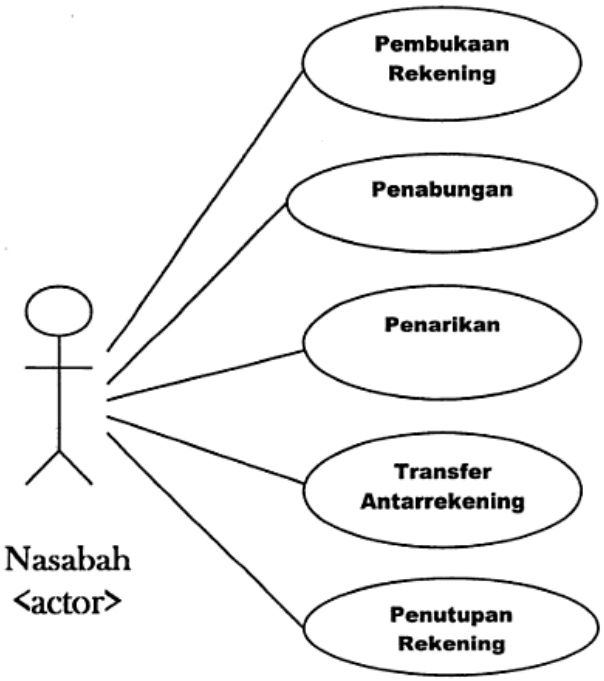
Pemodelan perangkat lunak bekerja dengan cara yang cukup serupa layaknya seorang arsitek atau insinyur teknik sipil yang akan membuat sebuah bangunan / gedung berskala besar. Saat seorang arsitek atau insinyur teknik sipil akan membuat sebuah bangunan / gedung berskala besar, ia biasanya membuat denah-denah 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.12.1. *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)..

Membuat use case diagram yang *komprehensif* merupakan hal yang sangat penting dilakukan pada tahap analisis. Dengan menggunakan *use case* diagram, kita akan mendapatkan banyak informasi yang sangat penting yang berkaitan dengan aturan-aturan bisnis yang coba kita tangkap. Dalam hal ini, setiap objek yang berinteraksi dengan sistem perangkat lunak misalnya, orang, suatu perangkat keras, sistem lain, dan sebagainya merupakan *actor* untuk sistem perangkat lunak, sementara *use case* merupakan *deskripsi* lengkap tentang bagaimana sistem perangkat lunak berperilaku untuk para *actornya*. Dengan demikian, use cara diagram merupakan deskripsi lengkap tentang interaksi yang terjadi antara para *actor* dengan sistem perangkat lunak yang sedang kita kembangkan. Untuk lebih jelas dapat dilihat pada Gambar II.16.



Gambar II.16. *Diagram Use Case*
(Sumber : Adi Nugroho; 2009 : 8)

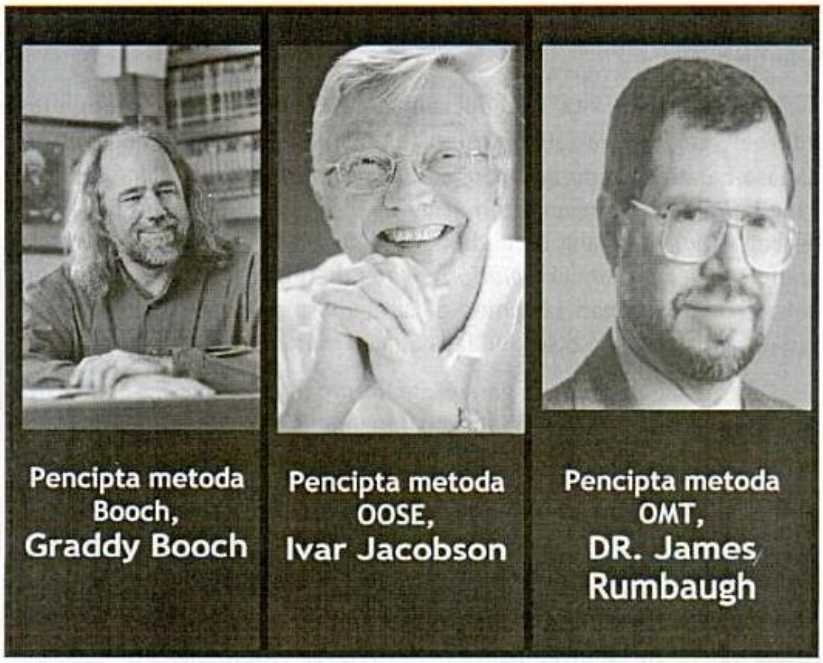
Actor pada dasarnya ditentukan berdasarkan perannya (*role*) pada program / aplikasi yang sedang kita kembangkan, bukan sebagai objek-objek secara mandiri. Sebagai contoh, jika mengambil kasus ATM (Anjungan Tunai Mandiri), seseorang (objek tunggal) mungkin bisa dikelompokkan sebagai *actor* Karyawan Bank serta Nasabah jika oarng tersebut merupakan karyawan bank yang bersangkutan sekaligus sebagai nasabah karena memiliki tabungan di bank tersebut. Sementara itu, Adi, Ana Geuis, dan beberapa orang lainnya dapat dikelompokkan menjadi *actor* nasabah jika mereka semua masing-masing memiliki tabungan di bank tersebut. Dalam hal ini, kita akan coba mengambil contoh *actor* nasabah untuk menentukan *use casenya*.

Untuk *actor* Nasabah ini, kita akan mencari tahu mengenai apa yang sebenarnya nasabah lakukan saat berinteraksi dengan aplikasi perbankan. Dalam hal ini, kita

mungkin mendapatkan perilaku-perilaku (*behaviour*) *actor* Nasabah seperti berikut.

- Nasabah membuka rekeningnya.
- Nasabah memeriksa *saldo* rekeningnya.
- Nasabah menyimpan uang pada rekeningnya.
- Nasabah menarik uang dari rekeningnya.
- Nasabah melakukan *transfer* uang antar rekening.
- Nasabah menutup rekeningnya.

Untuk menentukan *use case* dari kasus bank tersebut, kita harus melihat mekanisme yang dilakukan oleh masing-masing *actor* pada masing-masing perilaku yang telah disebutkan, apakah terlihat adanya perbedaan dalam mekanismenya. Sebagai contoh, Nasabah melakukan penyimpanan uangnya di bank dengan cara yang berbeda dengan penarikannya (Nasabah menaruh uangnya dengan cara dating ke bank, sementara Nasabah menarik uangnya melalui ATM). Dengan demikian, keduanya bisa kita kelompokkan menjadi 2 *use* cara yang berbeda. Lalu, bagaimana dengan perilaku lainnya? Nasabah memeriksa saldonya mungkin terlalu sederhana jika kita kelompokkan menjadi suatu *use case* tunggal sehingga kita bisa saja memasukkan perilaku ini ke *use case*. Penarikan sebab sering kali sebelum Nasabah melakukan penarikan uangnya, sistem akan memeriksa apakah *saldo* untuk Nasabah yang kelompokkan seperti yang terlihat dalam Gambar II.17.



Gambar II.17. Para Pengembangan Metode UML
(Sumber : Adi Nugroho; 2009 : 4)

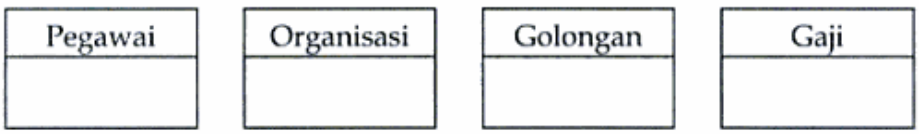
II.13. *Entity Relationship Diagram*

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.

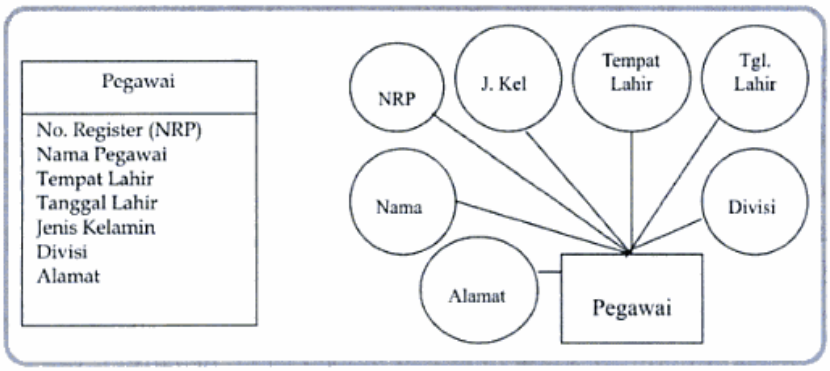
Untuk mempermudah dalam perancangan *database*, maka digunakan *Entity Relationship Diagram* (ERD). ERD diutamakan untuk permodelan dari desain konseptual. *Entity Relationship Diagaram* menggambarkan struktur dan keterkaitan *tabel-tabel* data yang menyusun *database* secara detail. ERD merupakan representasi 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.18.



Gambar II.18. Contoh Pembuatan Entitas
(Sumber : Marimin, Hendri Tanjung, Haryo Prabowo;2010:112)

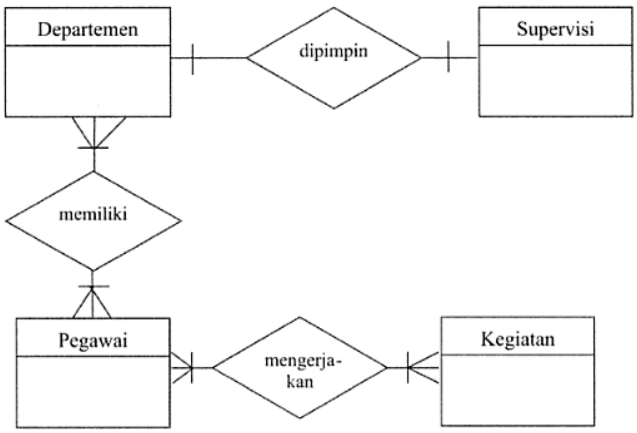
Entitas selanjutnya dijelaskan dengan *atribut-atribut* yang ada di dalamnya atau sering kali disebut elemen data. *Atribut* atau elemen data merupakan unit terkecil dari data yang dapat menjalaskan apa yang dimiliki oleh suatu entitas (*karakteristik* dari *entitas*). Misalnya *entitas* pegawai memiliki *atribut* yang terlihat pada Gambar II.19. Sedangkan *relasi* menjelaskan keterkaitan di antara dua *entitas* yang berbeda misalnya pegawai bekerja pada suatu departemen.



Gambar II.19. Contoh Atribut atau Elemen Data Suatu *Entitas*
(Sumber : Marimin, Hendri Tanjung, Haryo Prabowo;2010:112)

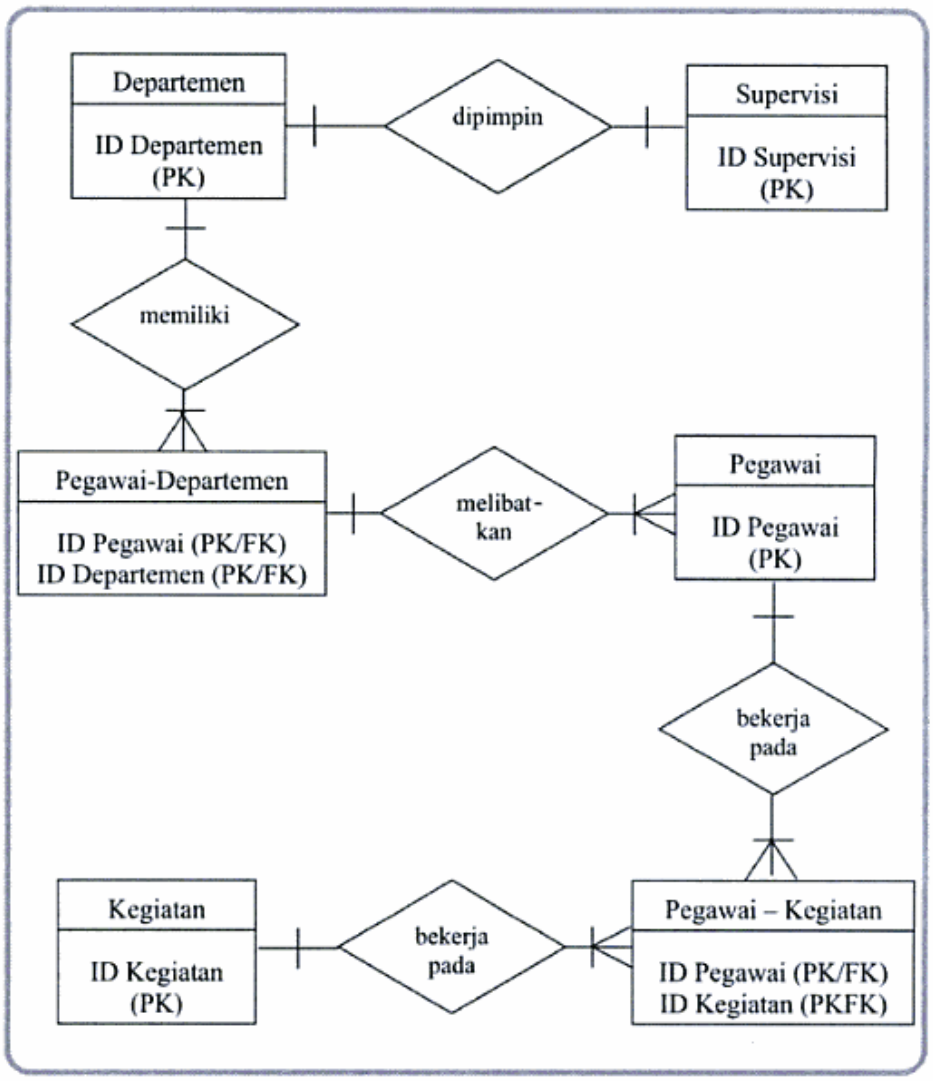
Sifat hubungan atau *relasi* antarentitas dapat dibedakan menjadi tiga jenis, yaitu hubungan satu ke satu (*one to one relationship*), satu kebanyak (*one to many relationship*) dan banyak ke banyak (*many to many relationship*).

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.20.



Gambar II.20. Contoh *One to One* dan *Many to Many Relationship*
(Sumber : Marimin, Hendri Tanjung, Haryo Prabowo;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.21.



Gambar 7.6 Contoh Relasi *One-to-One, One-to-Many,* dan *Many-to-Many*

Gambar II.21. Contoh *Relasi One to One, One to Many* dan *Many to Many*
(Sumber : Marimin, Hendri Tanjung, Haryo Prabowo;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.21 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.*