BAB II

LANDASAN TEORI

2.1 Kecerdasan Buatan (*Artificial Intelligence*)

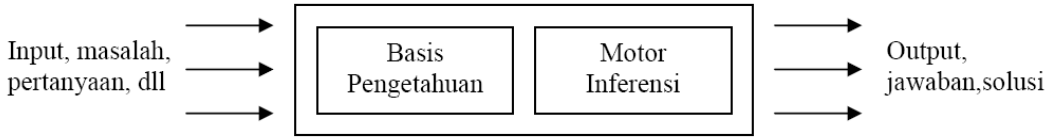
Kecerdasan buatan (*Artificial Intelligence*) adalah bagian dari ilmu komputer yang mempelajari bagaimana membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia bahkan bisa lebih baik daripada yang dilakukan manusia. [10]

Menurut John McCarthy, 1956, AI :

"Untuk mengetahui dan memodelkan proses – proses berpikir manusia dan mendesain mesin agar dapat menirukan perilaku manusia."

Cerdas adalah memiliki pengetahuan dan pengalaman, penalaran yaitu bagaimana membuat keputusan dan mengambil tindakan, serta moral yang baik. Agar mesin bisa cerdas atau bertindak seperti dan sebaik manusia, maka harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menalar. Penerapan kecerdasan buatan digambarkan pada Gambar II.1. Dua bagian utama yg dibutuhkan untuk aplikasi kecerdasan buatan adalah [10]:

a. Basis pengetahuan (*knowledge base*): berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
b. Motor inferensi (*inference engine*) : kemampuan menarik kesimpulan berdasarkan pengetahuan.



Gambar II.1 Penerapan Kecerdasan Buatan[10]

2.1.1 Perbedaan Kecerdasan Buatan dan Kecerdasan Alami

Berikut ini merupakan Perbedaan antara kecerdasan buatan dan kecerdasan alami :

1. Kelebihan kecerdasan buatan
    a. Lebih bersifat permanen. Kecerdasan alami bisa berubah karena sifat manusia pelupa. Kecerdasan buatan tidak berubah selama sistem komputer dan program tidak mengubahnya.
    b. Lebih mudah diduplikasi dan disebarkan. Memindahkan pengetahuan manusia dari 1 orang ke orang lain membutuhkan proses yang sangat lama dan keahlian tidak akan pernah dapat diduplikasi dengan lengkap.Jadi jika pengetahuan terletak pada suatu sistem komputer, pengetahuan tersebut dapat disalin dari komputer tersebut dan dapat dipindahkan dengan mudah ke komputer yang lain.
    c. Lebih murah. Menyediakan layanan komputer akan lebih mudah dan murah dibandingkan mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
    d. Bersifat konsisten dan teliti karena kecerdasan buatan adalah bagian dari teknologi komputer sedangkan kecerdasan alami senantiasa berubah-ubah.

e. Dapat didokumentasikan. Keputusan yang dibuat komputer dapat didokumentasikan dengan mudah dengan cara melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.

f. Dapat mengerjakan beberapa tugas dengan lebih cepat dan lebih baik dibanding manusia.

2. Kelebihan kecerdasan alami

a. Kreatif : manusia memiliki kemampuan untuk menambah pengetahuan, sedangkan pada kecerdasan buatan untuk menambah pengetahuan harus dilakukan melalui sistem yang dibangun.

b. Memungkinkan orang untuk menggunakan pengalaman atau pembelajaran secara langsung. Sedangkan pada kecerdasan buatan harus mendapat masukan berupa data-data simbolik.

c. Pemikiran manusia dapat digunakan secara luas, sedangkan kecerdasan buatan sangat terbatas.

2.1.2 Kajian Kecerdasan Buatan

Terdapat banyak irisan dalam ruang lingkup kecerdasan buatan yang membuat klasifikasi kecerdasan buatan menjadi rumit, sehingga untuk memudahkan klasifikasi kecerdasan buatan, pengklasifikasian kecerdasan buatan didasarkan pada keluaran yang diberikan. Namun, secara garis besar, beberapa kajian yang termasuk kecerdasan buatan adalah :

1. Sistem Pakar (*Expert System*)

Sistem pakar (*expert system*) adalah sistem yang berusaha mengapdosi pengetahuan manusia ke komputer, agar komputer dapat

menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli.

Jadi sistem pakar merupakan sistem yag memindahkan keahlian dari seorang pakar atau sumber kepakaran yang lain ke komputer, pengetahuan yang ada disimpan dalam komputer, dan pengguna dapat berkonsultasi pada komputer itu untuk suatu nasehat, lalu komputer dapat mengambil inferensi (menyimpulkan, mendeduksi, dll.) seperti layaknya seorang pakar, kemudian menjelaskannya ke pengguna tersebut, bila perlu dengan alasan-alasannya.

Sistem Pakar terkadang lebih baik dalam melakukan pekerjaan dari pada seorang pakar manusia. Dengan sistem pakar, orang awam pun dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Bagi para ahli, sistem pakar juga akan membantu aktivitasnya sebagai asisten yang sangat berpengalaman.

Manfaat sistem pakar antara lain:

a. Memungkinkan orang awam untuk bisa mengerjakan pekerjaan para ahli.
b. Bisa melakukan proses berulang-ulang secara otomatis.
c. Menyimpan pengetahuan dan keahlian para pakar.
d. Mampu mengambil dan melestarikan keahlian para pakar (terutama yang termasuk keahlian langka).

e. Mampu beroperasi dalam lingkungan yang berbahaya.
f. Memiliki kemampuan untuk bekerja dengan informasi yang tidak lengkap dan mengandung ketidakpastian. Pengguna bisa merespon dengan jawaban ’tidak tahu’ atau ’tidak yakin’ pada satu atau lebih pertanyaan selama konsultasi dan sistem pakar tetap akan memberikan jawaban.
g. Tidak memerlukan biaya saat tidak digunakan, sedangkan pada pakar manusia memerlukan biaya sehari-hari.
h. Dapat digandakan (diperbanyak) sesuai kebutuhan dengan waktu yang minimal dan sedikit biaya.
i. Dapat memecahkan masalah lebih cepat daripada kemampuan manusia dengan catatan menggunakan data yang sama.
j. Menghemat waktu dalam pengambilan keputusan.
k. Meningkatkan kualitas dan produktivitas karena dapat memberi nasehat yang konsisten dan mengurangi kesalahan.
l. Meningkatkan kapabilitas sistem terkomputerisasi yang lain. Integrasi Sistem Pakar dengan sistem komputer lain membuat lebih efektif, dan bisa mencakup lebih banyak aplikasi .
m. Mampu menyediakan pelatihan. Pengguna pemula yang bekerja dengan sistem pakar akan menjadi lebih berpengalaman. Fasilitas penjelas dapat berfungsi sebagai guru.

Kelemahan sistem pakar antara lain:

a. Biaya yang diperlukan untuk membuat, memelihara, dan mengembangkannya sangat mahal.
b. Sulit dikembangkan, hal ini erat kaitannya dengan ketersediaan pakar di bidangnya dan kepakaran sangat sulit diekstrak dari manusia karena sangat sulit bagi seorang pakar untuk menjelaskan langkah mereka dalam menangani masalah.
c. Sistem pakar tidak 100% benar karena seseorang yang terlibat dalam pembuatan sistem pakar tidak selalu benar. Oleh karena itu perlu diuji ulang secara teliti sebelum digunakan.
d. Pendekatan oleh setiap pakar untuk suatu situasi atau masalah bisa berbeda-beda, meskipun sama-sama benar.
e. Pemindahan pengetahuan dapat bersifat subjektif dan bias
f. Kurangnya rasa percaya pengguna dapat menghalangi pemakaian sistem pakar.

2. Permainan (*Game Playing*)

Games adalah fasilitas yang sangat menarik dalam komputer. Ide *games* pertama kali dimunculkan oleh Claude Shannon (1950) yang menulis karya tulis tentang mekanisme pembuatan program permainan catur. Beberapa alasan mengapa *games* merupakan domain yang baik untuk dieksplore, yaitu:

a. Sangat mudah untuk menentukan ukuran kesuksesan dan kegagalannya (menang atau kalah).

b. Tidak membutuhkan terlalu banyak pengetahuan. Permainan dapat diselesaikan dengan melakukan pencarian dari arah mulai sampai posisi menang. Namun alasan ini tidak cocok untuk *games* yang besar (kompleks).
c. Ruang keadaannya mudah direpresentasikan.
d. Operator-operator yang digunakan tidak terlalu banyak.
e. Sebagian besar *game* dapat dimodelkan dengan mudah.
f. Sangat mungkin untuk dibandingkan dengan kemampuan manusia.

3. Logika Fuzzy

Logika fuzzy adalah suatu cara untuk memetakan suatu ruang masukan ke dalam suatu ruang keluaran. Antara masukan dan keluaran terdapat suatu kotak hitam yang harus memetakan masukan ke keluaran yang sesuai. Skema logika fuzzy dapat dilihat pada Gambar II.2 berikut ini:



Gambar II.2 Skema Logika Fuzzy[10]

Beberapa alasan menggunakan logika fuzzy adalah sebagai berikut:

a. Konsep logika fuzzy mudah dimengerti.
b. Logika fuzzy sangat fleksibel.
c. Memiliki toleransi terhadap data-data yang tidak tepat.
d. Dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.
e. Logika fuzzy didasarkan pada bahasa alami.

4. Jaringan Syaraf Tiruan (*Artificial Neural Network*)

Jaringan syaraf merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran.

5. Algoritma Genetika (*Genetic Algorithm*)

Algoritma genetika adalah algoritma pencarian heuristik yg didasarkan atas mekanisme evolusi biologis. Keberagaman pada evolusi biologis adalah variasi dari kromosom antar individu organisme. Variasi kromosom ini akan mempengaruhi laju produksi dan tingkat kemampuan organisme untuk tetap hidup. Individu yang lebih kuat akan memiliki tingkat kelangsungan hidup dan tingkat reproduksi yang lebih tinggi dibandingkan dengan individu yang kurang fit. Komponen utama dalam algoritma genetika ada 6, yaitu sebagai berikut:

a. Teknik Penyandian
b. Prosedur Inisialisasi
c. Fungsi Evaluasi
d. Seleksi
e. Operator Genetika
f. Penentuan Parameter

6. Robotika

Kecerdasan buatan dalam robotika diterapkan sebagai algoritma yang dimasukkan kedalam kontroler robot. Robot dengan kecerdasan buatan membuat robot tersebut berkemampuan secara sendiri untuk merespon atau bereaksi didalam kondisi yang tidak ditentukan sebelumnya.

2.1.3 Penerapan Kecerdasan Buatan

Kecerdasan buatan banyak diterapkan pada berbagai bidang, misalnya pada bidang ilmu komputer (informatika), teknik elektro, kesehatan, manajemen dan lain sebagainya. Beberapa contoh penerapan kecerdasan buatan adalah sebagai berikut:

1. Pengenalan Pola

Pengenalan pola dilakukan untuk mengenali suatu objek tertentu, misalnya untuk keperluan absensi menggunakan pengenalan sidik jari (*finger recognition*), pengenalan suara (*voice recognition*) atau pengenalan wajah (*face recognition*). Selain itu juga dapat digunakan sebagai mesin pencarian dengan gambar atau suara sebagai kata

kunci(*keyword*)nya dengan menggunakan pengenalan gambar (*image recognition*) dan pengenalan ucapan (*speech recognition*).[20]

2. Sistem Pakar

Sistem pakar digunakan sebagai sarana untuk menyimpan pengetahuan para pakar, sehingga komputer dapat menyelesaikan permasalahan dengan meniru keahlian yang dimiliki oleh pakar. Misalnya analisis penyakit, kerusakan komputer, penasihat keuangan dan lain sebagainya.

3. Game Playing

Game merupakan fasilitas menarik yang terdapat dalam komputer. *Game* dalam komputer seperti catur, kartu, dan lainnya menggunakan kecerdasan buatan agar dapat bermain selayaknya manusia, memiliki strategi langkah selanjutnya, pengetahuan peraturan permainan dan dapat mengetahui pemenangnya.

4. Robotika dan Sistem Sensor

Salah satu contoh sistem sensor adalah pada mesin cuci yaitu menggunakan sensor optik, mengeluarkan cahaya ke air dan mengukur bagaimana cahaya tersebut sampai ke ujung lainnya. Makin kotor, maka sinar yang sampai makin redup. Sistem juga mampu menentukan jenis kotoran tersebut daki/minyak. Sistem juga bisa menentukan putaran yang tepat secara otomatis berdasarkan jenis dan banyaknya kotoran serta jumlah yang akan dicuci.

2.2 Jaringan Syaraf Tiruan (JST)

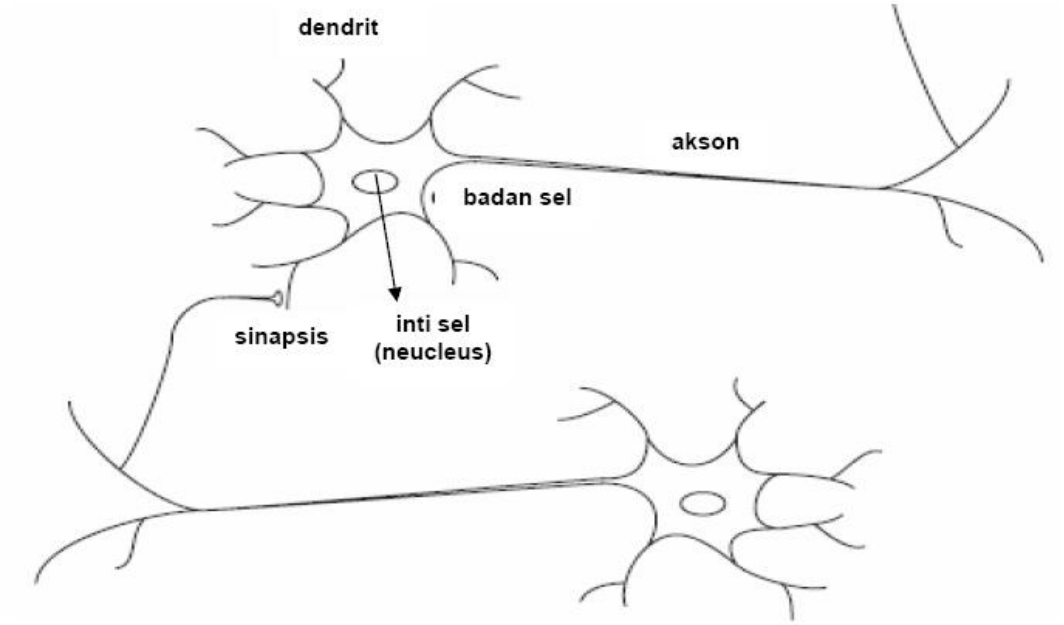
Jaringan syaraf adalah merupakan salah satu representasi buatan dari otak manusia yang selalu mencoba untuk mensimulasikan proses pembelajaran pada otak manusia tersebut. Istilah buatan disini digunakan karena jaringan syaraf ini diimplementasikan dengan menggunakan program komputer yang mampu menyelesaikan sejumlah proses perhitungan selama proses pembelajaran[12].

2.2.1 Jaringan Syaraf Biologi dan Tiruan

Jaringan syaraf biologi merupakan jaringan syaraf yang terdapat pada manusia, sedangkan jaringan syaraf tiruan merupakan jaringan syaraf yang dibuat dan dijalankan pada komputer.

A. Jaringan Syaraf Biologi (JSB)

Otak manusia memiliki struktur yang sangat kompleks dan memiliki kemampuan yang luar biasa. Otak terdiri dari neuron-neuron dan penghubung yang disebut sinapsis. Neuron bekerja berdasarkan impuls/sinyal yang diberikan pada neuron. Neuron meneruskannya pada neuron lain. Diperkirakan manusia memiliki 10^{12} neuron dan 6.10^{8} sinapsis. Dengan jumlah yang begitu banyak, otak mampu mengenali pola, melakukan perhitungan, dan mengontrol organ-organ dengan kecepatan yang lebih tinggi dibandingkan komputer digital. Sebagai perbandingan, pengenalan wajah seseorang yang sedikit berubah (misal memakai topi, memiliki jenggot tambahan, dan lain-lain) akan lebih cepat dilakukan manusia dibandingkan komputer. Gambar neuron seperti dilihat pada Gambar II.3



Gambar II.3 Neuron[7]

Komponen utama neuron dapat dikelompokkan menjadi 3 bagian, yaitu:

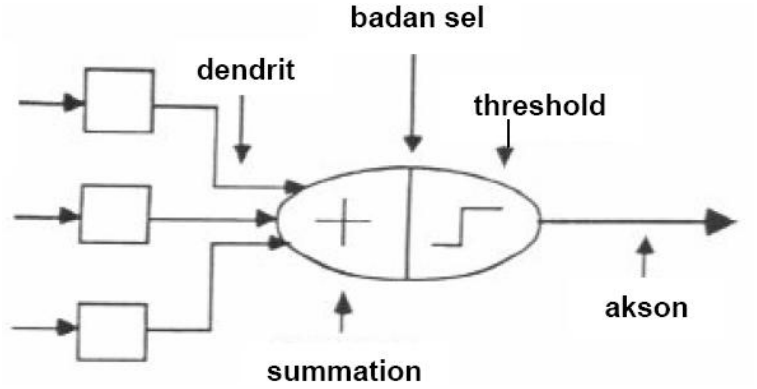
1. Dendrit = bertugas menerima informasi = jalur masukan bagi soma
2. Badan sel (soma) = tempat pengolahan informasi
3. Akson = bertugas mengirimkan impuls-impuls sinyal ke sel syaraf lain = jalur keluaran bagi soma

Dari Gambar II.3 didapat kesimpulan sebagai berikut:

1. Sebuah neuron menerima impuls-impuls sinyal (informasi) dari neuron lain melalui dendrit dan mengirimkan sinyal yang dibangkitkan (hasil penjumlahan) oleh badan sel melalui akson.
2. Akson dari sel syaraf ini bercabang-cabang dan berhubungan dengan dendrit dari sel syaraf lain dengan cara mengirimkan impuls melalui sinapsis.

3. Sinapsis adalah unit fungsional antara 2 buah sel syaraf, misal A dan B, dimana yang satu adalah serabut akson dari neuron A dan satunya lagi adalah dendrit dari neuron B.
4. Kekuatan sinapsis bisa menurun / meningkat tergantung seberapa besar tingkat propagasi (penyiaran) sinyal yang diterimanya.
5. Impuls-impuls sinyal (informasi) akan diterima oleh neuron lain jika memenuhi batasan tertentu, yang sering disebut dengan nilai ambang (*threshold*).

Model struktur neuron jaringan syaraf biologi ditunjukkan pada Gambar II.4.



Gambar II.4 Model Struktur Neuron JSB[7]

B. Jaringan Syaraf Tiruan (JST)

Jaringan Syaraf Tiruan (JST) merupakan suatu sistem pemrosesan informasi yang mempunyai karakteristik menyerupai jaringan syaraf biologi (JSB). JST tercipta sebagai suatu generalisasi model matematis dari pemahaman manusia (*human cognition*) yang didasarkan atas asumsi sebagai berikut :

1. Pemrosesan informasi terjadi pada elemen sederhana yang disebut neuron

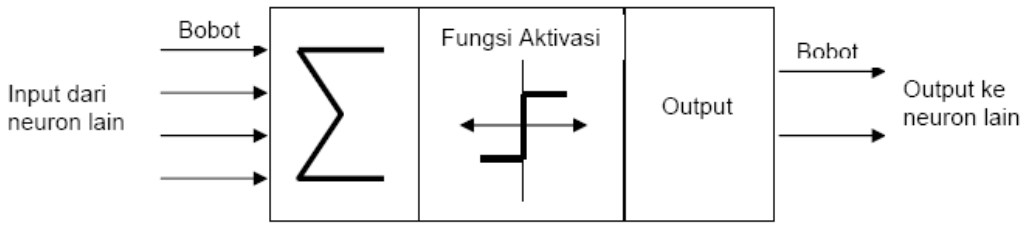
2. Sinyal mengalir diantara sel saraf/neuron melalui suatu sambungan penghubung
3. Setiap sambungan penghubung memiliki bobot yang bersesuaian. Bobot ini akan digunakan untuk menggandakan / mengalikan sinyal yang dikirim melaluinya.
4. Setiap sel syaraf akan menerapkan fungsi aktivasi terhadap sinyal hasil penjumlahan berbobot yang masuk kepadanya untuk menentukan sinyal keluarannya.

Analogi jaringan syaraf tiruan dengan jaringan syaraf buatan dijelaskan pada Tabel II.1 berikut ini:

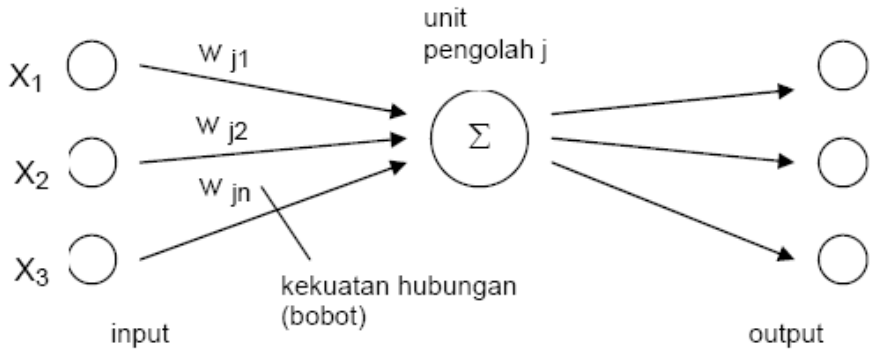
Tabel II.1 Analogi JST dengan JSB

JST	JSB
Node/masukan	Badan sel (soma)
Masukan	Dendrit
Keluaran	Akson
Bobot	Sinapsis

Model struktur neuron jaringan syaraf tiruan dijelaskan pada Gambar II.5 dan Gambar II.6.



Gambar II.5 Model Struktur JST[7]



Gambar II.6 Model Struktur JST[7]

Jaringan syaraf tiruan dapat belajar dari pengalaman, melakukan generalisasi atas contoh-contoh yang diperolehnya dan mengabstraksi karakteristik esensial masukan bahkan untuk data yang tidak relevan. Algoritma untuk JST beroperasi secara langsung dengan angka sehingga data yang tidak numerik harus diubah menjadi data numerik. JST tidak diprogram untuk menghasilkan keluaran tertentu. Semua keluaran atau kesimpulan yang ditarik oleh jaringan didasarkan pada pengalamannya selama mengikuti proses pembelajaran. Pada proses pembelajaran, ke dalam JST dimasukkan pola-pola masukan (dan keluaran) lalu jaringan akan diajari untuk memberikan jawaban yang bisa diterima. Pada dasarnya karakteristik JST ditentukan oleh :

1. Pola hubungan antar neuron (disebut arsitektur jaringan)
2. Metode penentuan bobot-bobot sambungan (disebut dengan pelatihan atau proses belajar jaringan)
3. Fungsi aktivasi

2.2.2 Arsitektur JST

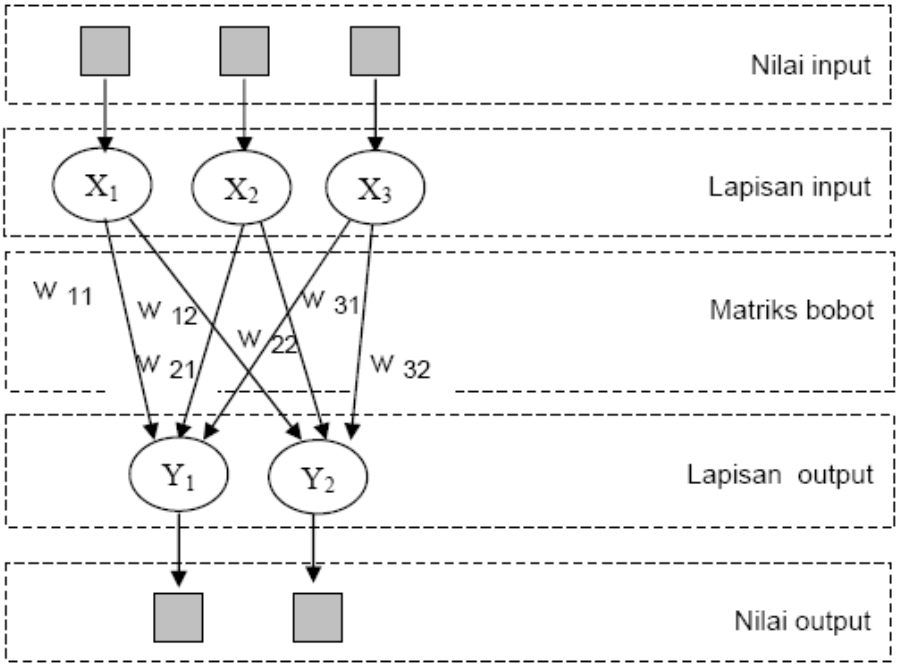
Pada JST, neuron-neuron akan dikumpulkan dalam lapisan-lapisan (*layer*) yang disebut dengan lapisan neuron (neuron *layers*). Neuron-neuron pada satu lapisan akan dihubungkan dengan lapisan-lapisan sebelum dan sesudahnya. Informasi yang diberikan pada jaringan syaraf akan dirambatkan lapisan ke lapisan, mulai dari lapisan masukan sampai ke lapisan keluaran melalui lapisan tersembunyi (*hidden layer*). Gambar berikut ini jaringan syaraf dengan 3 lapisan dan bukanlah struktur umum jaringan syaraf karena beberapa jaringan syaraf ada yang tidak memiliki lapisan tersembunyi.

Faktor terpenting dalam menentukan kelakuan suatu neuron adalah fungsi aktivasi dan pola bobotnya. Umumnya neuron-neuron yang terletak pada lapisan yang sama akan memiliki keadaan yang sama sehingga pada setiap lapisan yang sama neuron-neuron memiliki fungsi aktivasi yang sama. Bila neuron-neuron pada suatu lapisan (misal lapisan tersembunyi) akan dihubungkan dengan neuron-neuron pada lapisan lain (misal lapisan keluaran) maka setiap neuron pada lapisan tersebut (lapisan tersembunyi) juga harus dihubungkan dengan setiap neuron pada lapisan lainnya (lapisan keluaran). Terdapat 3 macam arsitektur JST, yaitu:

1. Jaringan dengan lapisan tunggal (*single layer net*)

   Jaringan ini hanya memiliki 1 lapisan dengan bobot-bobot terhubung. Jaringan ini hanya menerima masukan kemudian secara langsung akan mengolahnya menjadi keluaran tanpa harus melalui lapisan tersembunyi. Pada gambar berikut neuron-neuron pada kedua lapisan saling berhubungan.

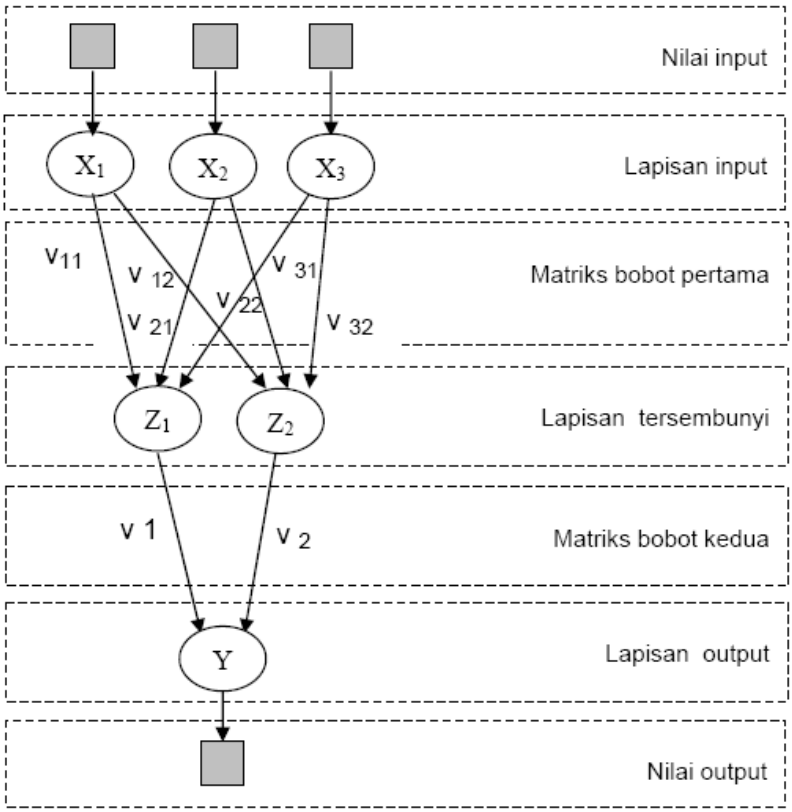
Seberapa besar hubungan antara 2 neuron ditentukan oleh bobot yang bersesuaian. Semua unit masukan akan dihubungkan dengan setiap unit keluaran seperti terlihat pada Gambar II.7.



Gambar II.7 Jaringan Dengan Lapisan Tunggal [7]

2. Jaringan dengan banyak lapisan (*multilayer net*)

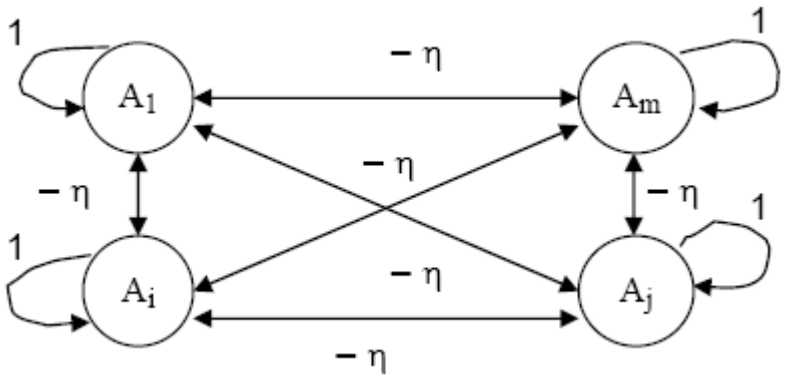
   Jaringan ini memiliki 1 atau lebih lapisan yang terletak diantara lapisan masukan dan lapisan keluaran. Umumnya ada lapisan bobot-bobot yang terletak antara 2 lapisan yang bersebelahan seperti terlihat pada Gambar II.8. Jaringan dengan banyak lapisan ini dapat menyelesaikan permasalahan yang lebih sulit daripada lapisan tunggal, tentu saja dengan pembelajaran yang lebih rumit. Pada banyak kasus, pembelajaran pada jaringan dengan banyak lapisan ini lebih sukses dalam menyelesaikan masalah.



Gambar II.8 Jaringan Dengan Banyak Lapisan[7]

3. Jaringan dengan lapisan kompetitif (*competitive layer net*)

   Pada jaringan ini sekumpulan neuron bersaing untuk mendapatkan hak menjadi aktif. Umumnya hubungan antar neuron pada lapisan kompetitif ini tidak diperlihatkan pada diagram arsitektur. Gambar II.9 menunjukkan salah satu contoh arsitektur jaringan dengan lapisan kompetitif yang memiliki bobot -η.



Gambar II.9 Jaringan dengan lapisan kompetitif[7]

2.2.3 Proses Pembelajaran Jaringan

Cara belajar JST adalah sebagai berikut:

a. Ke dalam JST dimasukkan informasi yang sebelumnya telah diketahui hasil keluarannya.
b. Proses memasukkan informasi ini dilakukan lewat node-node atau unit-unit masukan. Bobot-bobot antarkoneksi dalam suatu arsitektur diberi nilai awal dan kemudian JST dijalankan. Bobot-bobot ini bagi jaringan digunakan untuk belajar dan mengingat suatu informasi. Pengaturan bobot dilakukan secara terus-menerus dan dengan menggunakan kriteria tertentu sampai diperoleh keluaran yang diharapkan.

Hal yang ingin dicapai dengan melatih/mengajari JST adalah untuk mencapai keseimbangan antara kemampuan memorisasi dan generalisasi. Kemampuan memorisasi adalah kemampuan JST untuk memanggil kembali secara sempurna sebuah pola yang telah dipelajari. Kemampuan generalisasi adalah kemampuan JST untuk menghasilkan respon yang bisa diterima terhadap pola-pola masukan yang serupa (namun tidak identik) dengan pola-pola yang sebelumnya telah dipelajari.

Hal ini sangat bermanfaat bila pada suatu saat ke dalam JST dimasukkan informasi baru yang belum pernah dipelajari, maka JST masih akan tetap dapat memberikan tanggapan yang baik, memberikan keluaran yang paling mendekati. Paradigma/metode pembelajaran/pelatihan JST adalah sebagai berikut:

a. Pembelajaran terawasi (*supervised learning*)

   Pada pembelajaran ini kumpulan data masukan yang digunakan, data keluarannya telah diketahui. Perbedaan antara keluaran-keluaran aktual dengan data keluaran yang diinginkan digunakan untuk mengoreksi bobot JST agar JST dapat menghasilkan jawaban sedekat (semirip) mungkin dengan jawaban yang benar yang telah diketahui oleh JST.

b. Pembelajaran tak terawasi (*unsupervised learning*)

   Pada pembelajaran ini, JST mengorganisasi dirinya sendiri untuk membentuk vektor-vektor masukan yang serupa, tanpa menggunakan data atau contoh-contoh pelatihan. Struktur menggunakan dasar data atau korelasi antara pola-pola data yang dieksplorasi. Paradigma pembelajaran ini mengorganisasi pola-pola ke dalam kategori-kategori berdasarkan korelasi yang ada.

c. Gabungan pembelajaran terawasi dan tak terawasi (*hybrid*)

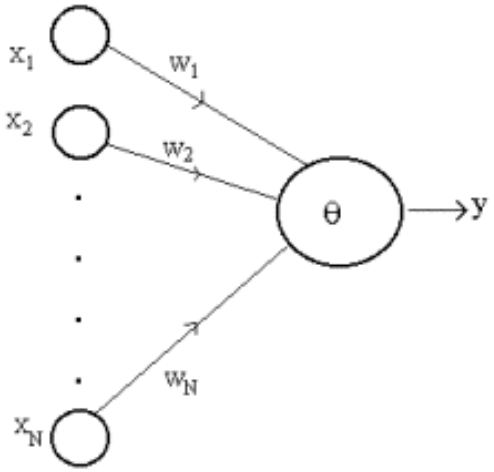
   Merupakan kombinasi dari kedua pembelajaran tersebut. Sebagian dari bobot-bobotnya ditentukan melalui pembelajaran terawasi dan sebagian lainnya melalui pembelajaran tak terawasi.

2.2.3.1 Pembelajaran Terawasi (*Supervised Learning*)

Metode pembelajaran pada jaringan syaraf disebut terawasi jika keluaran yang diharapkan telah diketahui sebelumnya, misalkan seperti pada proses AND. Pada proses pembelajaran, satu pola masukan akan diberikan ke satu neuron pada lapisan masukan . Pola ini akan dirambatkan di sepanjang jaringan syaraf hingga sampai ke neuron pada lapisan keluaran. Lapisan keluaran ini akan membangkitkan pola keluaran yang nantinya akan dicocokkan dengan pola keluaran targetnya. Apabila terjadi perbedaan antara pola keluaran hasil pembelajaran dengan pola target, maka akan muncul error. Apabila nilai error masih cukup besar, mengindikasikan bahwa masih perlu dilakukan lebih banyak pembelajaran lagi. Metode-metode yang termasuk dalam pembelajaran terawasi antara lain[6]:

B. Hebb Rule

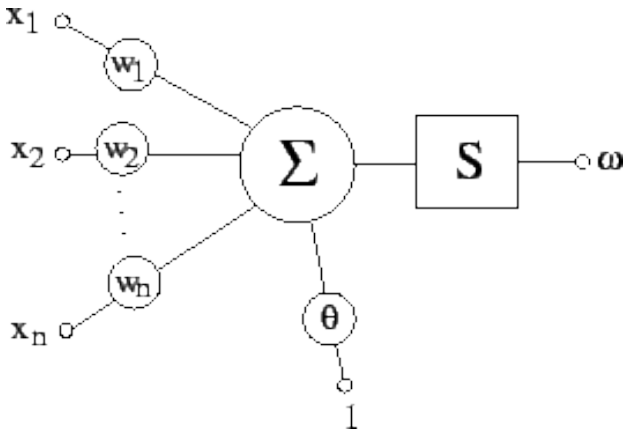
Hebb rule adalah metode pembelajaran yang paling sederhana. Pada metode ini pembelajaran dilakukan dengan cara memperbaiki nilai bobot sedemikian rupa sehingga jika ada 2 neuron yang terhubung, dan keduanya pada kondisi ‘hidup’ (*on*) pada saat yang sama, maka bobot antara keduanya dinaikkan. Gambar II.10 merupakan gambar jaringan hebb rule.



Gambar II.10 Hebb Rule[5]

C. Perceptron

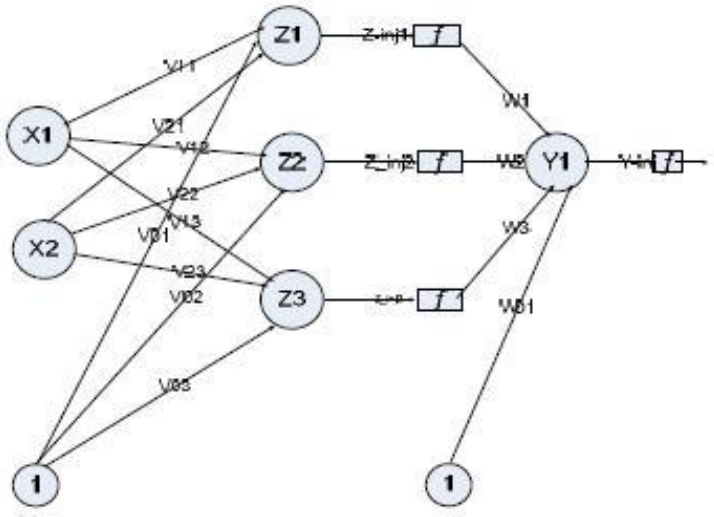
Jaringan jenis ini hanya terdiri dari *layer* masukan dan *layer* keluaran saja. Data masukan yang masuk melalui vector masukan akan langsung diproses dan kemudian ditentukan target keluaran dari hasil pengolahan. Jaringan perceptron dapat dilihat pada Gambar II.11.



Gambar II.11 Perceptron [11]

D. Backpropagation

Salah satu arsitektur jaringan saraf tiruan yang sangat populer adalah *multilayer feedforward networks*. Secara umum, jaringan seperti ini terdiri dari sejumlah unit neuron sebagai lapisan masukan, satu atau lebih lapisan simpul-simpul neuron komputasi lapisan tersembunyi, dan sebuah lapisan simpul-simpul neuron komputasi keluaran. Sinyal masukan dipropagasikan ke arah depan (arah lapisan keluaran), lapisan demi lapisan. Jenis jaringan ini adalah hasil generalisasi dari arsitektur *perceptron* satu lapisan, jadi biasa disebut sebagai *multilayer perceptron* (MLPs). *Error back propagation* adalah algoritma MLPs yang menggunakan prinsip pembelajaran terawasi. Propagasi balik (ke arah lapisan masukan) terjadi setelah jaringan menghasilkan keluaran yang mengandung *error*. Pada fase ini seluruh bobot *synaptic* (yang tidak memiliki aktivasi nol) dalam jaringan akan disesuaikan untuk mengkoreksi/memperkecil *error* yang terjadi (*error correction rule*). Untuk pelatihan jaringan, pasangan fase propagasi ke depan dan balik dilakukan secara berulang untuk satu set data latihan, kemudian diulangi untuk sejumlah *epoch* (satu sesi lewatan untuk seluruh data latihan dalam sebuah proses pelatihan jaringan) sampai *error* yang terjadi mencapai batas kecil toleransi tertentu atau nol. Jaringan *backpropagation* dapat dilihat pada Gambar II.12.[18]



Gambar II.12 *Backpropagation*[2]

Dari Gambar II.12 dapat dilihat bahwa yang bertindak sebagai dendrit adalah X1 dan X2, yaitu data masukan pada jaringan tersebut. Terdapat 2 sinapsis atau bobot yaitu V dan W, sedangkan Z dan Y merupakan bagian dari soma atau badal sel dari jaringan tersebut. Dan yang bertindak sebagai akson atau data keluaran adalah Y. Berikut ini algoritma pembelajaran metode *backpropagation*.

Langkah 0 : Inisialisasi bobot. (sebaiknya diatur pada nilai acak yang kecil),

Langkah 1 : Jika kondisi berhenti tidak tercapai, lakukan langkah 2-9, (kondisi berhenti jika perulangan mencapai maksimal perulangan atau *mean square error* (MSE) sudah lebih kecil dari rasio pembelajaran).

Langkah 2 : Untuk setiap pasangan pelatihan, lakukan langkah 3-8,

Perambatan Maju :

Langkah 3 : Tiap unit masukan (x_i, $i = 1,\ldots, n$) menerima sinyal x_i dan menghantarkan sinyal ini ke semua unit lapisan di atasnya (unit tersembunyi),

$$z_in_j = v_{oj} + \sum_{i=1}^{n} x_i\, v_{ij} \quad \text{..... (1)}$$

Langkah 4 : Setiap unit tersembunyi (x_i, $i = 1,\ldots, p$) jumlahkan bobot sinyal masukannya,

v_{oj} = bias pada unit tersembunyi j aplikasikan fungsi aktivasinya untuk menghitung sinyal keluarannya, $z_j = f(z_in_j)$, dan kirimkan sinyal ini keseluruh unit pada lapisan diatasnya (unit keluaran).

Langkah 5: Tiap unit keluaran (y_k, $k = 1,\ldots, m$) jumlahkan bobot sinyal masukannya,

$$y_in_k = w_{ok} + \sum_{j=1}^{n} z_j\, w_{jk} \quad \text{..... (2)}$$

w_{ok} = bias pada unit keluaran k dan aplikasikan fungsi aktivasinya untuk menghitung sinyal keluarannya, $y_k = f(y_in_k)$.

Perambatan Mundur :

Langkah 6 : Tiap unit keluaran (y_k, $k = 1,\ldots, m$) menerima pola target yang saling berhubungan pada masukan pola pelatihan, hitung kesalahan informasinya,

$$\delta_k = (t_k - y_k) f'(y_in_k) \quad \text{..... (3)}$$

hitung koreksi bobotnya (digunakan untuk memperbaharui wjk nantinya),

$$\Delta w_{jk} = \alpha\, \delta_k z_j \quad \text{..... (4)}$$

hitung koreksi biasnya (digunakan untuk memperbaharui w_{ok} nantinya), dan kirimkan δ_k ke unit-unit pada lapisan dibawahnya,

Langkah 7 : Setiap unit lapisan tersembunyi (z_j, $j = 1,\ldots, p$) jumlahkan hasil perubahan masukannya (dari unit-unit lapisan diatasnya),

$$\Delta_in_j = \sum_{k=1}^{m} \delta_k w_{jk} \quad \text{..... (5)}$$

kalikan dengan turunan fungsi aktivasinya untuk menghitung informasi kesalahannya,

$$\delta_j = \delta_in_j \, f^{l}(z_in_j) \quad \text{..... (6)}$$

hitung koreksi bobotnya (digunakan untuk memperbaharui v_{oj} nanti),

Langkah 8 : Tiap unit keluaran (y_k, $k = 1,\ldots, m$) update bias dan bobotnya ($j = 0,\ldots, p$) :

$$w_{jk}\,(\text{baru}) = w_{jk}\,(\text{lama}) + \Delta\, w_{jk} \quad \text{..... (7)}$$

Tiap unit lapisan tersembunyi (z_j, $j = 1,\ldots, p$) update bias dan bobotnya ($I = 0,\ldots,n$) :

$$v_{ij}\,(\text{baru}) = v_{ij}\,(\text{lama}) + \Delta\, v_{ij} \quad \text{..... (8)}$$

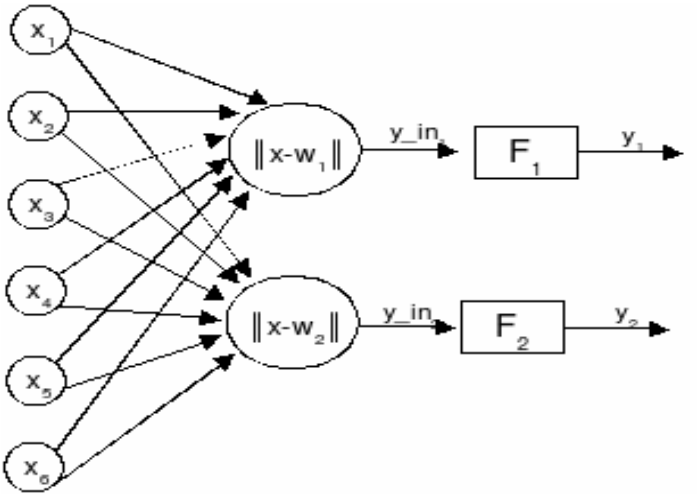
Langkah 9 : *Test* kondisi berhenti.

Keterangan Rumus:

z_in	:	Bobot sinyal unit tersembunyi
z	:	Keluaran unit tersembunyi
y_in	:	Bobot sinyal lapisan keluaran
y	:	Keluaran unit keluaran
δ	:	Kesalahan
Δw	:	Koreksi bobot unit keluaran
Δv	:	Koreksi bobot unit tersembunyi

E. Learning vector quantization (LVQ)

LVQ merupakan jaringan syaraf dengan tipe arsitektur jaringan lapis-tunggal umpan-maju (*Single Layer Feedforward*) yang terdiri atas unit masukan dan unit keluaran. Suatu lapisan kompetitif akan secara otomatis belajar untuk mengklasifikasikan vektor-vektor masukan . Kelas-kelas yang didapatkan sebagai hasil dari lapisan kompetitifini hanya tergantung pada jarak antara vektor vektor masukan. Jika 2 vektor masukan mendekati sama, maka lapisan kompetitif akan meletakkan kedua vektor masukan tersebut ke dalam kelas yang sama. Arsitektur jaringan *learning vector quantization* seperti terlihat pada Gambar II.13.[10]



***Gambar II.13 Learning vector quantization*[10]**

Gambar II.13 memperlihatkan bahwa yang bertindak sebagai dendrit atau data masukan adalah X1-X6, yang bertindak sebagai sinapsis atau bobot adalah

W, sedangkan soma atau badan sel dari jaringan ini adalah perhitungan $\|X-W\|$. Dan yang bertindak sebagai akson atau data keluaran adalah Y. Berikut ini algoritma pembelajaran metode *learning vector quantization*:

Langkah 0 : Inisialisasikan vektor referensi dan *learning* rate(α).

Langkah 1 : Selama kondisi berhenti bernilai salah, kerjakan langkah 2 dan 3. (Kondisi berhenti jika perulangan mencapai maksimal perulangan dan perubahan rasio lebih kecil dari minimal eror)

Langkah 2 : Untuk masing-masing pelatihan vektor masukan , kerjakan:

a. Temukan J sehingga $\|x-Wj\|$ bernilai minimum.

$$C_j = \sqrt{\sum_{i=1}^{n}\left(x_i - w_{ij}\right)^2} \quad \text{..... (9)}$$

b. Perbaiki Wj dengan :

i. Jika T=Cj maka

Wj(baru) = Wj(lama) + α [x - Wj(lama)] ..... (10)

ii. Jika T≠Cj maka

Wj(baru) = Wj(lama) - α [x - Wj(lama)] ..... (11)

Langkah 3 : Kurangi *learning* rate.

$$\alpha = \alpha * pengurang\ rasio$$

Langkah 4: Tes kondisi berhenti.

Keterangan rumus:

T	:	Target
J	:	Jumlah selisih data dan bobot
C	:	Kelas selisih bobot terkecil
W	:	Bobot
α	:	Rasio Pembelajaran
x	:	data

2.2.3.2 Pembelajaran Tak Terawasi (*Unsupervised Learning*)

Metode pembelajaran tak terawasi merupakan metode pembelajaran yang tidak memerlukan target keluaran. Pada metode ini, tidak dapat ditentukan hasil yang seperti apakah yang diharapkan selama proses pembelajaran. Selama proses pembelajaran, nilai bobot disusun dalam suatu range tertentu tergantung pada nilai masukan yang diberikan. Tujuan pembelajaran ini adalah mengelompokkan unit-unit yang hampir sama dalam suatu area tertentu. Pembelajaran ini biasanya sangat cocok untuk pengelompokan pola.[10]

A. Jaringan Kohonen

Jaringan kohonen pertama kali diperkenalkan oleh Prof. Teuvo Kohonen pada tahun 1982. Pada jaringan ini, suatu lapisan yang berisi neuron-neuron akan menyusun dirinya sendiri berdasarkan masukan nilai tertentu dalam suatu kelompok yang dikenal dengan istilah pengelompokkan (*cluster*). Selama proses penyusunan diri, *cluster* yang memiliki vektor bobot paling cocok dengan pola masukan (memiliki jarak yang paling dekat) akan terpilih sebagai pemenang. Neuron yang menjadi pemenang beserta neuron-neuron tetangganya akan memperbaiki bobotnya.

2.3 Pengenalan Pola (Pattern Recognition)

Pola(*pattern*) merupakan contoh model sesuatu yang dapat disalin. Tetapi pada saat menjelaskan berbagai jenis objek, baik untuk masalah yang tampak secara fisik ataupun abstrak, setiap hubungan antar data (analog/digital), konsep/kejadian, merupakan pola. Contoh masalah pengenalan pola adalah : pengenalan wajah, sebait musik, lukisan, perkataan yang direkam, tulisan tangan

dan lain sebagainya. Secara umum teknik pengenalan pola bertujuan mengklasifikasikan dan mendeskripsikan pola atau objek yang kompleks melalui pengukuran sifat atau ciri-ciri objek yang bersangkutan.[20]

Masalah pengenalan pola dapat dibagi menjadi dua bagian, yaitu bagian yang berhubungan dengan studi mekanisme pengenalan pola oleh manusia atau jasad hidup lainnya dan bagian mengenai perkembangan teori dan teknik untuk mendesain sebuah alat yang dapat melakukan tugas pengenalan secara otomatis. Beberapa contoh aplikasi pengenalan pola antara lain:

2.3.1 Pengenalan Suara (*Voice Recognition)*

Pengenalan suara merupakan salah satu upaya agar suara dapat dikenali atau diidentifikasi sehingga dapat dimanfaatkan. Pengenalan suara dapat dibedakan ke dalam tiga bentuk pendekatan, yaitu pendekatan akustik-fonetik *(the acoustic-phonetic approach*), pendekatan kecerdasan buatan (*the artificial intelligence approach*), dan pendekatan pengenalan pola (*the pattern recognition approach*). Pendekatan pengenalan pola terdiri dari dua langkah yaitu pembelajaran pola suara dan pengenalan pola melalui perbandingan pola. Tahap perbandingan pola adalah tahap bagi ucapan yang akan dikenali, dibandingkan polanya dengan setiap kemungkinan pola yang telah dipelajari dalam fase pembelajaran, untuk kemudian diklasifikasikan dengan pola terbaik yang cocok.

2.3.2 Pengenalan Ucapan (*Speech Recognition*)

Pengenalan pola suara adalah salah satu aplikasi yang dikembangkan dengan dasar pengenalan suara (*voice recognition*). Sistem ini mengijinkan kita untuk berkomunikasi antara manusia dengan memasukkan data ke komputer.

Meningkatkan efisiensi industri manufaktur, mengontrol mesin dengan berbicara pada mesin itu. Secara proses, kata yang diucapkan adalah digital (berubah menjadi urutan angka) dan cocok dengan kamus kode untuk mengidentifikasi kata-kata. Kebanyakan sistem harus "dilatih," sampel membutuhkan dari semua kata yang sebenarnya yang akan digunakan oleh pengguna sistem. Kata-kata sampel digital, disimpan di komputer dan digunakan untuk perbandingan untuk kata-kata yang baru. Sistem canggih lainnya memerlukan suara sampel, tetapi bukan dari setiap kata. Sistem ini menggunakan sampel suara bersama dengan kamus kosakata yang lebih besar untuk mencocokkan kata-kata yang masuk. Namun sistem lain bertujuan untuk menjadi "pembicara-independen" yaitu mereka akan mengenali kata-kata dalam kosakata mereka dari speaker tanpa pelatihan. Dikatakan (tahun 1994) bahwa komputer perlu sesuatu seperti 1000 kali lebih cepat sebelum kosakata besar (beberapa ribu kata-kata), speaker-independen, pengenalan suara suara yang terhubung akan layak.

2.3.3 Pengenalan Citra (*Image Recognition*)

Citra (*image*) adalah gambar pada bidang dua dimensi yang dihasilkan dari gambar analog dua dimensi yang kontinyu menjadi gambar diskrit melalui proses digitasi. Citra sebagai keluaran dari suatu sistem perekam data dapat bersifat :

1. Optik berupa foto
2. Analog berupa sinyal video seperti gambar pada TV
3. Digital yang dapat langsung disimpan pada storage device

Sebagai hasil keluaran, citra dapat berupa :

1. Citra diam *(still image)*
2. Citra bergerak *(moving image)*

Contoh citra diam dapat dilihat pada Gambar II.14.

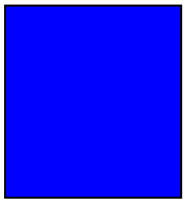


Gambar II.14 Gambar Diam (*Still Image*)[6]

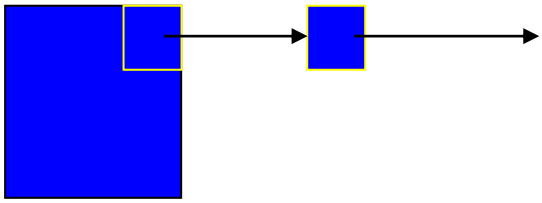
Pengenalan(*recognition)* yaitu suatu tindakan untuk mengenali sesuatu, baik benda maupun lainnya. Pengenalan Citra(*image recognition*) adalah kemampuan mesin atau program untuk mengidentifikasi pola dan bentuk dalam suatu gambar dan mengkonversikannya ke format yang dapat dibaca oleh mesin. Pengembangan dari pengenalan citra antara lain pengenalan wajah, pengenalan sidik jari, pengenalan tulisan tangan, pengenalan tanda rangan dan lain sebagainya.

Sebuah citra digital memiliki tingkat kedalaman warna 24 bit, pada setiap pixelnya terdiri dari 3 unsur warna yaitu *Red* (R), *Green* (G) dan *Blue* (B) yang masing-masing memiliki kedalaman warna 8 bit (0-255).[6] Berikut ini merupakan ilustrasinya:

1. Sebagai contoh, berikut ini gambar sebuah persegi dengan ukuran 3x3 px (kotak berwarna biru).



2. Setiap pixel terdiri dari 3 unsur warna yaitu red, green dan blue yang masing-masing memiliki kedalaman warna 8 bit (0-255).



Nilai pada ketiga unsur warna tersebut (RGB) merupakan kode yang ditentukan oleh suatu badan yang disebut *International Electrotechnical Commision* (IEC) untuk mewakili setiap kombinasi warna pada komputer.

Untuk dapat digunakan pada sistem pengenalan citra, citra digital harus diubah formatnya kedalam bentuk yang dapat dibaca oleh mesin, yaitu bentuk biner, atau dengan kata lain menjadi citra dengan kedalaman warna 2 bit (0 dan 1). Untuk mengubah citra 24 bit menjadi 2 bit, citra perlu melewati tahap *preprocessing*, tahap ini berfungsi untuk menyederhanakan citra sehingga dapat digunakan untuk proses pengenalan citra.

Mengubah citra dari 24 bit menjadi 2 bit tidak dapat dilakukan secara langsung karena hal tersebut akan menghilangkan informasi-informasi penting yang terdapat dalam citra tersebut. Untuk itu pertama-tama kedalaman warna perlu disederhanakan menjadi keabu-abuan (*grayscale*) atau dengan kata lain 8 bit. Setelah warna disederhanakan menjadi keabu-abuan, dilakukan proses untuk mendapatkan warna hitam dan putih, yaitu melalui proses deteksi tepi. Hasil dari

proses deteksi tepi berupa citra dengan warna hitam dan putih (0 dan 255), setelah itu citra melalui proses thresholding untuk mengubah nilai pixel menjadi biner (0 dan 1). Setelah citra sudah merubah menjadi 2 bit/biner, maka citra sudah dapat digunakan untuk perhitungan-perhitungan lain seperti pengenalan citra.

2.3.3.1 Pengolahan Citra

Pengolahan citra merupakan kegiatan memperbaiki kualitas citra agar mudah diinterpretasi oleh manusia/mesin (komputer). masukannya adalah citra dan keluarannya juga citra tapi dengan kualitas lebih baik daripada citra masukan, misal citra warnanya kurang tajam, kabur (*blurring*), mengandung *noise* (misal bintik-bintik putih), sehingga perlu ada pemrosesan untuk memperbaiki citra karena citra tersebut menjadi sulit diinterpretasikan karena informasi yang disampaikan menjadi berkurang. Operasi-operasi pada pengolahan citra diterapkan pada citra bila[6] :

1. Perbaikan atau memodifikasi citra dilakukan untuk meningkatkan kualitas penampakan citra/menonjolkan beberapa aspek informasi yang terkandung dalam citra *(image enhancement)*. Contoh : perbaikan kontras gelap/terang, perbaikan tepian objek, penajaman, pemberian warna semu, dll
2. Adanya cacat pada citra sehingga perlu dihilangkan/diminimumkan *(image restoration)*. Contoh : penghilangan kesamaran (*debluring*) citra tampak kabur karena pengaturan fokus lensa tidak tepat / kamera goyang, penghilangan *noise*.
3. Elemen dalam citra perlu dikelompokkan, dicocokan atau diukur *(image segmentation).* Operasi ini berkaitan erat dengan pengenalan pola.

4. Diperlukannya ekstraksi ciri-ciri tertentu yang dimiliki citra untuk membantu dalam pengidentifikasian objek *(image analysis).* Proses segementasi kadangkala diperlukan untuk melokalisasi objek yang diinginkan dari sekelilingnya. Contoh : pendeteksian tepi objek
5. Sebagian citra perlu digabung dengan bagian citra yang lain *(image reconstruction)*. Contoh : beberapa foto rontgen digunakan untuk membentuk ulang gambar organ tubuh
6. Citra perlu dimampatkan *(image compression)*

   Contoh : suatu file citra berbentuk BMP berukuran 258 KB dimampatkan dengan metode JPEG menjadi berukuran 49 KB.
7. Menyembunyikan data rahasia (berupa teks/citra) pada citra sehingga keberadaan data rahasia tersebut tidak diketahui orang *(steganografi* dan *watermarking)*

Beberapa jenis operasi pengolahan citra adalah sebagai berikut:

1. Modifikasi Kecemerlangan(*Brightness Modification*)

   Mengubah nilai keabuan/warna dari gelap menuju terang atau sebaliknya mengubah citra yang terlalu cemerlang/pucat menjadi gelap.
2. Peningkatan Kontras (*Contast Enhancement*)

   Dengan peningkatan kontras maka titik yang cenderung gelap menjadi lebih gelap dan yang cenderung terang menjadi lebih cemerlang.
3. Negasi

   Operasi untuk mendapatkan citra negatif (*negative image*)
4. Pengabuan (*grayscale*)

Merupakan proses konversi citra dengan warna sebenarnya (*true color*) menjadi citra keabuan (*grayscale*).

5. Pengambangan (*Thresholding*)

   Operasi pengambangan digunakan untuk mengubah citra dengan format skala keabuan, yang mempunyai kemungkinan nilai lebih dari 2 ke citra biner yang memiliki 2 buah nilai (yaitu 0 dan 1).

6. Pencerminan (*Flipping*)

   Pencerminan merupakan proses menggambar citra ke bentuk kebalikannya seperti ketika sedang bercermin.

7. Rotasi (*Rotating*)

   Rotasi yaitu proses memutar koordinat citra sesuai derajat yang ditentukan.

8. Pemotongan (*Cropping*)

   Memotong satu bagian dari citra sesuai kebutuhan.

9. Pengskalaan (*Scaling*)

   Mengubah ukuran citra menjadi lebih besar atau lebih kecil.

10. Deteksi Tepi (*Edge Detection*)

    Deteksi tepi *(edge detection)* pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra

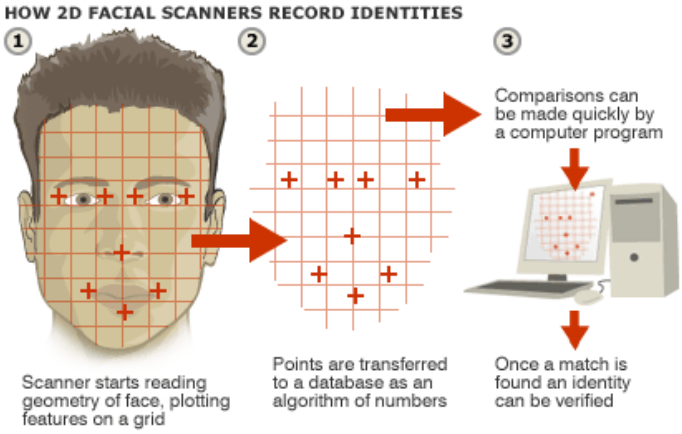
2.3.3.2 Pengenalan Wajah (*Face Recognition*)

Subyek pengenalan wajah seumur dengan visi komputer, baik karena kepentingan praktis dari topik dan keteoritisan dari ilmuwan kognitif. Meskipun kenyataan bahwa metode lain untuk identifikasi (seperti sidik jari atau *scan* iris)

dapat lebih akurat, pengenalan wajah selalu tetap menjadi fokus utama penelitian karena bersifat non-invasif dan merupakan metode identifikasi manusia.

Mungkin contoh awal dari suatu sistem pengenalan wajah adalah Kohonen, yang menunjukkan bahwa jaringan saraf sederhana dapat melakukan pengenalan wajah dengan selaras dan normalisasi gambar wajah. Sistem yang bekerja pada jenis jaringan ini yaitu dengan menghitung gambaran wajah eigenvektor yang autokorelasi dengan matriks, eigenvektor ini sekarang dikenal sebagai "*eigenfaces*" Pada tahun berikutnya banyak peneliti mencoba skema pengenalan wajah didasarkan pada tepi, jarak antar-fitur, dan pendekatan jaringan saraf.

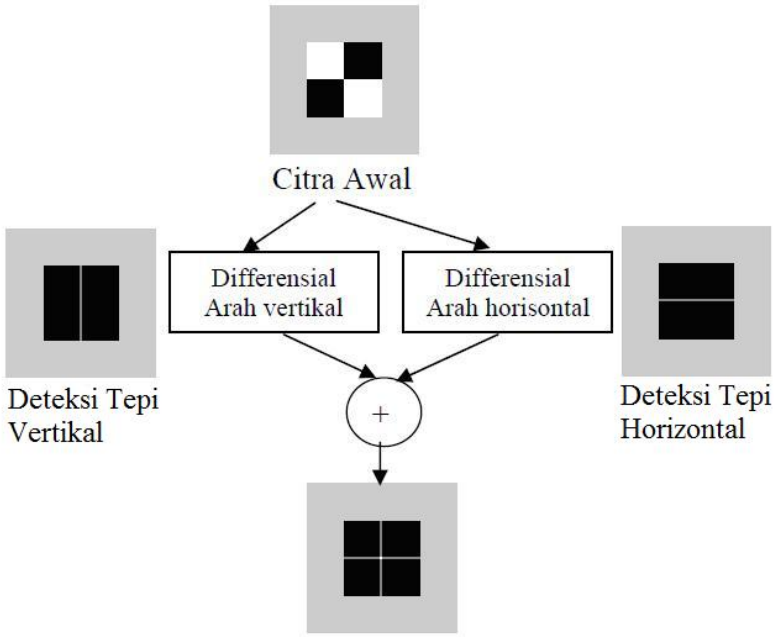
Pada perkembangannya, pengenalan wajah adalah sebuah sistem yang mengenali gambar wajah manusia yang biasanya digunakan dalam otomatisasi dan *security* sebuah industri. Gambar II.15 merupakan Gambar proses pengenalan wajah.



Gambar II.15 Pengenalan Wajah[22]

2.3.3.3 Deteksi Tepi (*Edge Detection*)

Deteksi tepi *(edge detection)* pada suatu citra adalah suatu proses yang menghasilkan tepi-tepi dari obyek-obyek citra, Tepi-tepi ini akan menandai bagian detail citra. Suatu titik (x,y) dikatakan sebagai tepi *(edge)* dari suatu citra bila titik tersebut mempunyai perbedaan yang tinggi dengan tetangganya. Gambar II.16 menggambarkan bagaimana tepi suatu gambar diperoleh[15].



Gambar II.16 Deteksi Tepi[15]

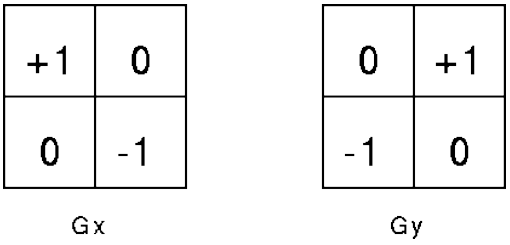
Macam-macam metode untuk proses deteksi tepi, antara lain:

1. Metode Robert

Metode pendektesian tepi dengan mencari perbedaan (*differensial*) pada arah horisontal dan perbedaan pada arah vertikal, dengan ditambahkan proses konversi biner setelah dilakukan perbedaan. Agar mendapatkan tepi-tepi yang lebih baik, maka konversi biner dilakukan dengan meratakan distribusi warna hitam dan putih atau dengan kata lain objek gambar yang akan digunakan untuk

metode ini sebaiknya adalah gambar hitam putih. Kelebihan operator ini yaitu cepat dan mudah untuk menghitung pengukuran gradien 2-D spasial pada gambar. Pixel nilai di setiap titik di keluaran mutlak mewakili besarnya estimasi gradien spasial citra masukan pada saat itu.

Operator robert terdiri dari sepasang kernel 2 × 2 konvolusi. Satu kernel yang lain hanya diputar oleh 90 °. Ini sangat mirip dengan operator Sobel. Matriks dari metode robert terlihat pada Gambar II.17.



Gambar II.17 Matriks Metode Robert[15]

Kernel ini dirancang untuk merespon secara maksimal untuk tepi berjalan pada 45 ° ke jaringan pixel, satu kernel untuk masing-masing dua orientasi tegak lurus. Kernel dapat diterapkan secara terpisah dengan citra masukan, untuk menghasilkan pengukuran yang terpisah dari komponen gradien di setiap orientasi (Gx dan Gy). Ini kemudian dapat digabungkan bersama-sama untuk mencari besarnya nilai mutlak gradien pada setiap titik dan orientasi gradien itu. Besarnya gradien diberikan oleh:

$$|G| = \sqrt{Gx^2 + Gy^2} \quad \text{.....(12)}$$

walaupun biasanya, sebuah besaran perkiraan dihitung dengan menggunakan:

$$|G| = |Gx| + |Gy| \quad \text{.....(13)}$$

yang lebih cepat untuk menghitung.

Sudut orientasi tepi menimbulkan gradien spasial (relatif terhadap orientasi grid pixel) diberikan oleh:

$$\theta = \arctan(Gy/Gx) - 3\pi/4 \quad \text{.....(14)}$$

2. Metode Prewitt

Metode ini adalah pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. Metode ini mengambil prinsip dari fungsi laplacian yang dikenal sebagai fungsi untuk membangkitkan HPF. Sehingga tepi-tepi yang dihasilkan lebih banyak dari metode Robert. Gambar II.18 merupakan matriks metode prewitt.

$s_x =$

-1	0	1
-1	0	1
-1	0	1

$s_y =$

1	1	1
0	0	0
-1	-1	-1

Gambar II.18 Matriks Metode Prewitt[15]

Filter HPF menciptakan sebuah gambar di mana tepi (perubahan tajam nilai tingkat keabuan) yang akan ditampilkan. Hanya filter dengan ukuran 3x3 yg dapat digunakan dengan filter ini. Filter ini menggunakan dua template 3x3 untuk menghitung nilai Prewitt gradien seperti yang ditunjukkan di bawah ini:

Template:

$$\begin{bmatrix} -101 \\ -101 \\ -101 \end{bmatrix}$$ sebagai X .....(15)

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$ sebagai Y.....(16)

Gunakan template untuk jendela filter 3x3.

$$\begin{bmatrix} a1 & a2 & a3 \\ a4 & a5 & a6 \\ a7 & a8 & a9 \end{bmatrix}$$ 3x3 filter jendela.....(17)

dimana a1 .. a9 adalah tingkat keabuan dari setiap piksel di jendela filter.

X = -1 * a1 + 1 * a3 - 1 * a4 + 1 * A6 - 1 * A7 + 1 * A9.....(18)

Y = 1 * a1 + 1 * a2 + 1 * a3 - 1 * A7 - 1 * A8 - 1 * A9.....(19)

Prewitt Gradient = sqrt (X * X + Y * Y).....(20)

3. Metode Sobel

Metode ini juga merupakan pengembangan metode robert. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi *noise* sebelum melakukan perhitungan deteksi tepi sehingga tepi-tepi yang dihasilkan lebih banyak dibanding 2 metode sebelumnya. Operator terdiri dari sepasang kernel 3 × 3 konvolusi seperti yang ditunjukkan pada Gambar II.19. Satu kernel yang lain hanya diputar oleh 90 °.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

Gy

Gambar II.19 Matriks Metode Sobel[15]

Kernel ini dirancang untuk merespon secara maksimal untuk tepi berjalan secara vertikal dan horizontal relatif terhadap grid pixel, satu kernel untuk masing-masing dua orientasi tegak lurus. Kernel dapat diterapkan secara terpisah dengan citra masukan, untuk menghasilkan pengukuran yang terpisah dari komponen gradien di setiap orientasi (menyebut Gx dan Gy). Ini kemudian dapat digabungkan bersama-sama untuk mencari besarnya nilai mutlak gradien pada setiap titik dan orientasi gradien itu. Besarnya gradien diberikan oleh:

$$|G| = \sqrt{Gx^2 + Gy^2} \quad \text{.....(21)}$$

Biasanya, sebuah perkiraan besarnya dihitung dengan menggunakan:

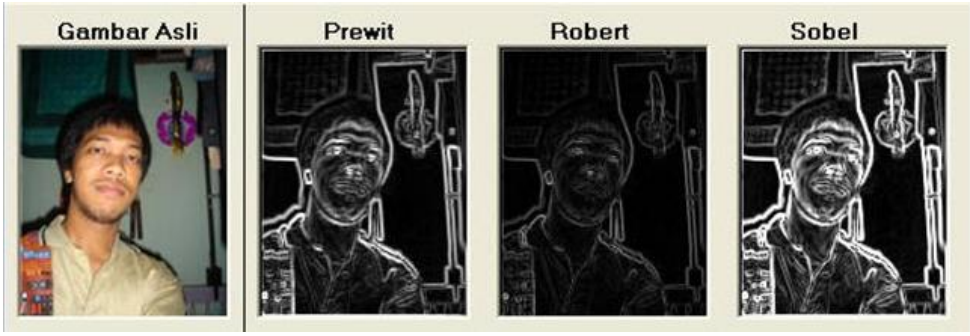
$$|G| = |Gx| + |Gy| \quad \text{.....(22)}$$

yang lebih cepat untuk menghitung.

Sudut orientasi tepi (relatif ke grid pixel) menimbulkan gradien spasial diberikan oleh:

$$\theta = \arctan(Gy/Gx) \quad \text{.....(23)}$$

Melihat dari perbandingan 3 metode yaitu Roberts, Prewitt, dan Sobel, maka keluaran yang dihasilkan dapat dilihat pada Gambar II.20.



Gambar II.20 Keluaran Metode Prewitt, Sobel, dan Robert [8]

2.4 Basis Data

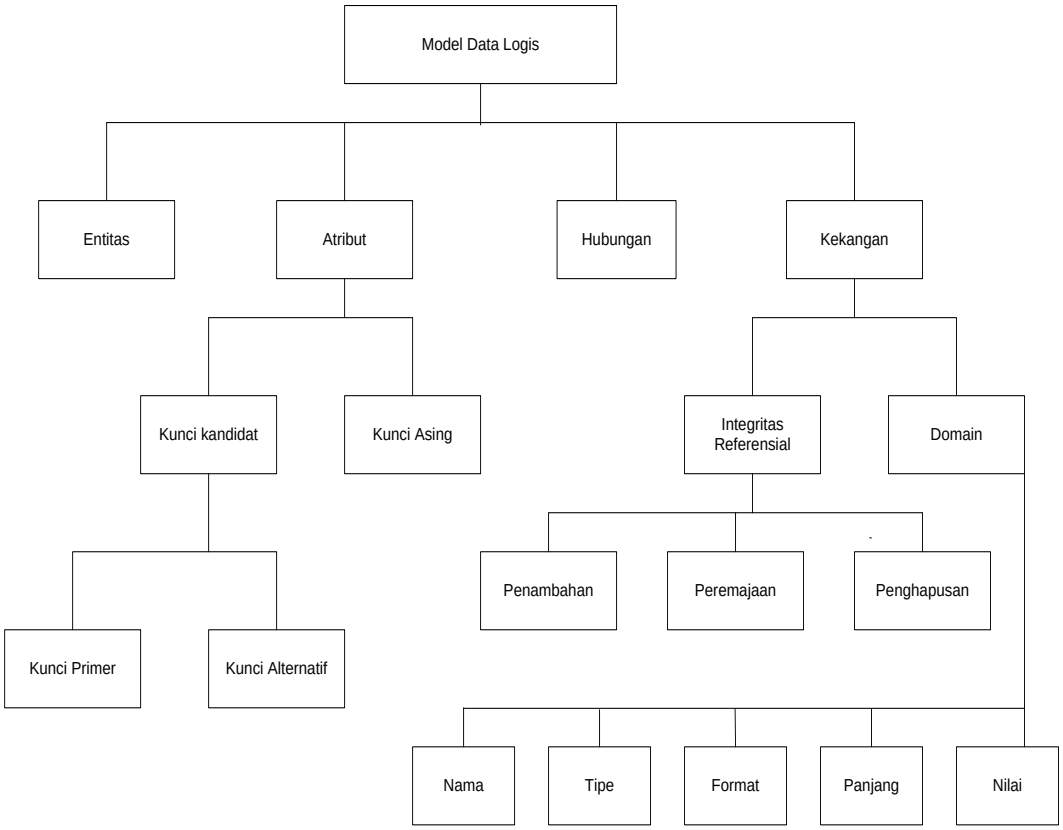
Basis data adalah sekumpulan data persistence yang saling terkait, menggambarkan suatu organisasi (*enterprise*). Sistem Basis data (DBS) suatu sistem yang mengelola data dan menyediakan data tersebut apabila dibutuhkan.[16]

Komponen utama Sistem Basis data[9]:

1. Data yang disimpan dalam basis data.
2. H/W : *storage, processor, memory*.
3. S/W : DBMS, *Report-writer, design arts*, dll.
4. Penggguna :
    a. Pengguna Awam (*Naive User*)
    b. Pengguna Biasa (*Casual User*)
    c. Programmer
    d. Pengujiistrator

2.4.1 Model Entitas

Peter Pin-Shan Chen, "*The Entity-Relationship Model-Toward a unifield view of data*", 1976. Model data E-R didasarkan pada persepsi bahwa dunia nyata merupakan sekumpulan dari sejumlah obyek dasar (entitas) dan relasi antar obyek-obyek data tersebut. Termasuk dalam kelompok *object-based logical*. Merupakan contoh kelas *Extended* atau *semantic models*. Komponen pada perancangan basis data secara konseptual dijelaskan pada Gambar II.21[17].



Gambar II.21 Komponen pada perancangan basis data secara Konseptual[17]

1. Entitas(*Entity*)

Sebuah entitas adalah sebuah obyek yang dapat dibedakan dari obyek-obyek lainnya. Pengelompokkan entitas antara lain:

a. Entitas konkrit / nyata dan entitas abstrak.

b. Entitas biasa (*Regular/Strong*) dan

c. Entitas lemah (*Weak*).

Himpunan entitas (*Entity set*) adalah kumpulan sejumlah entitas yang memiliki tipe yang sama. Sebuah entitas memiliki sejumlah properti (atribut). Setiap atribut memiliki sekumpulan nilai yang diizinkan, yang disebut domain.

Sebuah basis data mengandung sekumpulan himpunan entitas, yang masing-masingnya memiliki sejumlah entitas dari tipe yang sama.

2. Batasan(Kekangan)

Kekangan digunakan untuk melindungi integritas data (misalnya, melindungi kesalahan sewaktu pengisian data).

3. Atribut

a. Key Vs Non Key

Key adalah jika atribut tersebut mampu mewakili identitas keunikan dari suatu entitas. *Key* dapat dibangun dari satu atribut atau gabungan beberapa atribut. *Key* terbagi atas beberapa jenis, diantaranya :

a) Super Key

b) Primary Key

c) Candidate Key

d) Foreign Key

b. Atribut sederhana Vs Atribut Komposit

Atribut sederhana adalah atribut yang tidak perlu di-*breakdown* kembali. Atribut Komposit adalah atribut yang perlu di-*breakdown* kembali.

c. Atribut bernilai tunggal Vs Banyak

Atribut bernilai tunggal adalah atribut yang isinya hanya 1, sementara atribut bernilai banyak adalah atribut yang memiliki isi banyak.

d. Atribut Mandatory dan Non Mandatory.

Atribut Mandatory adalah atribut yang harus diisi tidak boleh kosong, sebaliknya Non mandatory boleh null (kosong).

e. Atribut Turunan

Atribut yang nilainya dapat diproleh dari atribut lain/proses lain.

4. Relasi (*Relationship*)

Sebuah relasi menggambarkan suatu asosiasi antar sejumlah entitas. Himpunan relasi (*Relationship set*) adalah kumpulan sejumlah relasi yang memiliki tipe yang sama. Jumlah entitas terlihat dalam 2 buah relasi disebut derajat. Kebanyakan relasi yang muncul adalah relasi binary, ada beberapa yang ternary, lebih dari itu sangat jarang. Fungsi sebuah entitas di dalam relasi disebut peran (*role*). Sebuah relasi dapat memiliki atribut.

5. Kardinalitas Relasi (Derajat Hubungan)

Relasi yang dimungkinkan antar entitas, antara lain:

a. 1 – 1
b. 1 – N
c. N – 1
d. N – N/N – M

6. Diagram E- R

Diagram E-R menggambarkan struktur lojik keseluruhan basis data. Simbol yang digunakan :

a. Persegi empat, merepresentasikan himpunan entitas (untuk entitas lemah diberi garis ganda).
b. Elips, merepresentasikan atribut.
c. Wajik, merepresentasikan himpunan keterhubungan.
d. Garis, menghubungkan simbol-simbol pada diagram.

e. Label dari persegi empat, elips, dan wajik menunjukkan nama.

f. Kardinalitas pemetaan dinyatakan dengan 2 cara :

   a) [*Korth*] garis berarah (1) dan garis tidak berarah (Banyak).

   b) [*Date*] menuliskan kardinalitasnya pada garis.

g. Peran dapat dituliskan sebagai label dari garis.

7. Varian Entitas

   a. Entitas Kuat : Keberadaannya tidak tergantung dari entitas lain.

   b. Entitas Lemah : Keberadaanya tergantung dari entitas lain.

   c. *Sub Entitas* (Generalisasi dan Spesialisasi)

   Generalisasi dibentuk dengan metode *Bottom Up*, sementara Spesialisasi dengan metode *Top Down*.

8. Varian Relasi

   a. Relasi Tunggal

      Relasi yang dihasilkan dari 1 *Entity* set.

   b. Relasi Multi entitas

      Relasi yang terjadi apabila memang ketiganya saling berelasi.

   c. Relasi Ganda

      Relasi yang dihasilkan oleh 2 *Entity* yang saling terhubung pada 1 proses.

   d. Agregasi

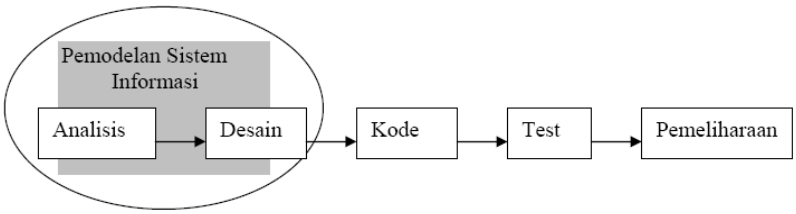
      Relasi yang tergantung pada keberadaan relasi lain.

2.5 Metode Pengembangan Perangkat Lunak

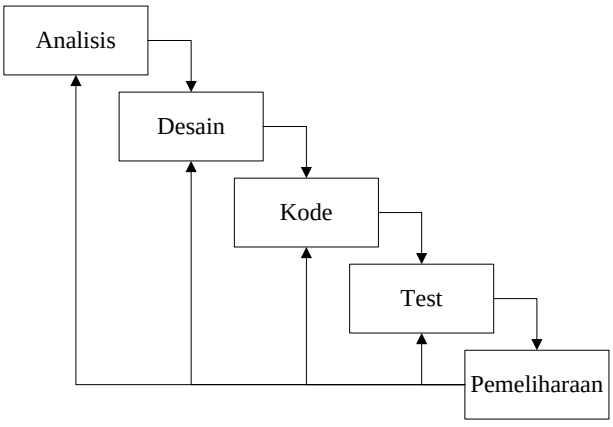
Metode pengembangan perangkat lunak memiliki beberapa jenis model proses dalam membangun suatu software. Adapun macamnya yaitu :

2.5.1 *Linear Sequential Model/ Waterfall Model*

Linear Sequential Model (Model Sekuensial Linier) atau sering disebut *Waterfall Model* (Model Air Terjun) merupakan paradigma rekayasa perangkat lunak yang paling tua dan paling banyak dipakai. Model ini mengusulkan sebuah pendekatan perkembangan perangkat lunak yang sistematik dan sekunsial yang dimulai pada tingkat dan kemajuan sistem pada seluruh analisis, desain , kode, pengujian, dan pemeliharaan. Tahapan dari model sekuensial linear ditunjukkan pada Gambar II.22 dan II.23 berikut ini[1].



Gambar II.22 Tahapan Model Sekuensial Linear[1]



Gambar II.23 Tahapan Model Sekuensial Linear[1]

A. Tahapan-tahapan Model Sekuensial Linier

Model Sekunsial Linier mengikuti aktivitas-aktivitas yaitu:

1. Rekayasa dan Pemodelan Sistem/Informasi

   Karena perangkat lunak merupakan bagian dari suatu sistem maka langkah pertama dimulai dengan membangun syarat semua elemen sistem dan mengalokasikan ke perangkat lunak dengan memeperhatiakn hubungannya dengan manusia, perangkat keras dan *database*.

2. Analisis Kebutuhan Perangkat Lunak

   Proses menganalisis dan pengumpulan kebutuhan sistem yang sesuai dengan domain informasi tingkah laku, unjuk kerja, dan antar muka (*interface)* yang diperlukan. Kebutuhan-kebutuhan tersebut didokumentasikan dan dilihat lagi dengan pelanggan.

3. Desain

   Proses desain akan menerjemahkan syarat kebutuhan ke sebuah perancangan perangkat lunak yang dapat diperkirakan sebelum dibuat coding. Proses ini berfokus pada struktur data, arsitektur perangkat lunak, representasi interface, dan detail (algoritma) prosedural.

4. Pengkodeaan (Coding)

   Pengkodean merupakan proses menerjemahkan desain ke dalam suatu bahasa yang bisa dimengerti oleh komputer.

5. Pengujian

   Proses pengujian dilakukan pada logika internal untuk memastikan semua pernyataan sudah diuji. Pengujian eksternal fungsional untuk menemukan

kesalahan-kesalahan dan memastikan bahwa masukan akan memberikan hasil yang aktual sesuai yang dibutuhkan

6. Pemeliharaan

Perangkat lunak yang sudah disampaikan kepada pelanggan pasti akan mengalami perubahan. Perubahan tersebut bisa karena mengalami kesalahan karena perangkat lunak harus menyesuaikan dengan lingkungan (periperal atau sistem operasi baru) baru, atau karena pelanggan membutuhkan perkembangan fungsional atau unjuk kerja.

B. Keunggulan dan Kelemahan Model Sekuensial Linier

Keunggulan dan kelemahan Model Sekuensial Linier adalah sebagai berikut:

1. Keunggulan

Keunggulan Model Sekuensial Linear adalah sebagai berikut:

a. Mudah aplikasikan
b. Memberikan template tentang metode analisis, desain, pengkodean, pengujian, dan pemeliharaan

2. Kelemahan

a. Jarang sekali proyek pada dunia nyata mengikuti aliran sekuensial yang dianjurkan model karena model ini bisa melakukan itersi tidak langsung. Hal ini berakibat ada perubahan yang diragukan pada saat proyek berjalan.
b. Pelanggan sulit untuk menyatakan kebutuhan secara eksplisit sehingga sulit untuk megakomodasi ketidakpastian pada saat awal proyek.

c. Pelanggan harus bersikap sabar karena harus menunggu sampai akhir proyrk dilalui. Sebuah kesalahan jika tidak diketahui dari awal akan menjadi masalah besar karena harus mengulang dari awal.

d. Pengembang sering malakukan penundaan yang tidak perlu karena anggota tim proyek harus menunggu tim lain untuk melengkapi tugas karena memiliki ketergantungan hal ini menyebabkan penggunaan waktu tidak efesien.

2.5.2 Object Oriented Programing (OOP)

Object-Oriented Programming (OOP) adalah sebuah pendekatan untuk pengembangan / *development* suatu software dimana dalam struktur software tersebut didasarkan kepada interaksi objek dalam penyelesaian suatu proses/tugas. Interaksi tersebut mengambil form dari pesan-pesan dan mengirimkannya kembali antar objek tersebut. Objek akan merespon pesan tersebut menjadi sebuah tindakan / *action* atau metode. Jika kita mencoba melihat bagaimana tugas disekitar kita diselesaikan, kita akan mengetahui bahwa kita berinteraksi dalam sebuah *object-oriented world*. Jika akan bepergian kita pasti berinteraksi dengan object mobil. Sebagai sebuah objek, mobil berisi objek-objek lain yang berinteraksi untuk melakukan tugasnya membawa kita. [13]

Object-oriented programs terdiri dari objek yang berinteraksi satu sama lainnya untuk menyelesaikan sebuah tugas. Seperti dunia nyata, pengguna dari *software programs* dilibatkan dari logika proses untuk menyelesaikan tugas. Contoh, ketika kamu mencetak sebuah halaman di *word processor*, kamu berarti

melakukan inisialisasi tindakan dengan mengklik tombol *printer*. Kemudian kamu hanya menunggu respon apakah *job* tersebut sukses atau gagal, sedangkan proses terjadi internal tanpa kita ketahui. Tentunya setelah kamu menekan tombol *printer*, maka secara simultan objek tombol tersebut berinteraksi dengan objek *printer* untuk menyelesaikan *job* tersebut.

Adapun karakteristik yang dimiliki OOP yaitu :

a. *Objects*

   Object adalah sebuah *structure* yang menggabungkan data dan prosedur untuk bekerja bersama-sama.

b. *Abstraction*

   Ketika berinteraksi dengan objek-objek di dunia ini, sering hanya berkonsentrasi dengan sebuah bagian dari propertinya. Tanpa kemampuan untuk mensarikan/abstrak atau menyaring untuk membuang objek properti yang asing / tidak ada hubungannya, maka kamu akan ditemukan kesulitan untuk memproses informasi.

   Ketika membangun objek dalam aplikasi OOP, adalah penting untuk menggabungkan konsep abstraction ini. Jika kamu membangun aplikasi *shipping*, kamu harus membangun objek produk dengan atribut seperti ukuran dan berat. Warna adalah contoh informasi yang tidak ada hubungannya dan harus dibuang. Tetapi ketika kamu membangun *order-entry application*, warna menjadi penting dan harus termasuk atribut objek produk.

c. Encapsulation

Ciri penting lainnya dari OOP adalah *encapsulation*. *Encapsulation* adalah sebuah proses dimana tidak ada akses langsung ke data yang diberikan, bahkan *hidden*. Jika kamu ingin mendapat data, kamu harus berinteraksi dengan object yang bertanggung jawab atas data tersebut.

d. Polymorphism

Polymorphisms adalah kemampuan 2 buah object yang berbeda untuk merespon pesan permintaan yang sama dalam suatu cara yang unik. Contohnya, kamu dapat mengirim pesan *print* ke objek *printer* yang akan mencetak pada *printer*, dan kamu juga dapat mengirim pesan yang sama ke *object screen* yang akan menuliskan pada screen monitor. Dalam OOP, kamu menerapkan tipe *polymorphism* melalui proses yang disebut *overloading*. Kamu dapat mengimplementasikan metode yang berbeda pada sebuah objek yang mempunyai nama yang sama.

e. Inheritance

Penggunaan *inheritance* dalam OOP untuk mengklasifikasikan objects dalam program sesuai karakteristik umum dan fungsinya. Hal ini akan membuat pekerjaan bersama objek lebih mudah dan lebih intuitif. Hal ini juga membuat *programming* lebih mudah karena memungkinkan kita untuk mengkombinasikan karakteristik umum kedalam objek *parent* dan mewariskan karakteristik ini ke *child object*.

2.5.3 UML (*Unified Modeling Language*)

Model ini digunakan oleh para pengembang profesional untuk berkomunikasi tentang proyek mereka kepada pembuat aplikasi mereka atau untuk pengembang aplikasi yang lain. *Unified Modeling Language* (UML) telah menjadi bagian penting dari pengembangan perangkat lunak sejak diperkenalkan pada tahun 1997. UML telah berkembang selama bertahun-tahun. Sebuah keuntungan penting dari UML adalah konsistensi, dan bagian dari konsistensi tersebut adalah penerapan model umum elemen-elemen di diagram yang berbeda. Terdapat 2 Tipe diagram UML yaitu *structural diagrams* dan *behavior diagrams*. Berikut ini penjelasannya[4]:

A. *Structural Diagrams*

Structural diagrams digunakan untuk mendeskripsikan relasi antar kelas. Diagram-diagram yang termasuk kedalam tipe ini adalah sebagai berikut:

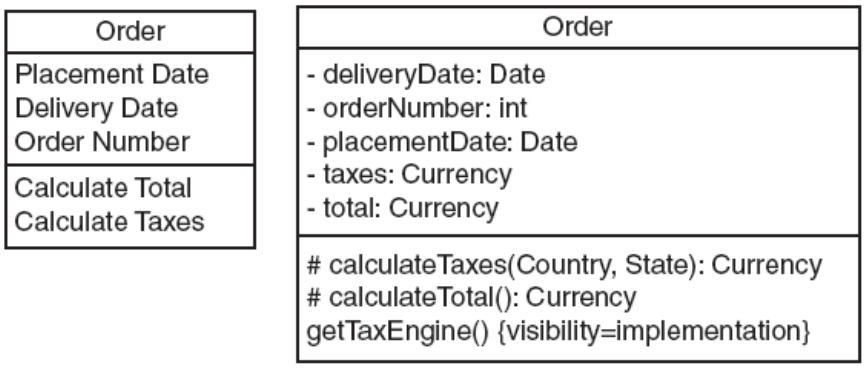
1. *Class Diagram*

   Class diagram (diagram kelas) dari sebuah UML menunjukkan kelas dari sistem, hubungan timbal balik antar kelas, operasi dan atribut setiap kelas. Diagram kelas digunakan untuk:

   a) Mengekplorasi konsep domain dalam bentuk model domain.
   b) Menganalisis persyaratan dalam bentuk analisis/konseptual model.
   c) Menggambarkan detail desain dari orientasi objek atau objek berdasarkan perangkat lunak. Sebuah model kelas terdiri dari satu atau lebih diagram kelas dan mendukung spesifikasi yang

menggambarkan unsur-unsur model, termasuk kelas, hubungan antar kelas dan *interface*.

Contoh diagram kelas digambarkan pada Gambar II.24 berikut ini:

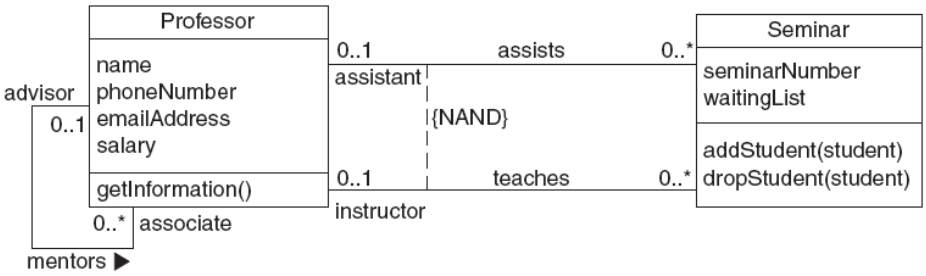


Gambar II.24 Diagram Kelas[4]

Kelas biasanya dimodelkan sebagai persegi panjang dengan tiga bagian: bagian atas untuk nama kelas, bagian tengah untuk atribut kelas, dan bagian bawah untuk metode kelas. Atribut adalah informasi yang disimpan tentang suatu objek sedangkan metode adalah hal-hal yang objek atau kelas lakukan. Kelas didalam kelas diagram memiliki relasi kepada kelas lain. Berikut ini relasi yang tedapat pada kelas diagram:

a) Asosiasi

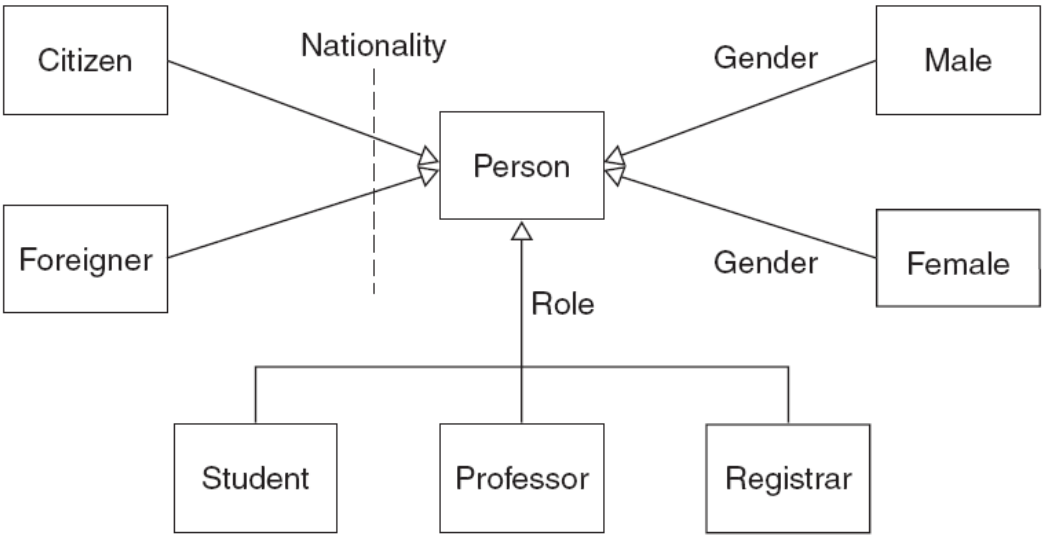
Objek yang sering dikaitkan dengan, atau terkait dengan, objek lain. Asosiasi dimodelkan sebagai garis yang menghubungkan dua kelas yang terlibat dalam suatu hubungan. Kepala panah pada akhir baris menunjukkan arah dari asosiasi. Sebuah baris dengan satu panah adalah uni-directional sedangkan garis dengan baik nol atau dua kepala panah adalah dua arah. Gambar jenis relasi asosiasi ditunjukkan pada Gambar II.25.



Gambar II.25 Relasi Asosiasi Pada Diagram Kelas[4]

b) Pewarisan

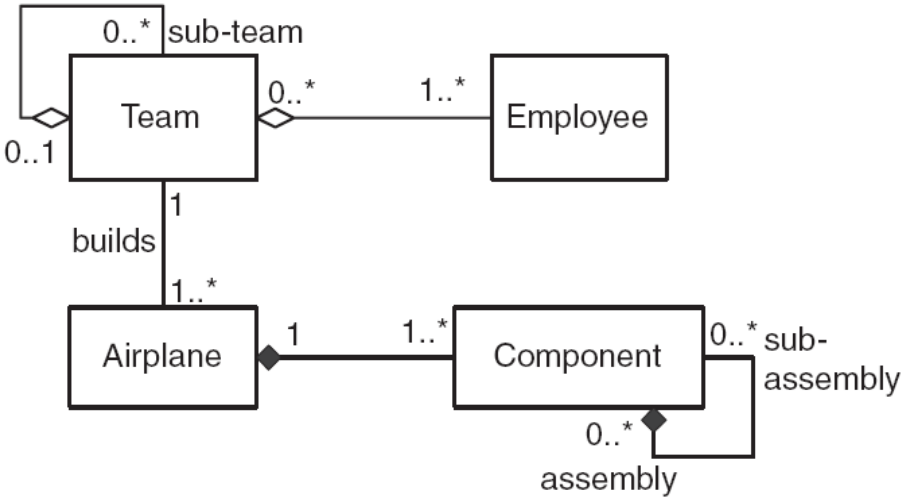
Pewarisan (*Inheritance*) sering juga disebut generalisasi, model hubungan "*is a*" dan "*is like*", memungkinkan untuk menggunakan kembali data dan kode yang ada. Ketika A mewarisi B, dapat dikatakan bahwa A adalah *subclass* B dan B adalah *superclass* dari A. Kemudian bila A mewarisi semua atribut dan metode B, maka dapat dikatakan warisan murni (*pure inheritance*). Notasi UML untuk menggambarkan pewarisan adalah sebuah garis dengan panah tertutup menunjuk dari subclass ke superclass.



Gambar II.26 Relasi Pewarisan Pada Diagram Kelas[4]

c) Agregasi dan Komposisi

Agregasi adalah spesialisasi dasar, menentukan hubungan keseluruhan-bagian diantara dua benda. Komposisi adalah bentuk yang lebih kuat dari agregasi, dimana keseluruhan dan bagian-bagian memiliki daya tahan yang sama, dan sangat umum untuk mengelola seluruh bagiannya. Gambar II.27 menggambarkan relasi agregasi dan komposisi pada diagram kelas.



Gambar II.27 Relasi Agregasi dan Komposisi Pada Diagram Kelas[4]

2. *Object Diagram*

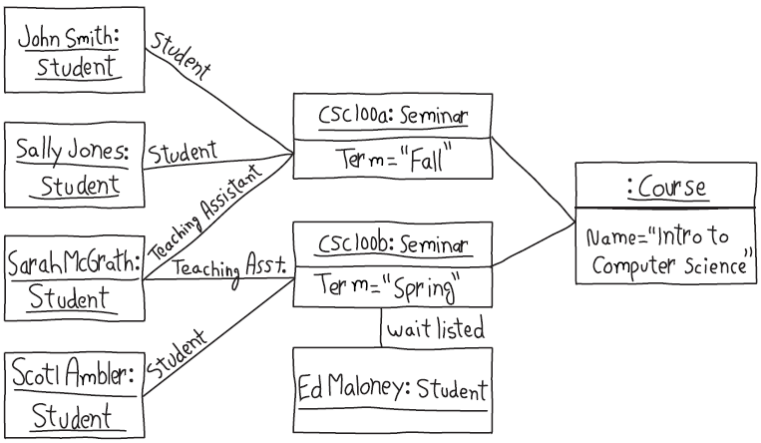
Object diagram UML menggambarkan kasus dan hubungan pada suatu titik waktu. *Object diagram* UML digunakan untuk:

a) Mencari tahu contoh nyata objek dan hubungannya.

b) Menjelaskan hubungan yang kompleks antara kelas.

c) Menjadi masukan untuk menggambarkan diagram kelas UML.

Object diagram pada UML digambarkan pada Gambar II.28 berikut ini:



Gambar II.28 *Object Diagram* Pada UML[4]

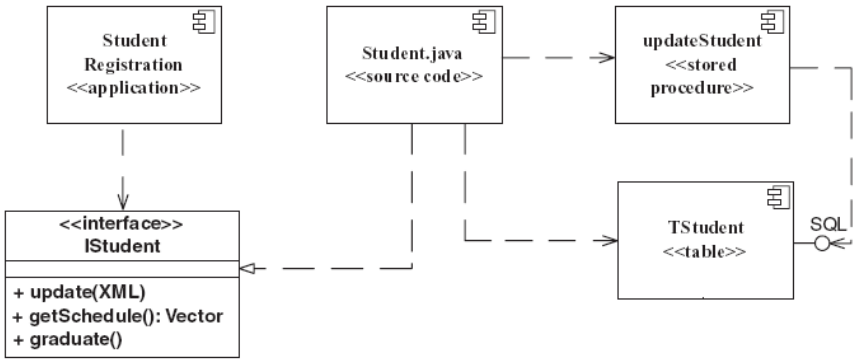
3. *Component Diagram*

Component diagram (Gambar II.29) pada UML menunjukkan ketergantungan antara komponen perangkat lunak, termasuk pengklasifikasian yang menentukan, seperti kelas implementasi dan artefak yang mengimplementasikannya, seperti kode program, file kode biner, *executable* file, *script* dan tabel. Diagram ini dibuat untuk:

a) Memodelkan desain level terbawah konfigurasi dari sistem.

b) Memodelkan infrastruktur teknis (Ambles 1998).

c) Memodelkan bisnis sistem.

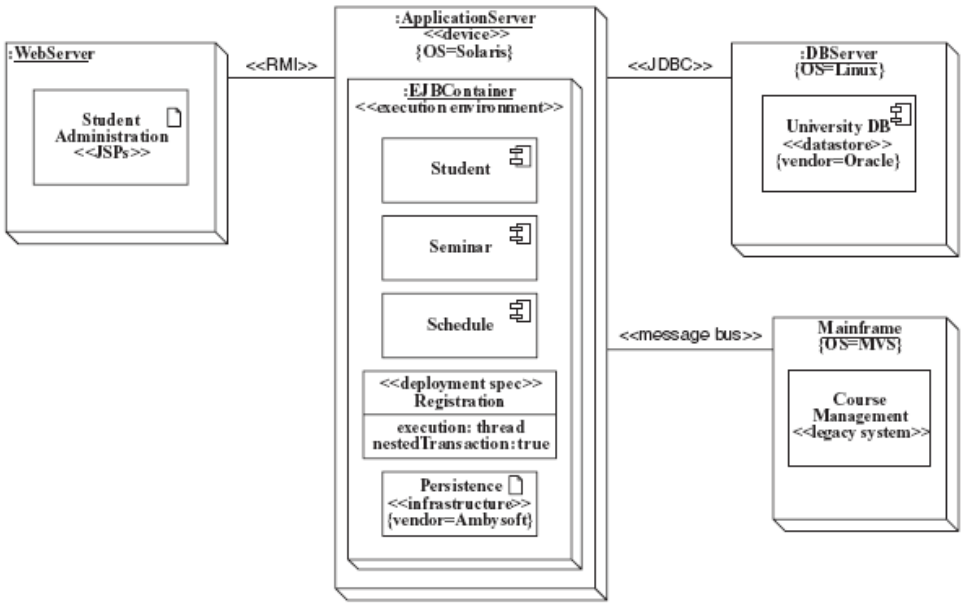


Gambar II.29 *Component Diagram* Pada UML[4]

4. *Deployment Diagram*

   Deployment diagram (Gambar II.30) pada UML melukiskan pandangan statis konfigurasi dari setiap perangkat keras dan perangkat lunak yang berjalan pada sistem tersebut. *Deployment diagram* menunjukkan perangkat keras untuk sistem anda, perangkat lunak yang diinstal pada perangkat keras itu, dan perangkat tengah (*middleware*) yang digunakan untuk menghubungkan mesin-mesin yang berbeda satu sama lain. Diagram ini digunakan untuk:

   a) Mengeksplorasi isu yang terlibat dalam instalasi sistem yang dibuat.
   b) Mengeksplorasi ketergantungan sistem yang dibuat terhadap sistem lain yang saat ini berjalan, atau direncanakan atau terhadap lingkungan sistem.
   c) Menggambarkan konfigurasi penyebaran utama bisnis aplikasi.
   d) Desain konfigurasi hardware dan software dari sebuah sistem.
   e) Menggambarkan topologi fisik infrastruktur / perangkat keras organisasi jaringan.



Gambar II.30 *Deployment Diagram* Pada UML[4]

B. *Behavior Diagrams*

Behavior diagrams digunakan untuk mendeskkripsikan interaksi antara aktor dan sebuah *use case* (bagaimana seorang aktor menggunakan sistem). Diagram-diagram yang termasuk ke dalam *behavior diagram* adalah:

1. *Use Case Diagram*

   Sebuah *use case diagram* pada UML menunjukkan hubungan antara aktor dan *use case* dalam sistem. Diagram ini digunakan untuk:

   a) Memberikan gambaran dari semua atau sebagian dari persyaratan penggunaan untuk sistem atau organisasi dalam bentuk model penting (Konstantin dan Lockwood 1999, Ambler 2004) atau model bisnis (Rasional Coorporation 2002)

   b) Mengkomunikasikan lingkup dari proyek pembangunan.

   c) Model analisis persyaratan penggunaan dalam bentuk sistem *use case model* (Cockburn 2001).

Sebuah model *use case* terdiri dari satu atau lebih diagram dan dokumen pendukung seperti spesifikasi *use case* dan definisi aktor. Dalam model *use case*, spesifikasi *use case* cenderung menjadi artefak primer, dengan *use case* fiagram mengisi peran pendukung sebagai perekat yang membuat kebutuhan model tetap bersama. Terdapat 4 jenis relasi di dalam diagram *use case*, yaitu:

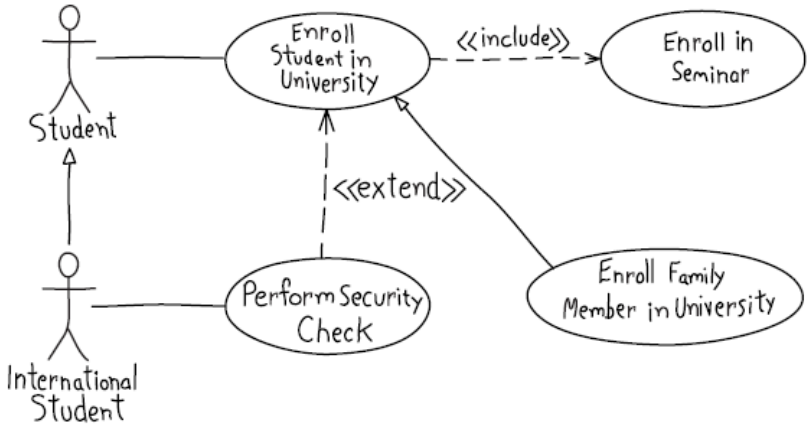
a) Asosiasi antara aktor dan sebuah *use case.*

b) Asosiasi antara 2 *use case.*

c) Generalisasi antara 2 aktor.

d) Generalisasi antara 2 *use case.*

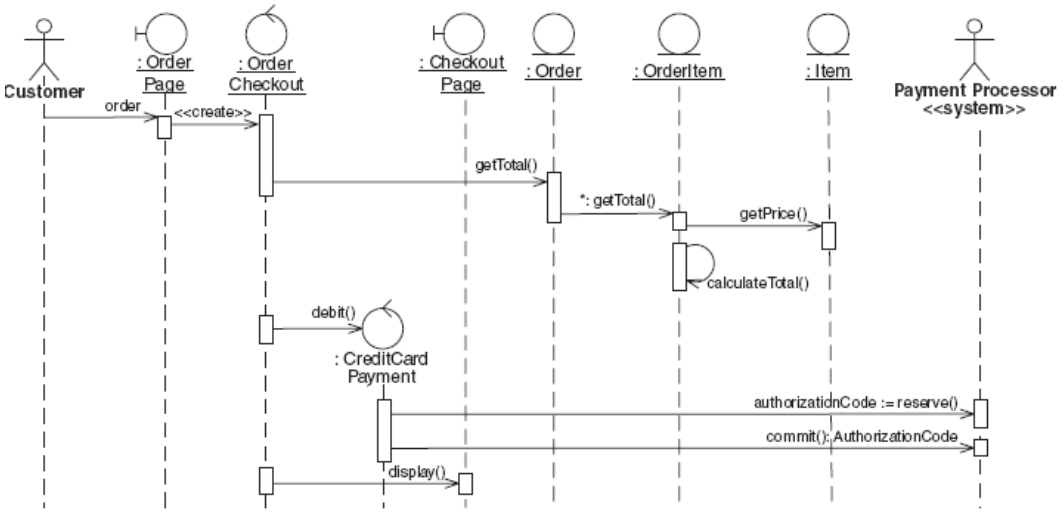
Gambar *use case* diagram dan relasi didalam diagram ini digambarkan pada Gambar II.31.



Gambar II.31 *Use Case Diagram*[4]

2. *Sequence Diagram*

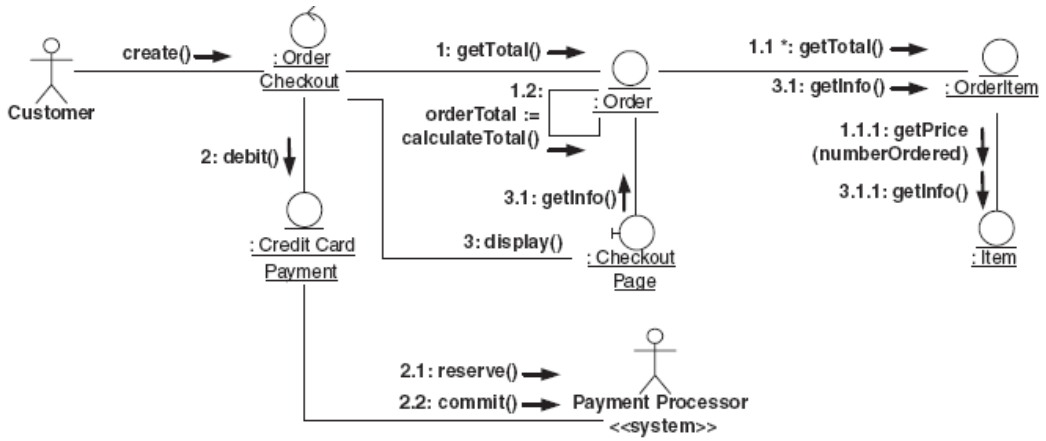
   Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Kegunaannya untuk menunjukkan rangkaian pesan yang dikirim antara objek juga interaksi antara objek, sesuatu yang terjadi pada titik tertentu dalam eksekusi sistem. *Sequence diagram* dapat dilihat pada Gambar II.32.



Gambar II.32 *Sequence Diagram*[4]

3. *Collaboration Diagram*

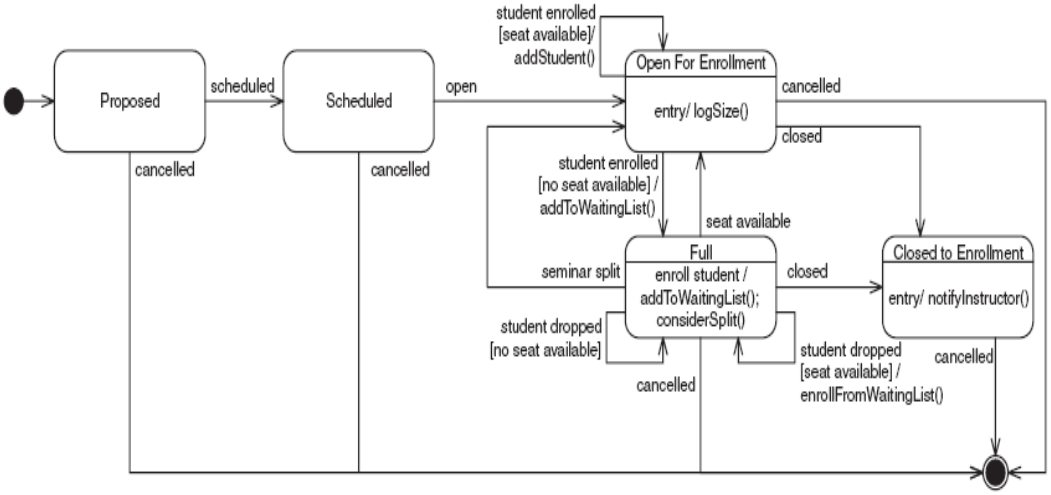
   Menggambarkan kolaborasi dinamis seperti *sequence diagram*. Dalam menunjukkan pertukaran pesan, collaboration diagrams menggambarkan object dan hubungannya (mengacu ke konteks). Jika penekannya pada waktu atau urutan gunakan *sequence diagrams*, tapi jika penekanannya pada konteks gunakan *collaboration diagram*. *Collaboration diagram* dapat dilihat pada Gambar II.33.



Gambar II.33 *Collaboration Diagram*[4]

4. *Statechart Diagram*

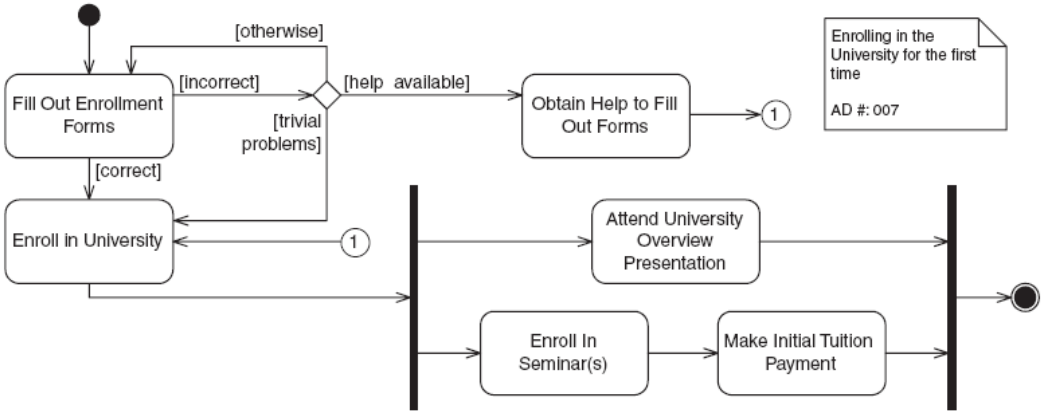
   Menggambarkan semua *state* (kondisi) yang dimiliki oleh suatu objek dari suatu *class* dan keadaan yang menyebabkan *state* berubah. Kejadian dapat berupa object lain yang mengirim pesan. *State class* tidak digambarkan untuk semua *class*, hanya yang mempunyai sejumlah *state* yang terdefinisi dengan baik dan kondisi *class* berubah oleh *state* yang berbeda. *State diagram* dapat dilihat pada Gambar II.34.



Gambar II.34 *Statechart Diagram*[4]

5. *Activity Diagram*

   Menggambarkan rangkaian aliran dari aktivitas, digunakan untuk mendeskripsikan aktifitas yang dibentuk dalam suatu operasi sehingga dapat juga digunakan untuk aktifitas lainnya seperti *use case* atau interaksi. *Activity diagram* dapat dilihat pada Gambar II.35.



Gambar II.35 *Activity Diagram*[4]

2.6 Pembangun Perangkat Lunak

Pembangun perangkat lunak yang digunakan Visual C#, Visual Studio 2010 dan Sql Server 2008.

2.6.1 C# (C Sharp)

C# (dibaca: C sharp) merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka .NET *Framework*. Bahasa pemrograman ini dibuat berbasiskan bahasa C++ yang telah dipengaruhi oleh aspek-aspek ataupun fitur bahasa yang terdapat pada bahasa-bahasa pemrograman lainnya seperti Java, Delphi, Visual Basic, dan lain-lain dengan beberapa penyederhanaan. Menurut standar ECMA-334 C# *Language Specification*, nama C# terdiri atas sebuah huruf Latin C (U+0043) yang diikuti

oleh tanda pagar yang menandakan angka # (U+0023). Tanda pagar # yang digunakan memang bukan tanda kres dalam seni musik (U+266F), dan tanda pagar # (U+0023) tersebut digunakan karena karakter kres dalam seni musik tidak terdapat di dalam keyboard standar. Gambar II.36 merupakan gambar logo C#.[19]



Gambar II.36 Logo C# [21]

2.6.2 Visual Studio 2010

Pengembangan sistem Studio Visual adalah rangkaian lengkap alat yang dirancang untuk membantu perangkat lunak pengembang menciptakan inovatif, aplikasi generasi berikutnya. Ini adalah lingkungan kerja yang sempurna untuk pengembang aplikasi. Microsoft Visual Studio 2010 dan adalah. NET *Framework* 4,0 pengembangan alat dan *platform* generasi berikutnya untuk Windows Vista, Office 2007 System, dan Web. Visual Studio 2010 dan 4.0. NET *Framework* menandai generasi berikutnya alat pengembang dari Microsoft. Dirancang untuk memenuhi kebutuhan terbaru dari pengembang, Visual Studio memberikan inovasi kunci dalam pilar berikut ini: Microsoft Visual Studio memberikan pada visi Microsoft aplikasi klien pintar dengan memungkinkan pengembang untuk secara cepat membuat aplikasi yang terhubung yang memberikan kualitas

tertinggi, pengalaman pengguna yang kaya. Tampilan awal visual studio 2010 dapat dilihat pada Gambar II.37.[23]



Gambar II.37 Visual Studio 2010[3]

2.6.3 SQL *SERVER*

SQL Server pada awalnya merupakan hasil kerja sama antara Sy*base* dengan Microsoft. Dimana pada saat itu masih terjadi kerja sama yang apik antara Microsoft dengan IBM untuk membuat sebuah Operating system baru yaitu OS/2. Keduanya sepakat menginginkan sebuah *database* baru yang juga bisa dijalankan pada OS/2 sehingga mereka bermitra dengan Sy*base*, dan meminta Sy*base* untuk menyertakan produk DataServer yang berbasis UNIX ke OS/2. Selain dengan Sy*base*, Microsoft juga menggandeng Ashton-Tate yang memiliki mayoritas pasar *database* dengan produk unggulan mereka yang terkenal yaitu d*Base*, kerja sama ini dirasa Microsoft sangat penting untuk mendapatkan pengakuan dan pangsa pasar bagi OS/2.

Sebuah versi Beta dari Ashton-Tate dan Microsoft SQL Server yang dijalankan pada OS/2 dirilis pada tahun 1988. Seiring dengan penghentian kerja sama Microsoft dengan Ashton-Tate, maka pada tahun 1989 bekerja sama dengan teknologi yang di usung oleh Sy*base* diluncurkan Microsoft SQL Server versi 1.0.

Microsoft SQL Server 1.1 dirilis pada tahun 1990 dan berjalan pada platform OS/2 dengan keunggulan untuk memperbaiki bug pada versi 1.0. Selain itu, versi ini juga mempunyai fitur yang mampu mendukung sebuah platform client baru yaitu Windows 3.0. Dengan tambahan utilitas, pustaka pemrograman dan tool-tool Pengujiistrasi menjadikan Microsoft SQL Server merupakan seri *database* terbaik pada masanya. Microsoft SQL Server versi 1.11 perawatan dirilis pada 1991 seiring penghentian kerjasama dengan IBM untuk pengembangan OS/2. Penghentian ini didasari untuk melakukan pengembangan pada *Operating System multi user* Windows NT.

Tool Pengujiistrasi *database* GUI berbasis-Windows terdapat pada Microsoft SQL Server versi 4.2 yang dirilis pada awal tahun 1992. Ditandai dengan makin harmonisnya kerjasama Microsoft dengan Sy*base* untuk melakukan pengembangan pada versi SQL Server Microsoft yang dijalankan pada OS/2 yang diselaraskan dengan versi Sy*base* dan berjalan pada UNIX. Kehadiran Windows NT menyebabkan Microsoft merilis Microsoft SQL Server yang berjalan pada *platform* tersebut. Versi ini mendukung arsitektur 32-bit dan sepenuhnya ditulis-ulang dari dasar. Pada tahun itu juga Microsoft memutuskan untuk menghentikan pengembangan SQL Server pada OS/2 dan memfokuskan untuk versi Windows NT saja.

Pada tahun 1994 kerjasama Microsoft dan Sy*base* berakhir, selanjutnya Microsoft bertanggung jawab sepenuhnya pada pengembangan kode SQL server. Seiring dengan penghentian tersebut, para ahli di Microsoft terus melakukan pengembangan hingga akhirnya meluncurkan versi Microsoft SQL Server 6.0

pada tahun 1995. Versi 6.0 menghadirkan fitur yang sangat diperlukan seperti replikasi, kursor *scrollbar* dan *tool management* yang memasukkan edisi pertama *Enterprise Manager*. Kehadirian versi 6.0 menjadikannya sebagai kompetitor bagi Sy*base*, Oracle dan Informix.

Keseriusan Microsoft terlihat saat awal tahun 1996 kembali menghadirkan versi 6.5 Microsoft SQL Server yang mampu meningkatkan penyimpanan data dan tunduk pada standar ANSI SQL. Microsoft SQL Server versi 7.0 dirilis pada tahun 1999. Mesin *database* baru yang telah dirancang bangun, prosesor *query* dan penguncian tingkat-baris penuh dihadirkan pada versi ini, disamping utilitas Pengujiistrasi baru dan wizard yang baru pula. Kehadiran versi ini seakan menjawab kebutuhan desktop sampai internet hingga *big company* yang membutuhkan skalabilitas. Versi ini juga merupakan *platform database* untuk skala-enterprise dan aplikasi penyimpanan data dengan teknologi seperti OLAP.

SQL Server 2000 dirilis pada tahun 2000 dengan meningkatkan skalabilitas perangkat-keras juga mendukung piranti *handheld* Windows CE dan *server cluster multiprosesor* delapan jalur. Dukungan terhadap *built-in* XML, dukungan failover 4-node, *update*, penghapusan *cascading*, pencarian berbasis teks dan kemampuan menjalankan multi bentuk pada sebuah server tunggal ditanamkan pada versi ini.

SQL Server juga terdiri dari tiga versi yaitu edisi Small Bussiness Server (SBS), edisi Standard dan edisi Enterprise. Semua edisi berisi mesin dan *tool database* inti. Perbedaan utama dari ketiga edisi ini adalah sistem operasi yang dijalankan

masing-masing. Setiap edisi memiliki sebuah instalasi standar dan instalasi desktop. Instalasi standar adalah sebuah server *database* penuh dan berjalan hanya pada Windows NT Server. Sementara instalasi desktop hanya digunakan dan dirancang untuk kebutuhan penyimpanan *database* lokal saja. SQL Server versi instalasi desktop mampu berjalan pada Windows 95, Windows 98, Windows NT Workstation, Windows NT Server atau Windows NT Enterprise. Microsoft Data *Engine* (MSDE) merupakan sebuah *database* final yang juga versi skala bawah dari SQL Server yang disertakan dalam Microsoft Office dan Visual Studio. Dengan kemampuan ukuran 2GB, MSDE hanya disarankan untuk digunakan pada aplikasi dengan pengguna tidak lebih dari 5 pengguna pada saat bersamaan. MSDE juga bisa digunakan bersama dengan *Access* atau aplikasi client Visual Basic. Namun ada keuntungan terbesar dari penggunaan MSDE dibandingkan dengan mesin Jet, yakni MSDE sepenuhnya kompatibel dengan SQL Server, sehingga pada saat kebutuhan meningkat, maka dengan mudah MSDE dapat di upgrade menjadi SQL Server. Hal ini dikarenakan MSDE mempunyai jalur peng-upgrade-an dengan SQL Server yang tidak dimiliki oleh mesin Jet. Logo sql server 2008 dapat dilihat pada Gambar II.38.[14]



Gambar II.38 SQL Server 2008[9]