BAB II

LANDASAN TEORI

II.1. Sistem

Sistem merupakan sekumpulan elemen-elemen yang saling terintegrasi serta melaksanakan fungsinya masing-masing untuk mencapai tujuan yang telah ditetapkan. Karakteristik sistem terdiri dari :

1. Komponen Sistem

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

2. Batasan Sistem

   Batasan merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang suatu kesatuan. Batasan suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem

   Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkung luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem

   Penghubung merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya

5. Masukan Sistem

   Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintance input*) dan masukan sinyal (*signal input*).

6. Keluaran Sistem

   Keluaran sistem adalah hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

7. Pengolah Sistem

   Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem

   Suatu sistem mempunyai tujuan (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya (Sulindawati ; 2010 : 135).

II.2. Kecerdasan Buatan (*Artificial Intelegent*)

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.

Manusia cerdas (pandai) dalam menyelesaikan permasalahan karena manusia mempunyai pengetahuan dan pengalaman. Pengetahuan diperoleh dari belajar. Semakin banyak bekal pengetahuan yang dimiliki tentu akan lebih mampu menyelesaikan permasalahan. Tapi bekal pengetahuan saja tidak cukup, manusia juga diberi akal untuk melakukan penalaran,mengambil kesimpulan berdasarkan pengetahuan dan pengalaman yang dimiliki. Tanpa memiliki kemampuan untuk menalar dengan baik, manusia dengan segudang pengalaman dan pengetahuan tidak akan dapat menyelesaikan masalah dengan baik. Demikian juga dengan kemampuan menalar yang sangat baik,namun tanpa bekal pengetahuan dan pengalaman yang memadai,manusia juga tidak akan bisa menyelesaikan masalah dengan baik. Agar mesin bisa cerdas (bertindak seperti dan sebaik manusia) maka harus diberi bekal pengetahuan dan mempunyai kemampuan untuk menala. Kelebihan kecerdasan buatan :

1. Lebih bersifat permanen. Kecerdasan alami bisa berubah karena sifat manusia pelupa. Kecerdasan buatan tidak berubah selama sistem computer dan program tidak mengubahnya.
2. Lebih mudah diduplikasi dan disebarkan. Mentransfer pengetahuan manusia dari 1 orang ke orang lain membutuhkan proses yang sangat lama & 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.

3. Lebih murah. Menyediakan layanan komputer akan lebih mudah & murah dibandingkan mendatangkan seseorang untuk mengerjakan sejumlah pekerjaan dalam jangka waktu yang sangat lama.
4. Bersifat konsisten karena kecerdasan buatan adalah bagian dari teknologi komputer sedangkan kecerdasan alami senantiasa berubah-ubah
5. Dapat didokumentasi. Keputusan yang dibuat komputer dapat didokumentasi dengan mudah dengan cara melacak setiap aktivitas dari sistem tersebut. Kecerdasan alami sangat sulit untuk direproduksi.
6. Lebih cepat
7. Lebih baik (Idhawati Hestiningsih ; 2012 : 1).

II.3. Jaringan Syaraf Tiruan

Jaringan Syaraf Tiruan adalah model komputasi dari otak manusia. JST merupakan suatu sistem pemrosesan informasi yang mempunyai karakteristik baik struktur dan cara kerja menyerupai jaringan syaraf manusia. Hal ini dikarenakan manusia memiliki banyak keunggulan dibandingkan makhluk lain, otak manusia memiliki struktur yang sangat komplek dan mampu berpikir yang dapat memecahkan persoalan yang dihadapinya dan mampu belajar dari pengalaman masa lalu. Jarigan syaraf biologis pada otak manusia terdiri dari sel-sel syaraf yang disebut *neuron* yang saling berhubungan satu dengan yang lain, pada suatu penghubung yang disebut sinapsis.

Tiga komponen penting pada sel syaraf biologis yang digunakan untuk memahami JST :

(1) Dendrit, merupakan elemen pemrosesan yang menerima dan melewatkan sinyal masukan dari neuron lain. Sebuah neuron mampu menerima 5.000 sampai 15.000 sinyal masukan. Sinyal tersebut dimodifikasi dengan bobot (diperkuat/diperlemah) pada sinapsis penerima,

(2) Soma/badan sel, berfungsi mengakumulasi sinyal masukan terbobot yang dilewatkan melalui dendrit. Jika sinyal – sinyal tersebut lebih besar dari batas ambang tertentu (*threshold*), maka sel akan dipicu sehingga akan mentransmisikan ke neuron lain,

(3) Akson, berfungsi sebagai saluran keluaran dari suatu neuron yang akan menyalurkan sinyal ke neuron yang lain (Didi Supriyadi ; 2013 : 6).

II.3.1. Tahapan Jaringan Syaraf Tiruan

Berikut tahapan-tahapan yang akan dilakukan dengan fungsi aktivasi sigmoid. Tahapan yang harus dilakukan adalah sebagai berikut :

1. Inisialisasi (*initialization*), merupakan tahap di mana variabel-variabel nilai akan diset atau didefinisikan terlebih dahulu, misalnya seperti : nilai data input, weight, nilai output yang diharapkan, learning rate dan nilai-nilai data lainnya.
2. Aktivasi (*activation*),merupakan proses perhitungan terhadap nilai aktual *output* pada *hidden layer* dan menghitung nilai actual output pada output layer.

3. *Weight Training*, merupakan proses perhitungan nilai error gradient pada o*utput layer* dan menghitung nilai *error gradient* pada *hidden layer*. (Zekson Arizona Matondang ; 2013 : 91)

II.3.2. Inspirasi Biologi

Jaringan Syaraf Tiruan keluar dari penelitian kecerdasan buatan, terutama percobaan untuk menirukan *fault-tolerence* dan kemampuan untuk belajar dari sistem syaraf biologi dengan model struktur *low-level* dari otak. Otak terdiri dari sekitar (10.000.000.000) sel syaraf yang saling berhubungan. Sel syaraf mempunyai cabang struktur input (dendrites), sebuah inti sel dan percabangan struktur output (axon). Axon dari sebuah sel terhubung dengan dendrites yang lain melalui sebuah synapse. Ketika sebuah sel syaraf aktif, kemudian menimbulkan suatu signal *electrochemical* pada *axon*. Signal ini melewati *synapses* menuju ke sel syaraf yang lain. Sebuah sel syaraf lain akan mendapatkan signal jika memenuhi batasan tertentu yang sering disebut dengan nilai ambang atau (*threshold*) (Eli Yani ; 2015 : 1).

II.3.3. Perbandingan Jaringan Syaraf Tiruan dengan Konvensial

Jaringan Syaraf Tiruan memiliki pendekatan yang berbeda untuk memecahkan masalah bila dibandingkan dengan sebuah komputer konvensional. Umumnya komputer konvensional menggunakan pendekatan algoritma (komputer konvensional menjalankan sekumpulan perintah untuk memecahkan masalah). Jika suatu perintah tidak diketahui oleh komputer konvensional maka komputer

konvensional tidak dapat memecahkan masalah yang ada. Sangat penting mengetahui bagaimana memecahkan suatu masalah pada komputer konvensional dimana komputer konvensional akan sangat bermanfaat jika dapat melakukan sesuatu dimana pengguna belum mengatahui bagaimana melakukannya.

Jaringan Syaraf Tiruan dan suatu algoritma komputer konvensional tidak saling bersaing namun saling melengkapi satu sama lain. Pada suatu kegiatan yang besar, sistim yang diperlukan biasanya menggunakan kombinasi antara keduanya (biasanya sebuah komputer konvensional digunakan untuk mengontrol Jaringan Syaraf Tiruan untuk menghasilkan efisiensi yang maksimal. Jaringan Syaraf Tiruan tidak memberikan suatu keajiban tetapi jika digunakan secara tepat akan menghasilkan sasuatu hasil yang luar biasa (Eli yani ; 2015 : 2).

II.4. Metode *Backpropagation*

Penggunaan dan penerapan metode JST backpropagation tergolong algoritma pembelajaran/pelatihan yang bersifat supervised dan menggunakan aturan pembelajaran pengoreksian error. Proses pelatihan algoritma/metode *backpropagation* didasarkan pada hubungan yang sederhana, yaitu jika keluaran memberikan hasil yang salah, maka penimbang (*weight*) dikoreksi supaya galatnya dapat diperkecil dan tanggapan JST.

JST Backpropagation memiliki lapisan terdiri dari lapisan input (1 buah), lapisan tersembunyi, dan lapisan output (1 buah). Lapisan input terdiri dari neuron-neuron atau unit-unit input, mulai dari 1 sampai unit input n. Lapisan tersembunyi (minimal 1). Lapisan tersembunyi terdiri dari unit-unit tersembunyi

mulai dari unit tersembunyi 1 sampai unit tersembunyi p. Lapisan output (1 buah). Lapisan output terdiri dari unit-unit output mulai dari unit output 1 sampai unit output m. Nilai n, p, m masing-masing merupakan bilangan integer sembarang yang digunakan metode Backpropagation (Sandy Kosasi ; 2014 : 21).

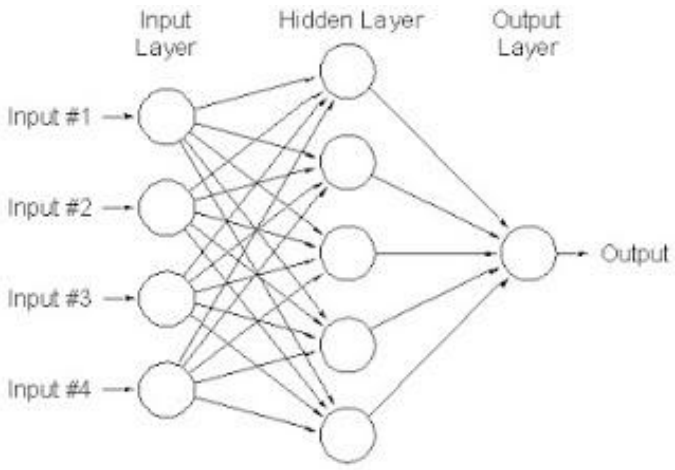
Berikut adalah ilustrasi perhitungan dari metode backpropagation :

contoh soal:

1. Di dalam algoritma backpropagation, terdapat 1 layer input yang berisi 4 node, 1 hidden layer yang berisi 4 node dan 1 layer output yang berisi 1 node.
   a. tentukan gambarnya dan
   b. berikan notasinya juga
   c. aktifkan matriks tersebut (random)

jawaban:

a.



Gambar II.1 Layer

Sumber : Sandy Kosasi ; 2014 : 21

b. kita masukkan dalam notasi sebagai berikut [u v], dimana u adalah input layer x output layer = 4 x 5 dan v adalah 5 x 1...

maka kita masukkan menjadi:

U11 U12 U13 U14 U15 --> INPUT 1
U21 U22 U23 U24 U25 --> INPUT 2
U31 U32 U33 U34 U35 --> INPUT 3
U41 U42 U43 U44 U45 --> INPUT 4
V11 --> INPUT 1 (HIDDEN LAYER KE OUTPUT LAYER)
V21 --> INPUT 2 (HIDDEN LAYER KE OUTPUT LAYER)
V31 --> INPUT 3 (HIDDEN LAYER KE OUTPUT LAYER)
V41 --> INPUT 4 (HIDDEN LAYER KE OUTPUT LAYER)
V51 --> INPUT 5 (HIDDEN LAYER KE OUTPUT LAYER)

c. karena ini adalah random, maka kita bisa membuat semau angka kita, tetapi karena penulis ingin mencontohkan jadi penulis buat batas $-0.5 \leq x \leq 0.5$ menjadi :

U =

0.2	0.3	-0.2	-0.4	0.3
0.1	-0.1	-0.5	0.5	0.2
0.1	-0.4	-0.5	0.3	0.1
0.1	0.3	-0.4	-0.1	0.4

V =

0.1
-0.2
-0.35
-0.25
0.5

II.4.1. Pelatihan Algoritma Backpropagation

Backpropagasi memiliki metode pembelajaran supervised learning dan lapisan masukan akan menerima pola masukan dan melakukan proses komputasi

berdasarkan bobot awal yang diperoleh secara acak (random). Jika keluaran dari jaringan berbeda dengan target yang diharapkan maka jaringan melakukan penyesuaian terhadap bobot yang ada. Proses itu akan terus berlanjut hingga keluaran dari jaringan dan target yang diharapkan menjadi sama. Proses pembelajaran membutuhkan waktu yang lama hingga mencapai nilai tersebut. Oleh karena itu, proses learning dibatasi dan akan berhenti jika perbedaan antara output dan target sudah mencapai nilai yang lebih kecil dari nilai toleransi (error rate). Besarnya penyesuaian bobot pada setiap siklus pembelajaran ditentukan oleh parameter yang disebut learning rate. Setelah tahap pembelajaran, jaringan saraf tiruan siap untuk memasuki tahap recalling/ searching yang merupakan proses ketika jaringan saraf tiruan menerima masukan dari dunia luar melalui lapisan masukan dan melalui komputasi pda masing-masing neuron yang terdapat di dalam lapisan akan dihasilkan keluaran pada lapisan keluaran Analogi pada otak manusia, seperti menerima masukan berupa gambar buah kemudian otak manusia akan melakukan komputasi sehingga mengenali nama buah tersebut. Pada pelatihan backpropagation terdapat dua fase, yaitu proses propagasi nilai aktivasi atau masukan dan proses penyesuaian dengan keluaran yang diharapkan.

Proses propagasi nilai aktivasi tersebut adalah proses perubahan nilai bobot koneksi antar-neuron yang menghubungkan lapisan jaringan, baik itu antara lapisan masukan dengan lapisan tersembunyi, lapisan tersembunyi yang satu dengan yang lainnya, maupun bobot koneksi lapisan tersembunyi dengan lapisan keluaran. Nilai neuron dari setiap keluaran merupakan hasil dari fungsi aktivasi. Fungsi itu biasanya digunakan untuk menurunkan nilai aktivasi dan mengubahnya

menjadi suatu nilai keluaran yang berarti. Kadang-kadang fungsi itu juga digunakan untuk menambahkan nilai bias. Fungsi sigmoid merupakan fungsi aktivasi yang digunakan dalam penelitian ini (Diaz D. Santika ; 2011 : 33).

II.4.2. Layer Backpropagation

Secara umum, arsitektur jaringan saraf tiruan dibedakan menjadi dua, yakni jaringan saraf singlelayer dan jaringan saraf multi-layer.

1. Jaringan saraf single-layer
    a. Neuron-neuron dikelompokkan menjadi dua bagian, yakni unit-unit input dan unit-unit output.
    b. Unit-unit input menerima masukan dari luar, unitunit output mengeluarkan respon dari jaringan sesuai dengan masukannya.
2. Jaringan saraf multi-layer
    a. Neuron-neuron dikelompokkan menjadi tiga bagian, yakni unit-unit input, unit-unit hidden, dan unit-unit output.
    b. Jumlah unit-unit hidden tergantung pada kebutuhan. Semakin kompleks jaringan, unit-unit hidden yang dibutuhkan semakin banyak, demikian pula jumlah layer-nya (Sari Indah Anatta Setiawan ; 2011 : 24).

II.5. Pengertian Visual Basic

Microsoft Visual Studio.net merupakan salah satu *software* buatan *Microsoft Corp.* yang didesain khusus dalam pembuatan program-program profesional berbasis *windows platform*. *Microsoft Visual Studio .net* merupakan perangkat lunak yang terintegrasi, di dalamnya terdapat beberapa paket *software*

yang dapat digunakan oleh programer dalam membangun sebuah program profesional, diantaranya adalah *Visual Basic, Visual J#, Visual C, #Visual C++* dan *Java Runtime* yang sama-sama berada dalam naungan *platform Microsoft.NET Framework*. Bagian – bagian dari *software* ini diantaranya *toolbox*, jendela *properties, server explorer* dan *solution explorer*. *Toolbox* digunakan untuk pemilihan kontrol–kontrol yang akan digunakan pada program yang akan dirancang. Kontrol ini merupakan kontrol standar yang digunakan oleh aplikasi *Windows*, dan kontrol–kontrol tambahan yang disebut *ActiveX*. Kontrol yang ada pada jendela ini dapat ditambah dan dikurangi sesuai kebutuhan. Jendela *Properties* merupakan jendela yang digunakan untuk mengatur properti sebuah objek. Jendela *Properties* ini terbagi dalam dua bagian yaitu *Alphabetic* dan *Catagirozed*. Perbedaan dari keduanya hanyalah cara menampilkan *properties* dalam sebuah objek. Pada bagian *Alphabetic*, properti diatur berdasarkan urutan abjad, sedangkan di bagian *Catagorized*, properti diatur dalam kelompok-kelompok kategori (Fajar Rahadian ; 2011 : 3).

II.6. *SQL Server*

SQL Server 2008 adalah sebuah terobosan baru dari Microsoft dalam bidang *database*. *SQL Server* adalah DBMS (*Database Management System*) yang dibuat oleh Microsoft untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan *Oracle*. *SQl Server* 2008 dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server* 2008 membawa beberapa

terobosan dalam bidang pengolahan dan penyimpanan data. *Microsoft* merilis *SQL Server* 2008 dalam beberapa versi yang disesuaikan dengan segment-segment pasar yang dituju. Versi-versi tersebut adalah sebagai berikut. Menurut cara pemrosesan data pada prosesor maka *Microsoft* mengelompokkan produk ini berdasarkan 2 jenis yaitu :

1. Versi 32-bit(x86), yang biasanya digunakan untuk komputer dengan single prosesor (Pentium 4) atau lebih tepatnya prosesor 32 bit dan sistem operasi *Windows* XP.
2. Versi 64-bit(x64), yang biasanya digunakan untuk komputer dengan lebih dari satu prosesor (Misalnya Core 2 Duo) dan *system* operasi 64 bit seperti *Windows* XP 64, Vista, dan *Windows* 7.

Sedangkan secara keseluruhan terdapat versi-versi seperti berikut ini:

1. Versi *Compact*, ini adalah versi "Tipis" dari semua versi yang ada. Versi ini seperti versi *desktop* pada *SQL Server 2000*. Versi ini juga digunakan pada *handled drvice* seperti *Pocket PC*, PDA, *SmartPhone*, *Tablet PC.*
2. Versi *Express*, ini adalah versi "Ringan" dari semua versi yang ada (tetapi versi ini berbeda dengan versi *compact*) dan paling cocok untuk latihan para pengembang aplikasi. Versi ini memuat *Express Manager standar*, integrasi dengan CLR dan XML (Wenny Widya ; 2012 : 3).

II.7. Teknik Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi

adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel rasional. Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang tekah ditemukan.

II.7.1. Bentuk-bentuk Normalisasi

1. Bentuk normal tahap pertama (1" Normal Form)

Contoh yang kita gunakan di sini adalah sebuah perusahaan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri. Contoh normalisasi 1NF dapat dilihat pada tabel II.1.

Tabel II.1. Contoh Tabel Bentuk Normal Pertama (1NF)

p#	Status	Kota	b#	Qty
p1	20	Yogyakarta	b1	300
p1	20	Yogyakarta	b2	200
p1	20	Yogyakarta	b3	400
p1	20	Yogyakarta	b4	200
p1	20	Yogyakarta	b5	100
p1	20	Yogyakarta	b6	100
p2	10	Medan	b1	300
p2	10	Medan	b2	400
p3	10	Medan	b2	200
p4	20	Yogyakarta	b2	200
p4	20	Yogyakarta	b4	300
p4	20	Yogyakarta	b5	400

(Janner Simarmata ; 2010 : 78)

2. Bentuk normal tahap kedua (2nd normal form)

Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada seluruh kolom yang membentuk kunci utama yang dapat dilihat pada table II.2.

Tabel II.2. Contoh Tabel Bentuk Normal Kedua (2NF)

Pemasok2

p#	Status	Kota
P1	20	Yogyakarta
P2	10	Medan
P3	10	Medan
P4	20	Yogyakarta
P5	30	Bandung

Barang

p#	b#	Qty
p1	b1	300
p1	b2	200
p1	b3	400
p1	b4	200
p1	b5	100
p1	b6	100
p2	b1	300
p2	b2	400
p3	b2	200
p4	b2	200
p4	b4	300
p4	b5	400

(Janner Simarmata ; 2010 : 78)

3. Bentuk normal tahap ketiga (3rd normal form)

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada pada 2NF dan setiap kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya yang dapat dilihat pada tabel II.3.

Tabel II.3. Contoh Tabel Bentuk Normal Ketiga (3NF)

Pemasok Kota

p#	Kota
P1	Yogyakarta
P2	Medan
P3	Medan
P4	Yogyakarta
P5	Bandung

Kota Status

Kota	status
Yogyakarta	20
Medan	10
Yogyakarta	20
Bandung	30

(Janner Simarmata ; 2010 : 78)

4. *Boyce Code Normal Form* (BCNF)

Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3NF sudah cukup karena sangat jarang entitas yang berada pada 3NF bukan merupakan 4NF dan 5NF.

5. Bentuk Normal Keempat (4NF)

Sebuah tabel rasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD). Sebuah ketergantungan multivalue tiga kolom, satu kolom mempunyai banyak baris bernilai sama, tetapi kolom lain bernilai berbeda yang dapat dilihat pada tabel II.4.

Tabel II.4. Contoh Tabel Bentuk Normal Keempat (4NF)

Pegawai Proyek

peg#	Pry#
1211	P1
1211	P3

Pegawai Ahli

Peg#	Ahli
1211	Analisis
1211	Perancangan
1211	Pemrograman

(Janner Simarmata ; 2010 : 78)

6. Bentuk Normal Kelima

Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*) yang dapat dilihat pada table II.5 (Janner Simarmata ; 2010 : 78).

Tabel II.5. Contoh Tabel Bentuk Normal Kelima (5NF)

peg#	Pry#	Ahli
1211	11	Perancangan
1211	28	Pemrograman

(Janner Simarmata ; 2010 : 78)

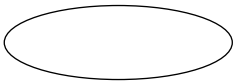
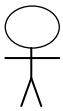
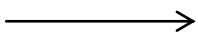
II.8. UML (*Unified Modeling Language*)

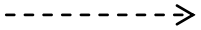
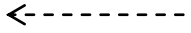
Menurut Windu Gata (2013 : 4), Hasil pemodelan pada OOAD (*Object Oriented Analysis & Design*) terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use case* Diagram

   Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram yang dapat dilihat pada tabel II.6.

Tabel II.6. Simbol *Use Case*

Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.

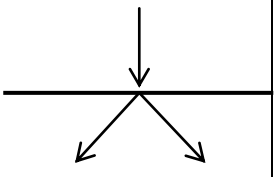
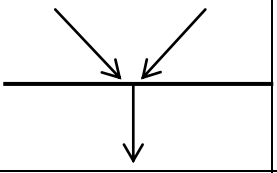
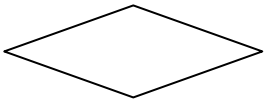
	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Windu Gata ; 2013 : 4)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram,* yang dapat dilihat pada tabel II.7.

Tabel II.7. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true, false.*

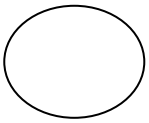
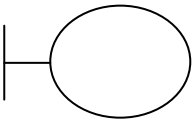
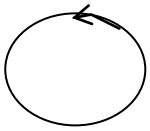
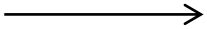
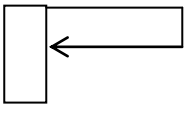
New Swimline	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata ; 2013 : 6)

3. Diagram Urutan (*Sequence Diagram*)

   Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* yang dapat dilihat pada tabel II.8.

Tabel II.8. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class*, merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class*, berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class*, suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation*, *activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.

	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Windu Gata ; 2013 : 7)

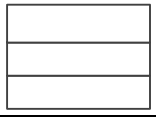
4. *Class Diagram* (Diagram Kelas)

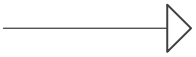
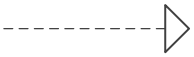
Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan.

Class diagram secara khas merupakan Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal pada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti yang dapat dilihat pada tabel II.9.

Tabel II.9. Simbol *Class Diagram*

Gambar	Keterangan
	*Class,* Menambahkan kelas baru pada diagram
	*Interface*, Menambahkan kelas antarmuka *(interface)* pada diagram

	Association, Menggambar relasi asosiasi
	Association class, Menghubungkan kelas asosiasi *(association class)* pada suatu relasi asosiasi
	*Generalization*, Menggambarkan relasi generalisasi
	*Realize*, Menggambarkan relasi realisasai
	Aggregation, Menggambarkan relasi agregasi

(Sumber : Windu Gata ; 2013 : 9)