

BAB II

LANDASAN TEORI

II.1. Sistem

Sistem adalah suatu jaringan kerja dari prosedur - prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan kegiatan atau untuk melakukan sasaran yang tertentu. Pendekatan sistem yang merupakan jaringan kerja dari prosedur lebih menekankan urutan-urutan operasi di dalam sistem. 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 (*boundry*)

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 (*environment*)

Lingkungan luar sistem (*environment*) adalah diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkungan luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem (*interface*)

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

5. Masukan Sistem (*input*)

Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintenanceinput*) dan masukan sinyal (*signalinput*).

6. Keluaran Sistem (*output*)

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 (Jeperson Hutahaeon ; 2014 : 2).

II.2. Sistem Pakar

Sistem pakar adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer yang dirancang untuk memodelkan kemampuan menyelesaikan masalah seperti layaknya seorang pakar. Tujuan utama sistem pakar bukan untuk menggantikan kedudukan seorang ahli maupun pakar, tetapi untuk memasyarakatkan pengetahuan dan pengalaman pakar-pakar yang ahli di bidangnya. sistem pakar disusun oleh dua bagian utama, yaitu :

1. Lingkungan pengembangan (*development environment*), digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar.
2. Lingkungan konsultasi (*consultation environment*), digunakan oleh pengguna yang bukan pakar guna memperoleh pengetahuan pakar (Andri Saputra ; 2011 : 204).

II.2.1. Komponen Sistem Pakar

Komponen - komponen sistem pakar adalah seperti di bawah ini :

1. Antarmuka (*User Interface*)

User Interface merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Antarmuka menerima informasi dari pemakai dan mengubahnya kedalam bentuk yang dapat diterima oleh sistem.

2. Basis Pengetahuan

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar ini disusun atas dua elemen dasar, yaitu fakta dan aturan.

3. Akuisisi Pengetahuan (*Knowledge Acquisition*)

Akuisisi pengetahuan adalah akumulasi, transfer dan transformasi keahlian dalam menyelesaikan masalah dari sumber pengetahuan kedalam program komputer.

4. Mesin *Inferensi*

Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah.

5. *Workplace*

Workplace merupakan area dari sekumpulan memori kerja (*working memory*). *Workplace* digunakan untuk merekam hasil-hasil antara dan kesimpulan yang dicapai.

6. Fasilitas Penjelasan

Fasilitas penjelasan adalah komponen tambahan yang akan meningkatkan kemampuan sistem pakar.

7. Perbaikan Pengetahuan

Pakar memiliki kemampuan untuk menganalisis dan meningkatkan kinerjanya serta kemampuan untuk belajar dari kinerjanya (Andri Saputra ; 2011 : 204).

II.3. Kondisi Lahan

Istilah lahan dalam arti *land* adalah serangkaian atribut permukaan bumi yang penting bagi kehidupan manusia. Unsur utamanya berupa tanah (*soil*), sedangkan unsur pelengkapinya meliputi apa saja yang ada di atasnya (air, udara dan tumbuhan) dan apa yang ada di bawahnya (batuan induk). Dengan demikian,

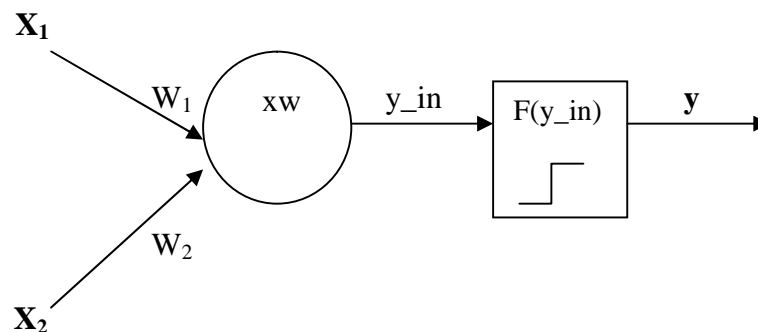
lahan pertanian meliputi tanah (pertanian), air (air irigasi dan air hujan), udara (cuaca dan iklim), tumbuhan (tumbuhan yang dibudidayakan dan tidak dibudidayakan) dan batuan induk (Fitriana Susanti ; 2013 : 322).

II.4. Metode *Hebb Rule*

Hebb rule adalah aturan pelatihan yang paling awal dan paling sederhana untuk jaringan syaraf tiruan secara umum. Pada aturan *hebb rule* ini pelatihan yang terjadi yaitu dengan memodifikasi kekuatan sinapsis (bobot). Jika kedua syaraf kedua-duanya “*on*” pada waktu yang sama, maka bobot neuron akan bertambah.

Kita akan mengarah pada jaringan syaraf single layer (*feedforward*) dilatih menggunakan (perluasan *hebb rule* sebagai jaringan *hebb rule*. Aturan *hebb rule* juga digunakan untuk pelatihan jaringan lain yang dibahas kemudian. Karena kita sedang mempertimbangkan jaringan *single - layer*, salah satu neuron yang saling berhubungan merupakan satu unit masukan dan satu unit keluaran (karena tidak ada unit input yang terhubung satu sama lain, tidak juga banyak unit *output* terhubung).

Jika data adalah biner, formula ini tidak membedakan antara pasangan pelatihan di mana unit input adalah “*on*” dan nilai target adalah “*off*” dan pasangan pelatihan yang mana antara unit input dan nilai target adalah “*off*” (Yun Eninggar ; 2012 : 2).



Gambar II.1. Arsitektur Jaringan Pada Hebb Rule Dua Input

(Sumber : Yun Eninnggar ; 2012 ; 2)

Algoritma Hebb Rule untuk memperbaiki bobot ke-i (untuk setiap pola)

adalah $w(\text{baru}) = w(\text{lama}) + (t-y) * X_i$;

X_i = Vektor Input

y = Output Jaringan

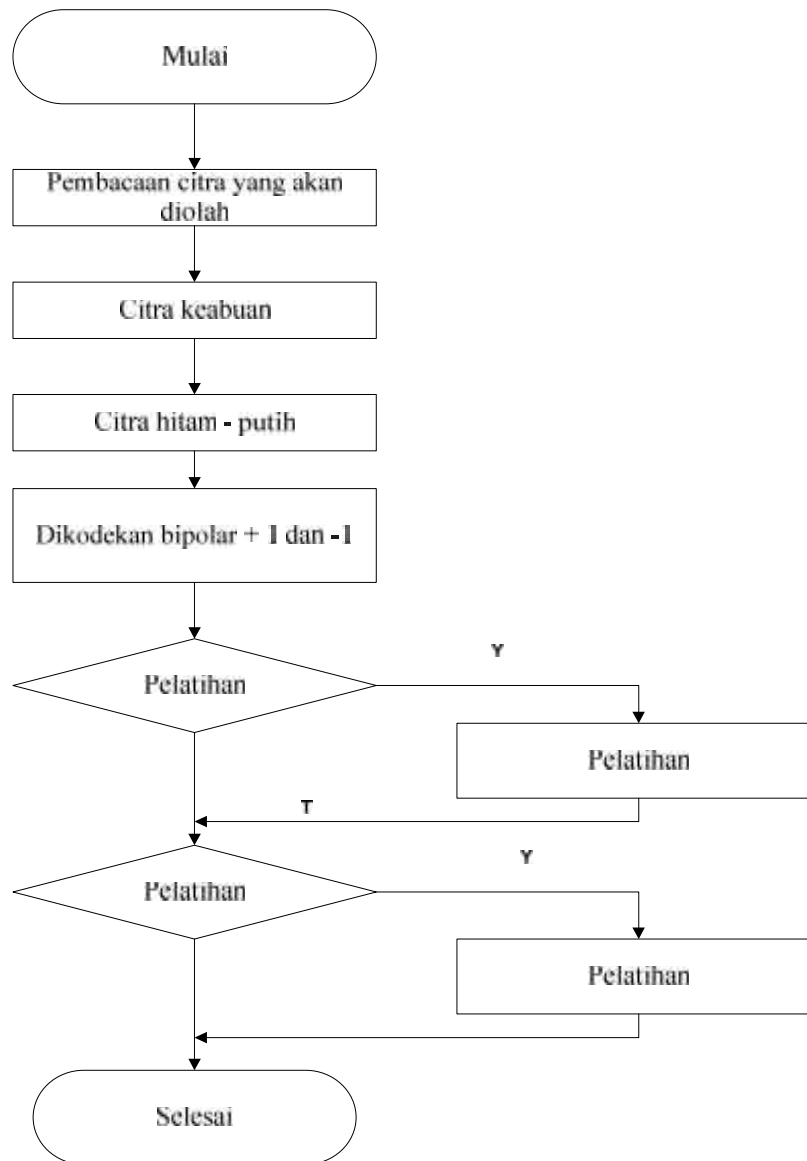
t = Target

η = Learning Rate

Pelatihan akan dihentikan jika nilai eror $(t-y)$ pada suatu epoch bernilai nol.

II.4.1. Studi Kasus Metode Hebb Rule

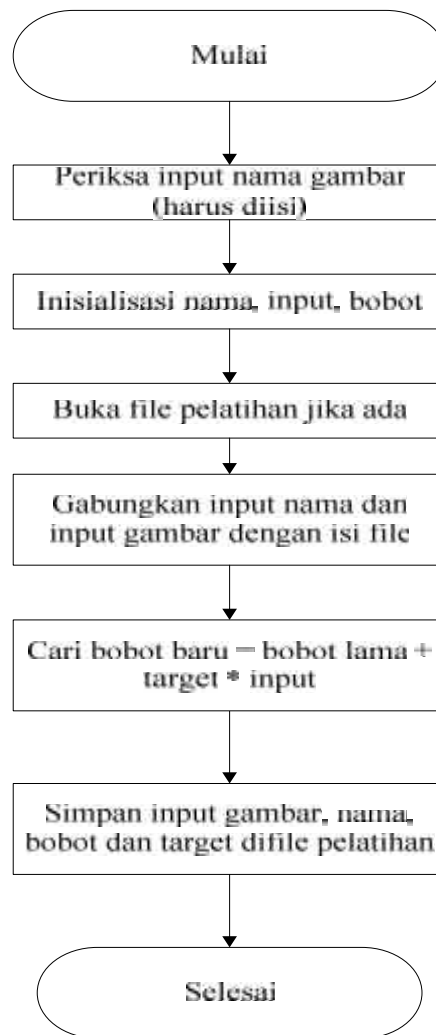
Berdasarkan penelitian Yun Enninggar (2012) program pada penelitian menggunakan perangkat lunak Matlab 6.5. Sistem pengolahan citra dilakukan berdasarkan landasan teori pada Bab II. Perancangan program pengenalan huruf Braile menggunakan jaringan syaraf tiruan dengan metode Hebb Rule menggunakan diagram alir pada gambar berikut ini.



Gambar II.2. Diagram Pengenalan Huruf Braille

(Sumber : Yun Eninggar ; 2012 : 2)

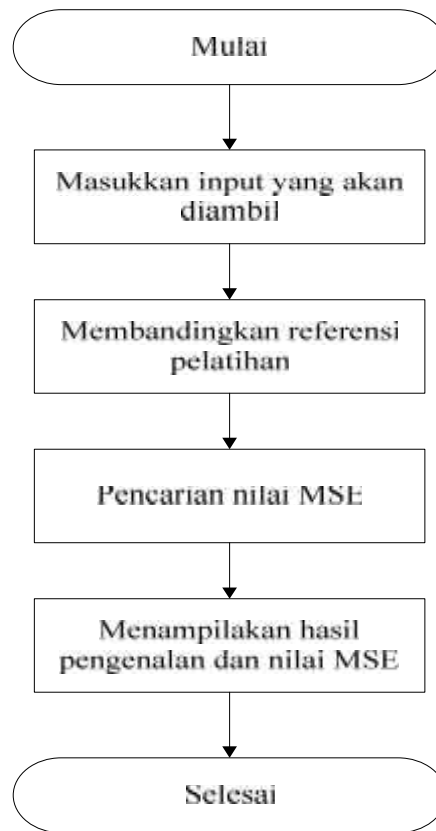
Setelah proses pembacaan citra yang akan diolah, pengubahan ke citra keabuan, pengubahan citra hitam – putih, dan pengkodean bipolar +1 dan -1, maka proses selanjutnya adalah pelatihan. Berikut ini adalah diagram alir yang akan dikerjakan pada saat proses pelatihan.



Gambar II.3. Diagram Proses Pembacaan Citra

(Sumber : Yun Eninggar ; 2012 : 2)

Setelah proses pembacaan citra yang akan diolah, pengubahan ke citra keabuan, pengubahan citra hitam – putih, pengkodean bipolar +1, dan proses pelatihan maka proses selanjutnya adalah pengenalan. Pada program untuk menentukan proses pengenalan yaitu dengan memilih tombol yang telah tersedia dengan pilihan “pengenalan”. Berikut ini adalah diagram alir yang dikerjakan pada saat proses pengenalan.



Gambar II.4. Diagram Proses Pengenalan

(Sumber : Yun Eninggar ; 2012 : 2)

II.5. Macromedia Dreamweaver

Macromedia dreamweaver adalah sebuah perangkat lunak aplikasi untuk mendesain dan membuat halaman *web*. Dengan menggunakan Dreamweaver 8, ketika membuat sebuah halaman web, Anda tidak perlu lagi mengetik kodekode HTML atau kode-kode lainnya secara manual. Anda cukup melakukan klik beberapa kali, halaman *web* yang Anda inginkan sudah jadi. Selain HTML, Dreamweaver 8 juga mendukung *CSS*, *JavaScript*, *PHP*, *ASP*, dan bahasa pemrograman lainnya untuk membuat *web*. Hal ini akan sangat menguntungkan

Anda. Sebagai contoh, jika dahulu Anda harus mengetikkan kode-kode CSS untuk membuat *Style* tertentu, maka dengan Dreamweaver 8, Anda cukup melakukan klik beberapa kali saja. Dreamweaver 8 adalah versi terbaru dari keluarga Dreamweaver. Versi pertamanya sendiri diluncurkan sekitar tahun 1994 oleh Macromedia Inc. Dalam versi terbaru ini, banyak sekali fasilitas baru yang ditambahkan. Contohnya, Anda akan dapat membuat dan menggunakan *Style* dalam CSS dengan mudah dan fleksibel. Panel untuk pengolahan CSS juga sudah diperbarui dan lebih mudah digunakan. Dreamweaver 8 juga menyediakan beberapa template halaman web baru, termasuk *fasilitas Starter Pages*. Akhirnya, selamat memulai mempelajari Dreamweaver 8 (Arief Ramadhan ; 2011 : 2).

II.6. Pengertian PHP

PHP adalah akronim dari *hypertext preprocessor*, yaitu suatu bahasa pemrograman berbasis kode-kode (*script*) yang digunakan untuk mengolah suatu data dan mengirimkannya kembali ke *web browser* menjadi kode HTML. Kode PHP mempunyai ciri-ciri khusus, yaitu :

1. Hanya dapat dijalankan menggunakan *web server*
2. Kode PHP diletakkan dan dijalankan di *web server*
3. Kode PHP dapat digunakan untuk mengakses *database*
4. Merupakan *software* yang bersifat *open source*
5. Gratis untuk di *download* dan digunakan
6. Memiliki sifat *multiform*, artinya dapat dijalankan menggunakan sistem operasi apapun.

Dengan menggunakan PHP, selain memberikan keuntungan seperti beberapa poin diatas, juga didukung oleh banyak komunitas. Hal ini yang membuat PHP terus berkembang (Diar Puji Oktavian ; 2010 : 31).

II.7. Pengertian *Database*

Secara sederhana *database* (basis data/pangkalan data) dapat diungkapkan sebagai suatu pengorganisasian data dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan cepat. Pengertian akses dapat mencakup pemerolehan data maupun pemanipulasian data seperti menambah serta menghapus data. Dengan memanfaatkan komputer, data dapat disimpan dalam media pengingat yang disebut *harddisk*. Dengan menggunakan media ini, keperluan kertas untuk menyimpan data dapat dikurangi. Selain itu, data menjadi lebih cepat untuk diakses terutama jika dikemas dalam bentuk *database*.

Pengaplikasian *database* dapat kita lihat dan rasakan dalam keseharian kita. *Database* ini menjadi penting untuk mengelola data dari berbagai kegiatan. Misalnya, kita bisa menggunakan mesin ATM (*anjungan tunai mandiri / automatic teller machine*) bank karena bank telah mempunyai *database* tentang nasabah dan rekening nasabah. Kemudian data tersebut dapat diakses melalui mesin ATM ketika bertransaksi melalui ATM. Pada saat melakukan transaksi, dalam konteks *database* sebenarnya kita sudah melakukan perubahan (*update*) data pada *database* di bank. Ketika kita menyimpan alamat dan nomor telepon di HP, sebenarnya juga telah menggunakan konsep *database*. Data yang kita simpan

di HP juga mempunyai struktur yang diisi melalui formulir (*form*) yang disediakan. Pengguna dimungkinkan menambahkan nomor HP, nama pemegang, bahkan kemudian dapat ditambah dengan alamat *email*, alamat *web*, nama kantor, dan sebagainya (Agustinus Mujilan ; 2012 : 23).

II.8. Pengertian MySQL

MySQL adalah suatu sistem manajemen basis data relasional (RDBMS - *Relational Database Management System*) yang mampu bekerja dengan cepat, kokoh, dan mudah digunakan. Contoh RDBMS lain adalah *Oracle*, *Sybase*. Basis data memungkinkan anda untuk menyimpan, menelusuri, menurutkan dan mengambil data secara efisien. *Server MySQL* yang akan membantu melakukan fungsionalitas tersebut. Bahasa yang digunakan oleh MySQL tentu saja adalah SQL - *standar* bahasa basis data relasional di seluruh dunia saat ini.

MySQL dikembangkan, dipasarkan dan disokong oleh sebuah perusahaan Swedia bernama MySQL AB. RDBMS ini berada di bawah bendera GNU GPL sehingga termasuk produk *Open Source* dan sekaligus memiliki lisensi komersial. Apabila menggunakan MySQL sebagai basis data dalam suatu situs *Web*. Anda tidak perlu membayar, akan tetapi jika ingin membuat produk RDBMS baru dengan basis MySQL dan kemudian menguálnua, anda wajib bertemu mudah dengan lisensi komersial (Antonius Nugraha Widhi Pratama ; 2010 : 10).

II.9. 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 telah ditemukan.

1. Bentuk normal tahap pertama (1 NF)

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.

PEMASOK 1

| 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 |

Gambar II.5. Normalisasi 1NF
(Sumber : Janner Simarmata ; 2010 : 79)

2. Bentuk normal tahap kedua (2NF)

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.

PEMASOK 2

| p# | Status | Kota |
|----|--------|------------|
| p1 | 20 | Yogyakarta |
| p2 | 20 | 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 |

Gambar II.6. Normalisasi 2NF
(Sumber : Janner Simarmata ; 2010 : 79)

3. Bentuk normal tahap ketiga (3 NF)

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.

PEMASOK_KOTA

| | |
|----|------------|
| p# | Kota |
| p1 | Yogyakarta |
| p2 | Medan |
| p3 | Medan |
| p4 | Yogyakarta |
| p5 | Bandung |

Gambar II.7. Normalisasi 3NF
(Sumber : Janner Simarmata ; 2010 : 79)

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 Tahap Keempat dan Kelima

Sebuah tabel relasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalued merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalued (MVD).

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*) (Janner Simarmata ; 2010 : 79).

PEGAWAI_PROYEK

| Peg# | Pry# |
|------|------|
| 1211 | p1 |
| 1211 | p3 |

Gambar II.9. Normalisasi 4NF
(Sumber : Janner Simarmata ; 2010 : 79)

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

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD 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 berbasis UML adalah sebagai berikut :


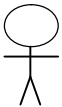

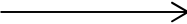
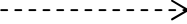
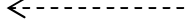
1. *Use case* Diagram

Use case diagram merupakan pemodelan untuk kelakuan (*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*, yaitu :

Tabel II.1. Simbol Use Case




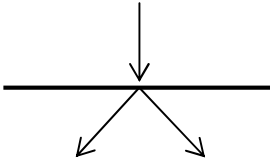
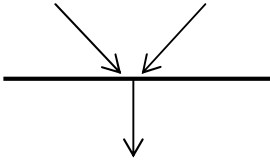
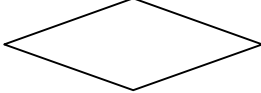

| Gambar | Keterangan |
|---|--|
|  | <i>Use case</i> menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama <i>use case</i> . |
|  | Aktor adalah <i>abstraction</i> dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasi 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 <i>use case</i> , tetapi tidak memiliki control terhadap <i>use case</i> . |
|  | Asosiasi antara aktor dan <i>use case</i> , digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan aliran data. |
|  | Asosiasi antara aktor dan <i>use case</i> yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem. |
|  | <i>Include</i> , merupakan di dalam <i>use case</i> lain (<i>required</i>) atau pemanggilan <i>use case</i> oleh <i>use case</i> lain, contohnya adalah pemanggilan sebuah fungsi program. |
|  | <i>Extend</i> , merupakan perluasan dari <i>use case</i> 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*, yaitu :

Tabel II.2. Simbol *Activity Diagram*

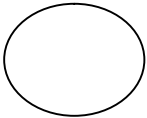
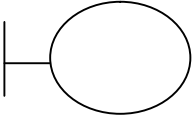
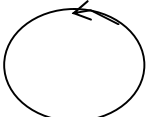

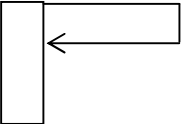


| Gambar | Keterangan |
|---|--|
|  | <i>Start point</i> , diletakkan pada pojok kiri atas dan merupakan awal aktifitas. |
|  | <i>End point</i> , akhir aktifitas. |
|  | <i>Activites</i> , menggambarkan suatu proses/kegiatan bisnis. |
|  | <i>Fork</i> (Percabangan), digunakan untuk menunjukkan kegiatan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

Tabel II.3. Simbol *Sequence Diagram*

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

(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 tanggung 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 meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.4. *Multiplicity Class Diagram*

| Multiplicity | Penjelasan |
|---------------------|---|
| 1 | Satu dan hanya satu |
| 0..* | Boleh tidak ada atau 1 atau lebih |
| 1..* | 1 atau lebih |
| 0..1 | Boleh tidak ada, maksimal 1 |
| n..n | Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4 |

(*Sumber : Windu Gata ; 2013 : 8*)