# BAB II

# TINJAUAN PUSTAKA

## II.1 Sistem Pendukung Keputusan

Sistem Pendukung Keputusan (SPK) adalah suatu sistem informasi yang mengevaluasi beberapa pilihan yang berbeda guna membantu seseorang memberikan keputusan terhadap masalahnya. Berdasarkan pada definisi yang bervariasi, SPK dapat dijelaskan sebagai sistem pembuat keputusan interaktif berbasiskan komputer yang dapat mendukung dalam pembuatan keputusan daripada menggantinya dengan yang baru, memanfaatkan data dan model, memecahkan masalah dengan struktur yang derajatnya bervariasi seperti nonstruktur, semistruktur dan unstruktur, serta berpusat pada keefektifan daripada keefisienan dalam proses pemberian keputusan.

Sistem pengambilan keputusan menggunakan logika *Fuzzy* pada studi kasus pemilihan *handphone* dengan metode *Fuzzy* model Tahani, yang dibuat oleh Denny Cristiono, ditujukan untuk menangani pencarian *handphone* yang sesuai dengan kriteria-kriteria dari konsumen. Sedangkan pada sistem pemilihan spesifikasi komputer yang akan dibuat, terdapat pencocokan antara *item* yang satu dengan yang lain. Misalkan saja jika type jenis dari prosesor adalah Intel *socket* LGA 775, maka *mainboard*-nya pun juga harus dengan jenis yang sama. Dari berbagai macam item yang tersedia, akan dipilih sesuai dengan kriteria yang diinginkan, untuk selanjutnya digabungkan menjadi suatu paket yang utuh. Pada

sistem pemilihan *handphone*, tidak terdapat penggabungan antar *item*, karena suatu *handphone* sendiri sudah merupakan satu kesatuan yang mempunyai berbagai macam variabel *fuzzy* seperti harga, dimensi (panjang, lebar dan tebal), berat, *standby time*, *talk time* dan *games* maupun variabel *non-fuzzy* seperti *polyphonic*, MMS, WAP, GPRS, *bluetooth*, *infrared* dan kamera .

## II.2 DSS *(Decision Support System)*

DSS (*Decision Support System*) adalah suatu sistem informasi yang mengevaluasi beberapa pilihan yang berbeda guna membantu seseorang memberikan keputusan terhadap masalahnya.

Berdasarkan pada definisi yang bervariasi, DSS dapat dijelaskan sebagai sistem pembuat keputusan manusia-komputer interaktif berbasiskan komputer yang dapat:

1. Mendukung dalam pembuatan keputusan daripada menggantinya dengan yang baru.
2. Memanfaatkan data dan model.
3. Memecahkan masalah dengan struktur yang derajatnya bervariasi:
    - nonstruktur (unstruktur atau ill-struktur)
    - semistruktur
    - semistruktur dan unstruktur
4. Berpusat pada keefektifan daripada keefisienan dalam proses pemberian keputusan.

Kecerdasan Buatan atau *Artificial Intelligence* (AI) merupakan salah satu bagian ilmu komputer yang membuat mesin (komputer) dapat melakukan pekerjaan seperti dan sebaik yang dilakukan oleh manusia. Agar komputer dapat bertindak seperti dan sebaik manusia, maka komputer juga harus diberi bekal pengetahuan, dan mempunyai kemampuan untuk menalar. Untuk itu AI memberikan beberapa metoda untuk membekali komputer dengan kedua komponen tersebut agar komputer dapat menjadi mesin yang pintar.

Untuk melakukan aplikasi kecerdasan buatan ada dua bagian utama yang sangat dibutuhkan, yaitu :

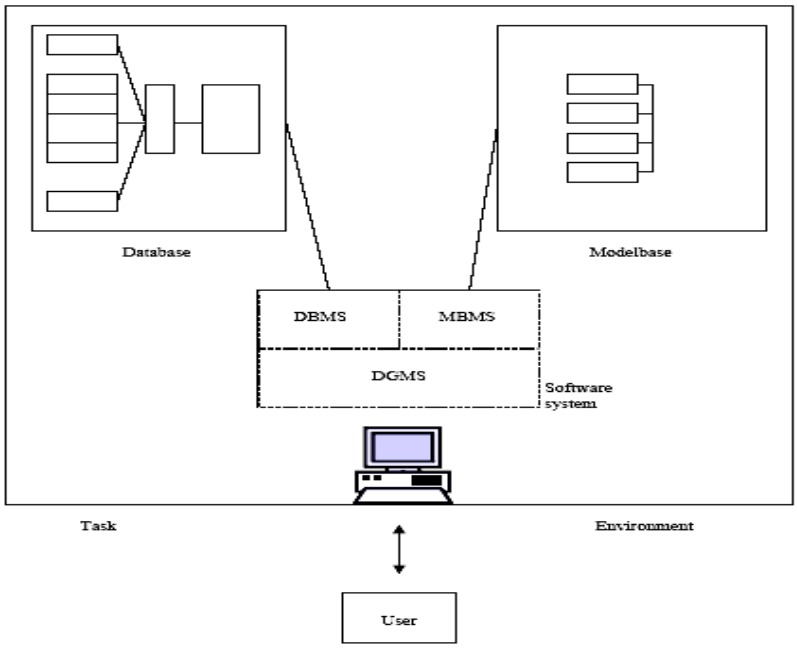
a. **Basis Pengetahuan (*Knowledge Base*),** berisi fakta-fakta, teori, pemikiran dan hubungan antara satu dengan lainnya.
b. **Motor Inferensi (*Inference Engine*),** yaitu kemampuan menarik kesimpulan berdasarkan pengalaman.

FDSS (*Fuzzy Decision Support System*) merupakan sistem pembuat keputusan manusia-komputer untuk mendukung keputusan manajerial, dan intuisi untuk memecahkan masalah manajerial dengan memberikan informasi yang diperlukan, menghasilkan, mengevaluasi dan memberikan putusan alternatif [4].

### II.2.1 Arsitektur DSS *(Decision Support System)*

Langkah pertama pada proses pembuatan keputusan yaitu dengan membuat model pendukung keputusan. *Interface* subsistem *user* menjembatani untuk menuju ke DBMS (*Database Management Systems*) dan MBMS (*Model-Based*

*Management Systems*). DBMS merupakan seperangkat program komputer yang membuat dan mengatur *database*. DBMS dapat menjadi salah satu program tersendiri atau disatukan dengan generator DSS yang mengijinkan *user* untuk membuat *file database* yang digunakan sebagai *input* pada DSS. MBMS merupakan seperangkat program komputer yang tersimpan dalam generator DSS yang mengijinkan *user* untuk membuat, meng-*edit*, meng-*update* dan/atau menghapus model. *User* membuat beberapa model dan *file* relasi *database* untuk membuat keputusan yang spesifik. Model dan *database* yang telah dibuat disimpan di model utama dan *database*-nya di media penyimpan seperti hardisk. Dari sudut pandang *user*, subsistem *interface user* hanya merupakan bagian dari komponen DSS. Oleh karena itu, dalam memberikan *interface user* yang efektif harus mengambil beberapa persoalan penting sebagai bahan pertimbangan, termasuk pemilihan media *input* dan *output*, desain layar, penggunaan warna, format penyajian data dan informasi, penggunaan jenis *interface* yang berbeda, dan lain-lain.




**Gambar II.1 Komponen DSS(Decision Support Sytem)**

**(Sumber : Jurnal Informatika)**

## II.3 Konsep Dasar Logika Fuzzy

Teori himpunan *fuzzy* diperkenalkan pertama kali oleh Lotfi A. Zadeh pada tahun 1965. Ada beberapa alasan mengapa orang menggunakan logika *fuzzy*, antara lain :

1. Konsep logika *fuzzy* mudah dimengerti. Konsep matematis yang mendasari penalaran *fuzzy* sangat sederhana dan mudah dimengerti.
2. Logika *fuzzy* sangat fleksibel.
3. Logika *fuzzy* memiliki toleransi terhadap data-data yang tidak tepat.
4. Logika *fuzzy* mampu memodelkan fungsi-fungsi nonlinear yang sangat kompleks.

5. Logika *fuzzy* dapat membangun dan mengaplikasikan pengalaman-pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan.
6. Logika *fuzzy* dapat bekerjasama dengan teknik-teknik kendali secara konvensional.
7. Logika *fuzzy* didasarkan pada bahasa alami.

Dalam logika *fuzzy* dikenal berhingga keadaan dari nilai "0" sampai ke nilai "1". Logika *fuzzy* tidak hanya mengenal dua keadaan tetapi juga mengenal sejumlah keadaan yang berkisar dari keadaan salah sampai keadaan benar .

Ada beberapa hal yang perlu diketahui dalam memahami sistem *fuzzy*, yaitu

1. Variabel *Fuzzy*

   Variabel *Fuzzy* merupakan variabel yang hendak dibahas dalam suatu sistem *Fuzzy*.

2. Himpunan *Fuzzy*

   Himpunan *Fuzzy* adalah himpunan yang tiap elemennya mempunyai derajat keanggotaan tertentu terhadap himpunannya.

   Himpunan *Fuzzy* memiliki dua atribut, yaitu :

   - Linguistik, yaitu penamaan suatu grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami.
   - Numeris, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel.

3. Semesta Pembicaraan

Semesta pembicaraan adalah suatu keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *Fuzzy*. Nilai semesta pembicaraan dapat berupa bilangan positif atau bilangan negatif.

4. *Domain*

*Domain* himpunan *Fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *Fuzzy*. Nilai *domain* dapat berupa bilangan positif maupun bilangan negatif .

## II.4 Karakteristik Logika Fuzzy

Logika *Fuzzy* memiliki beberapa karakteristik yaitu himpunan *Fuzzy* dan fungsi keanggotaan.

### II.4.1 Himpunan Fuzzy

Pada logika *boolean*, sebuah individu dipastikan sebagai anggota dari salah satu himpunan saja, sedangkan pada himpunan *fuzzy* sebuah individu dapat masuk pada dua himpunan yang berbeda. Seberapa besar eksistensinya dalam himpunan tersebut dapat dilihat pada nilai keanggotaannya .

Himpunan *fuzzy A* pada semesta *X* dinyatakan sebagai himpunan pasangan berurutan (*set of ordered pairs*) baik diskrit maupun kontinu.

$$A = \{(x, \mu_A(x)) | x \in X\} \qquad (2.1)$$

Dimana $\mu_A(x)$ fungsi keanggotaan himpunan *fuzzy A*. Fungsi keanggotaan memetakan setiap $x \in X$ pada suatu nilai antara [0,1] yang disebut derajat keanggotaan (*membership grade* atau *membership value*).

Beberapa operasi pada himpunan *fuzzy* adalah sebagai berikut :

- *Support*

  *Support* dari himpunan *fuzzy A* adalah kumpulan semua titik $x \in X$ yang memberikan nilai $\mu_A(x) > 0$, atau

$$Support(A) = \{x | \mu_A(x) > 0\} \qquad (2.2)$$

- *Core*

  *Core* dari himpunan *fuzzy A* adalah kumpulan semua titik $x \in X$ yang memberikan nilai $\mu_A(x) = 1$, atau

$$Core(A) = \{x | \mu_A(x) = 1\} \qquad (2.3)$$

- *Crossover*

  Titik *crossover* dari himpunan *fuzzy A* adalah titik dimana $\mu_A(x) = 0.5$ atau

$$Crossover(A) = \{x | \mu_A(x) = 0.5\} \qquad (2.4)$$

- Fungsi *Singleton*

  Fungsi *singleton* adalah himpunan *fuzzy* yang memiliki *support* pada satu titik $x \in X$ dengan $\mu_A(x) = 1$.

### II.4.2 Fungsi Keanggotaan *(*Membership Function*)*

Fungsi keanggotaan adalah suatu kurva yang menunjukkan pemetaan titik-titik *input* data ke dalam nilai keanggotaannya (disebut juga dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan nilai keanggotaan adalah dengan melalui pendekatan fungsi [6]. Derajat keanggotaan dalam himpunan (*degree of membership*) dilambangkan dengan μ.

Dalam sistem *fuzzy* banyak dikenal bermacam-macam fungsi keanggotaan (*membership function*). Beberapa fungsi keanggotaan yang sering digunakan adalah:

- Fungsi keanggotaan Linear (2.5)

$$\mu[x] = \begin{cases} 0; & x \le a \text{ atau } x \ge c \\ (x - a) / (b - a); & a \le x \le b \\ 1; & x \ge b \end{cases}$$

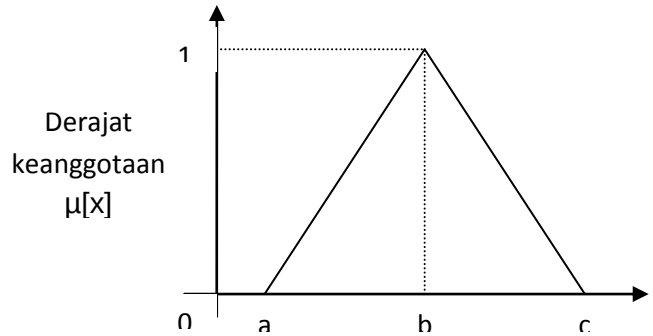
- Fungsi keanggotaan segitiga (2.6)

$$\mu[x] = \begin{cases} 0; & x \le a \text{ atau } x \ge c \\ (x - a) / (b - a); & a \le x \le b \\ (c - x) / (c - b); & b \le x \le c \end{cases}$$

Dalam kasus yang dibahas, fungsi keanggotaan yang dipakai adalah Representasi Kurva Segitiga dan Representasi Kurva Bahu.

1. Representasi Kurva Segitiga

   Kurva segitiga pada dasarnya merupakan gabungan antara dua garis.




**Gambar II.2 Fungsi Keanggotaan Kurva Segitiga**

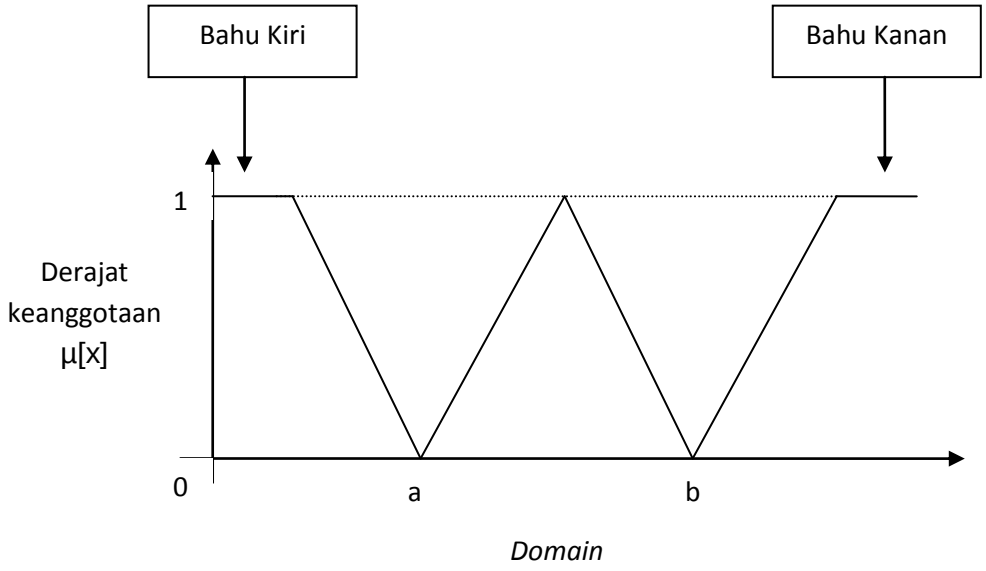
**(Sumber : Jurnal Informatika)**

Fungsi keanggotaan: (2.8)

$$\mu[x] = \begin{cases} 0 & x \le a \text{ atau } x \ge c \\ (x - a) / (b - & a \le x \le b \\ (c - x) / (c - & b \le x \le c \end{cases}$$

2. Representasi Kurva Bahu

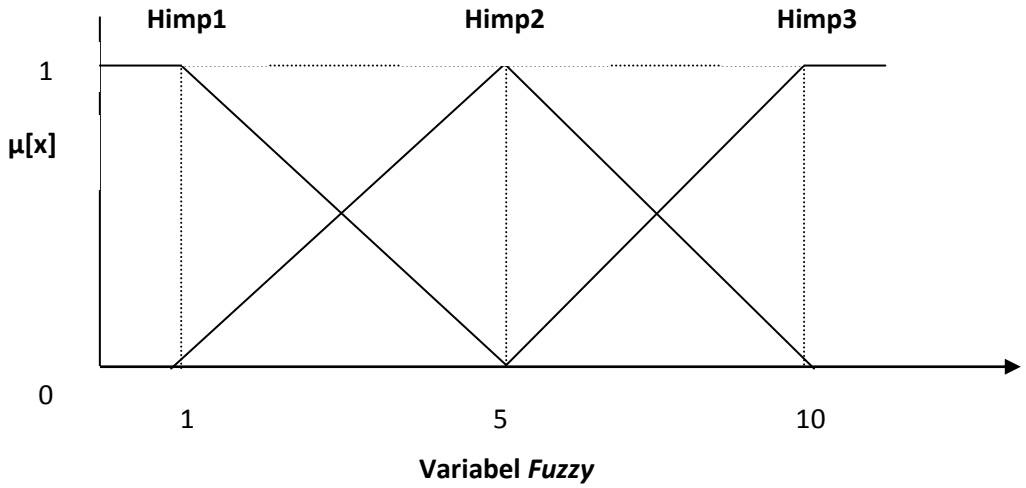
Kurva bahu merupakan daerah yang terletak di tengah-tengah suatu variabel yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan sisi kirinya akan naik dan turun. Himpunan *fuzzy* bahu digunakan untuk mengakhiri variabel suatu daerah *fuzzy*.




**Gambar II.3 Fungsi Keanggotaan Kurva Bahu**

**(Sumber : Jurnal Informatika)**

Fungsi keanggotaan pada kurva segitiga dan fungsi keanggotaan pada kurva bahu dapat dilihat pada gambar 2.4




**Gambar II.4 Himpunan Fuzzy untuk Suatu Variabel**

**(Sumber : Jurnal Informatika)**

## II.5 Operator Dasar Zadeh

❖ Operator AND

Operator ini berhubungan dengan operasi interseksi pada himpunan. *Fire strength* sebagai hasil operasi dengan operator AND diperoleh dengan mengambil nilai keanggotaan terkecil antar elemen pada himpunan-himpunan yang bersangkutan.

$$\mu A \cap B = \min(\mu_A[x], \mu_B[y]) \qquad (2.9)$$

❖ Operator OR

Operator ini berhubungan dengan operasi union pada himpunan. *Fire strength* sebagai hasil operasi dengan operator OR diperoleh dengan

mengambil nilai keanggotaan terbesar antara elemen pada himpunan-himpunan yang bersangkutan.

$$\mu A \cup B = \max(\mu_A[x], \mu_B[y]) \qquad (2.10)$$

❖ Operator NOT

Operator ini berhubungan dengan operasi komplemen pada himpunan. *Fire strength* sebagai hasil operasi dengan operator NOT diperoleh dengan mengurangkan nilai keanggotaan elemen pada himpunan yang bersangkutan dari 1.

$$\mu A^1 = 1 - \mu_A[x] \qquad (2.11)$$

## II.6 Fuzzifikasi

Fuzzifikasi adalah pengubahan seluruh variabel *input/output* ke bentuk himpunan *fuzzy*. Rentang nilai variabel *input* dikelompokkan menjadi beberapa himpunan *fuzzy* dan tiap himpunan mempunyai derajat keanggotaan tertentu.

Bentuk fuzzifikasi yang dipakai pada sistem ini adalah bentuk segitiga dan bentuk bahu. Bentuk fuzzifikasi menentukan derajat keanggotaan suatu nilai rentang *input/output*. Derajat keanggotaan himpunan *fuzzy* dihitung dengan menggunakan rumus fungsi keanggotaan dari segitiga fuzzifikasi .

## II.7 Inferensi Fuzzy (logika Pengambilan Keputusan)

Setelah fungsi keanggotaan untuk variabel masukan dan keluarannya ditentukan, basis aturan pengendalian dapat dikembangkan untuk menghubungkan

aksi keluaran pengendali terhadap kondisi masukannya. Tahap ini disebut sebagai tahap inferensi, yakni bagian penentuan aturan dari sistem logika *fuzzy*. Sejumlah aturan dapat dibuat untuk menentukan aksi pengendali *fuzzy* [8].

Pada basis aturan, aturan *If-Then* tersebut dapat menghubungkan banyak variabel masukan dan keluaran. Masukan *x* dipetakan menjadi keluaran *y*. Aturan *if-then* diinterpretasikan sebagai implikasi *fuzzy*. Terdapat banyak sekali model interpretasi implikasi yang telah dikembangkan. Pada sistem ini, metode yang digunakan adalah Metode MAMDANI.

### II.7.1 Metode Mamdani

Metode Mamdani sering juga dikenal dengan nama Metode Max-Min. Metode ini diperkenalkan oleh Ebrahim Mamdani pada tahun 1975. Untuk mendapatkan *output*, diperlukan empat tahapan :

1. Pembentukan Himpunan *Fuzzy*

   Pada Metode Mamdani, baik variabel *input* maupun variabel *output* dibagi menjadi satu atau lebih himpunan *fuzzy*.

2. Aplikasi Fungsi Implikasi

   Pada Metode Mamdani, fungsi implikasi yang digunakan adalah Min.

3. Komposisi Aturan

   Tidak seperti penalaran monoton, apabila sistem terdiri dari beberapa aturan, maka inferensi diperoleh dari kumpulan dan korelasi antar

aturan. Ada tiga metode yang digunakan dalam melakukan inferensi system *fuzzy*, yaitu max, additive dan probabilistic OR (probor).

a. Metode Max (Maximum)

Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara mengambil nilai maksimum aturan, kemudian menggunakannya untuk memodifikasi daerah *fuzzy*, dan mengaplikasikannya ke *output* dengan menggunakan operator OR (union). Secara umum dapat dituliskan :

$$\mu_{sf}[x_i] \leftarrow \max(\mu_{sf}[x_i], \mu_{kf}[x_i]) \qquad (2.14)$$

dengan :

$\mu_{sf}[x_i]$ = nilai keanggotaan solusi *fuzzy* sampai aturan ke-i;

$\mu_{kf}[x_i]$ = nilai keanggotaan konsekuen *fuzzy* sampai aturan ke-i;

Apabila digunakan fungsi implikasi MIN, maka metode komposisi ini sering disebut dengan nama MAX-MIN atau MIN-MAX atau MAMDANI.

b. Metode Additive (Sum)

Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara melakukan *bounded-sum* terhadap semua *output* daerah *fuzzy*. Secara umum dituliskan :

$$\mu_{sf}[x_i] \leftarrow \min(1, \mu_{sf}[x_i] + \mu_{kf}[x_i]) \qquad (2.15)$$

dengan :

$\mu_{sf}[x_i]$ = nilai keanggotaan solusi *fuzzy* sampai aturan ke-i;

$\mu_{kf}[x_i]$ = nilai keanggotaan konsekuen *fuzzy* sampai aturan ke-i;

c. Metode Probabilistik OR (probor)

Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara melakukan *product* terhadap semua *output* daerah *fuzzy*. Secara umum dituliskan :

$$\mu_{sf}[x_i] \leftarrow (\mu_{sf}[x_i] + \mu_{kf}[x_i]) - (\mu_{sf}[x_i] * \mu_{kf}[x_i])$$

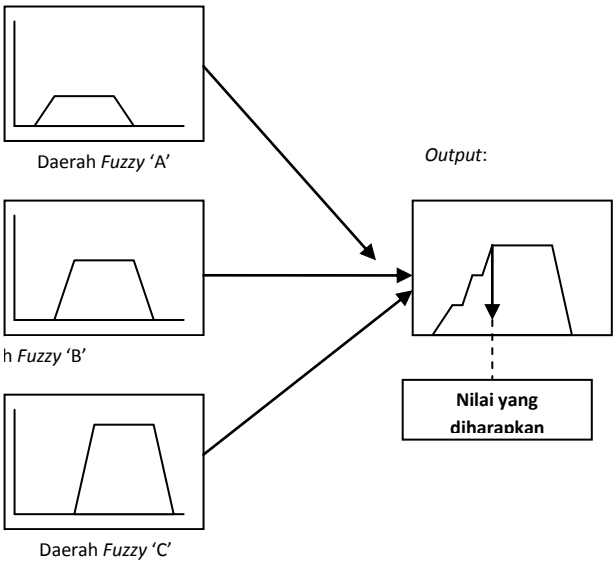
dengan : (2.16)

$\mu_{sf}[x_i]$ = nilai keanggotaan solusi *fuzzy* sampai aturan ke-i;

$\mu_{kf}[x_i]$ = nilai keanggotaan konsekuen *fuzzy* sampai aturan ke-i;

4. Penegasan (*defuzzy*)

*Input* dari proses defuzzifikasi adalah suatu himpunan *fuzzy* yang diperoleh dari komposisi aturan-aturan *fuzzy*, sedangkan *output* yang dihasilkan merupakan suatu bilangan pada domain himpunan *fuzzy* tersebut. Sehingga jika diberikan suatu himpunan *fuzzy* dalam range tertentu, maka harus dapat diambil suatu nilai crisp tertentu sebagai *output* seperti terlihat pada gambar 2.5




**Gambar II.5 Proses Defuzzifikasi.**

**(Sumber : Jurnal Informatika, Desember 2008)**

Ada beberapa metode defuzzifikasi pada komposisi aturan MAMDANI. Pada sistem ini proses defuzifikasinya menggunakan metode *Mean Of Maximum* (*MOM*).

a. Metode Centroid (*Composite Moment*)

   Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil titik pusat (z*) daerah *fuzzy*.

b. Metode Bisektor

   Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil nilai pada domain *fuzzy* yang memiliki nilai keanggotaan separuh dari jumlah total nilai keanggotaan pada daerah *fuzzy*.

c. Metode *Mean Of Maximum* (*MOM*)

Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil nilai rata-rata domain yang memiliki nilai keanggotaan maksimum.

d. Metode *Largest Of Maximum* (*LOM*)

Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil nilai terbesar dari domain yang memiliki nilai keanggotaan maksimum.

e. Metode *Smallest Of Maximum* (*SOM*)

Pada metode ini, solusi *crisp* diperoleh dengan cara mengambil nilai terkecil dari domain yang memiliki nilai keanggotaan maksimum.

## II.8 Tipe-Tipe Fuzzy

Ada beberapa macam tipe *Fuzzy*. Yang akan digunakan pada sistem ini adalah tipe *Fuzzy Database*.

### II.8.1 Fuzzy Database

*Fuzzy Database* dapat diartikan sebagai merepresentasikan, memasukkan, dan memanipulasi informasi yang tidak tepat dan tidak pasti. *Query* pada logika *fuzzy* dapat digunakan untuk pengambilan data yang diinginkan, tanpa memerlukan pendefinisian parameter yang pasti. Proses *query fuzzy* mencakup logika *boolean* yang hasil pencariannya berupa nilai benar atau salah dan juga akan menghasilkan nilai x% benar atau x% salah dari nilai keanggotaannya.

### II.8.2 Fuzzy Database Model Tahani

Dengan menggunakan *database* standar, kita bisa mencari data-data dengan spesifikasi tertentu dengan menggunakan *query*.

Pada kenyataannya, seseorang kadang membutuhkan informasi dari data yang bersifat *ambiguous*. Dan jika hal tersebut terjadi maka dapat digunakan *fuzzy database*. Selama ini sudah ada beberapa penelitian tentang *fuzzy database*. Salah satu diantaranya adalah model Tahani.

*Fuzzy database* model Tahani masih menggunakan relasi standar, tetapi model Tahani menggunakan teori himpunan *fuzzy* pada suatu variabel untuk mendapatkan informasi pada *query*-nya. Sehingga pada pencarian data menggunakan rumus dari derajat keanggotaan pada suatu variabel himpunan *fuzzy*.

Dalam hal ini menggunakan rumus dari derajat keanggotaan segitiga, dengan representasi kurva segitiga dan representasi kurva bahu .

## II.9 Pembentukan Query

Suatu sistem *query* adalah semacam sistem pengembalian informasi yang digunakan untuk mendapatkan kembali obyek yang relevan dari suatu *database*.

Pembentukan *query* pada *database* sistem *fuzzy* digunakan pada proses fuzzifikasi dan proses defuzzifikasi.

*Query* digunakan untuk mencari data dari *database* serta meng-*input*-kan data ke *database*. Seluruh nilai yang di-*input*-kan ke *database* merupakan nilai yang memenuhi kriteria dari fungsi derajat keanggotaan yang di-*input*-kan, dengan batas nilai antara 0 dan 1.

```
SELECT FIELD_A, FIELD_B, FIELD_C

  FROM TABLE_A

  WHERE FIELD_A >= 0 AND FIELD_B >= 0
```

Sedangkan data-data nilai yang diambil merupakan hasil perhitungan dari rumus fungsi derajat keanggotaan yang disimpan ke *database*. Nilai-nilai tersebut merupakan hasil dari proses fuzzifikasi, yang berguna sebagai nilai *input* untuk proses defuzzifikasi.

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

Menurut Adi Nugroho (2010) UML (*Unified Modeling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma ‘berorientasi objek’. Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami. Sedangkan

Menurut Julius Hermawan (2010), UML menyediakan beberapa notasi dan artifak standard yang bisa digunakan sebagai alat komunikasi bagi para proses analisis dan desain.

Kesimpulan UML adalah sebagai berikut: Sistem atau perangkat lunak yang berorientasi objek yang sesungguhnya digunakan untuk penyederhanaan permasalahan yang komplek sehingga lebi mudah dipelajai dan dipahami. Sebuah bahasa yang berdasarkan grafik atau gambar untuk menvisualisasikan, menspesifikasikan membangun dan mendokumentasikan sebuah sistem pengembangan perangkat lunak berbasis objek.

**Tabel II.1. View dan Diagram dalam UML**

| Major Area | View | Diagrams | Main Concepts |
|---|---|---|---|
| Structural | Static view | Class diagram | Class, association, generalization, dependency, realization, interface. |
| | Use case view | Use case diagram | Use case, actor, association, extend, include, use case generalization |
| | Implementation view | Component diagram | Component, interface, dependency, realization |
| | Deployment view | Deployment diagram | Node, component, dependency, location |
| Dynamic | State machine | Statechart diagram | State, event, transition, action |
| | Activity view | Activity diagram | State, activity, completion transition, fork, join |

| | | | |
|---|---|---|---|
| | Interaction view | Squence diargam | Interaction, object, message, activation |
| | | Collaboration diagram | Collaboration, interaction, collaboration role, message |
| Model management | Model management view | Class diagram | Package, subsystem, model |
| Extensibility | All | All | Constraint, stereotype, tagged values |

**(Sumber : Adi Nugroho ; 2010)**

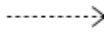
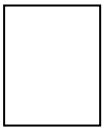
### II.10.1. *Use Case Diagram*

Menurut Adi Nugroho (2010) *Use Case Diagram* merupakan unit fungsional itas koheren yang diekspresikan sebagai transaksi yang terjadi antara actor dan sistem. *Use Case* sesungguhnya merupakan unit koheren dari fungsionalitas sitem/perangkat lunak yang tampak dari luar dan diekspresikan sebagai urutan pesan – pesan yang dipertukarkan unit – unit sistem dengan satu atau lebih actor yag ada diluar sistem. Sedangkan

Menurut Julius Hermawan (2010) Use case menjelaskan urutan kegiatan yang dilakukan Aktor dan system untuk mencapai tujuan tertentu.

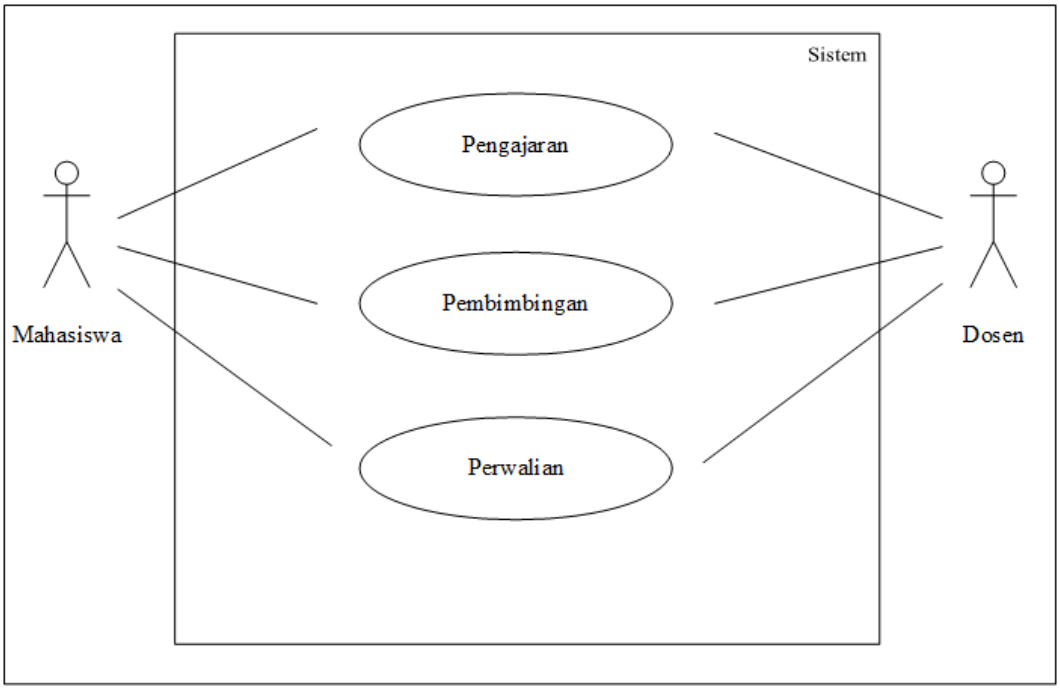
**Tabel II.2 Komponen Use Case Diagram**

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| 1 |  | *Actor* | Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*. |

| | | | |
|---|---|---|---|
| 2 |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*). |
| 3 |  | *Generalization* | Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
| 4 |  | *Include* | Menspesifikasikan bahwa *use case* sumber secara *eksplisit*. |
| 5 |  | *Extend* | Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan. |
| 6 |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya. |
| 7 |  | *System* | Menspesifikasikan paket yang menampilkan sistem secara terbatas. |
| 8 |  | *Use Case* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 9 |  | *Collaboration* | Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi). |
| 10 |  | *Note* | Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi |

**(Sumber : Grady Booch ; 2013)**

Kegunaan dari *use case* adalah untuk mendaftarkan *actor-actor dan use case-use case* dan memperlihatkan *actor-actor* mana yang berpatisipasi dalam masing-masing *use case*.




**Gambar II.6 *Use Case***

**(Sumber : Adi Nugroho ; 2010 )**

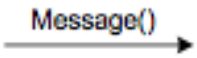
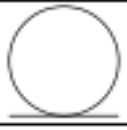
Kesimpulan *Use Case Diagram* adalah Sebagai berikut: Unit fungsional antara aktor dan sistem yang saling terkait dan konsisten yang terjadi diluar sistem tersebut. Dan Use Case Diagram dapat menggambarkan hubungan use case dengan actor.

### II.10.2. *Sequence Diagram*

Menurut Adi Nugroho (2010) *Sequence Diagram* memperlihatkan interaksi sebagai diagram dua matra (dimensi). Matra vertikal adalah sumbu

waktu bertambah dari atas ke bawah. Matra horizontal memperihatkan peran pengklasifikasi yang merepresentasikan objek–objek mandiri yang terlibat dalam kolaborasi. Masing–masing peran pengklasifikasikan direpresentasikan sebagai kolom–kolom vertikal dalam *sequence diagram* sering disebut garis waktu (*lifeline*). Selama objek ada, peran digambarkan menggunakan garis tegas. Selama aktivasi prosedur pada objek aktif, garis waktu digambarkan sebagai garis ganda. Pesan-pesan digambarkan sebagai suatu tanda panah dari garis waktu suatu objek ke garis waktu objek lainnya. Panah-panah yang menggambarkan aliran pesan antarperan pengklasifikasi digambarkan dalam urutan waktu kejadiannya dari atas ke bawah.

**Tabel II.3 Komponen Sequence Diagram**

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| 1 |  | *LifeLine* | Objek *entity*, antarmuka yang saling berinteraksi. |
| |  | *Actor* | Digunakan untuk menggambarkan user / pemgguna. |
| 2 |   | *Message* | Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi. |
| 3 |  | *Boundary* | Digunakan untuk menggambarkan sebuah form. |
| 4 |  | *Control Class* | Digunakan untuk menghubungkan *boundary* dengan tabel. |
| 5 |  | *Entity Clas* | Digunakan untuk menggambarkan hubungan kegiatan yang akan dilakukan. |

**( Sumber : Grady Bcooh, 2013)**

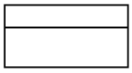
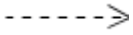
Kesimpulan *sequence diagram* adalah sebagai berikut: Menggambarkan interaksi antar objek didalam dan sekitar sistem pengguna,display, dan sequence diagram terdiri dari horizontal dan vertikal . Sequence diagram biasa digunakan untuk skenarioa atau rangkaian langkah-langkah yang dilakukan sebagai respon dari sebuah event untuk menghasilkan output tertentu.

### II.10.3. *Class Diagram*

Menurut Adi Nugroho (2010 ) *Class Diagram* adalah kelas yang sesungguhnya mempresentasikan suatu kodi dalam konsep diskret di dalam aplikasi yang dimodelkan, sesuatu yang bersifat fisika (misalnya mobil,pesawat terbang dan sebagainya), sesuatu yang bersifat bisnis (misalnya pesanan), sesuatu yang bersifat logika (misalnya penjadwalan ), sesuatu yang sangat berkait dengan aplikasi (misalnya tombol-tombol, ikon-ikon, dan sebagainya), sesuatu yang merupakan konsep yang dikenali dalam terminalogi ilmu komputer (misalnya tabel hash atau berbagai metode pengurutan dan pencarian (sorthing dan serching), atau sesuatu yang bersifat perilaku (behaviour) misalnya pekerjaan tertentu. Sedangkan

Menurut Julius Hermawan (2010) *Class Diagram* merupakan pembentuk utama dari system berorientasi objek karena *class* menunjukan kumpulan objek yang memiliki atribut dan operasi yang sama.

**Tabel II.4 Komponen Class Diagram**

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| 1 |  | *Generalization* | Hubungan dimana objek anak *(descendent)* berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*). |
| 2 |  | *Nary Association* | Upaya untuk menghindari asosiasi dengan lebih dari 2 objek. |
| 3 |  | *Class* | Himpunan dari objek-objek yang berbagi atribut serta operasi yang sama. |
| 4 |  | *Collaboration* | Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor |
| 5 |  | *Realization* | Operasi yang benar-benar dilakukan oleh suatu objek. |
| 6 |  | *Dependency* | Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempegaruhi elemen yang bergantung padanya elemen yang tidak mandiri |
| 7 |  | *Association* | Apa yang menghubungkan antara objek satu dengan objek lainnya |

**(Sumber : Grady Booch, 2013)**

Kesimpulan *class diagram* adalah sebagai berikut: Suatu sistem yang mempunyai kelas untuk mempresentasikan kodi dalam diskret yang didalamnya aplikasi yang suda dimodelkan. Yang bersifat fisika, bisnis dan logika. Diagram

Class merupakan konsep yang dikenali dalam terminalogi ilmu komputer yang memiliki metode pengurutan dan pencarian atau sesuatu yang bersifat perilaku.

### II.10.4. *Activity Diagram*

Menurut Adi Nugroho (2010) Diagram aktifitas (*Actvity Diagram)* merupakan bentuk khusus dari *State machine* yang bertujuan memodelkan komputasi-komputasi dan aliran-aliran kerja terjadi dalam sistem atau perangkat lunak yang sedang dikembangkan, suatu diagram aktivitas memuat didalamnya *activity state* merepresentasikan eksekusi peryataan dalam suatu prosedur atau kinerja suatu aktifitas dalam suatu aliran kerja. Dan diagram aktivitas dapat memperlihatkan aliran nilai-nilai objek, seperti layaknya aliran-aliran kendali.

**Tabel II.5 Komponen Activity Diagram**

| NO | GAMBAR | NAMA | KETERANGAN |
|---|---|---|---|
| **1** |  | *Actifity* | Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain |
| **2** |  | *Action* | State dari sistem yang mencerminkan eksekusi dari suatu aksi |
| **3** |  | *Initial Node* | Bagaimana objek dibentuk atau diawali. |
| **4** |  | *Actifity Final Node* | Bagaimana objek dibentuk dan dihancurkan |
| **5** |  | *Fork Node* | Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran |

**(Sumber : Grady Booch ,2013)**

Kesimpulan *activity diagram* adalah sebagai berikut: *activity diagram* merupakan bentuk kusus dari *state mechine* yang bertujuan memodelkan komputasi-komputasi dan aliran-aliran kerja terjadi dalam sistem atau perangkat lunak yang dikembangkan dan dapat memperlihatkan aliran nilai-nilai objek. *activity diagram* sistem yang menggambarkan alir activitas dalam sistem yang sedang dirancang. Dan merepresentasikan eksekusi penyataan dalam suatu prosedur atau kinerja suatu aktifitas dalam suatu aliran kerja.

## II.11. Normalisasi

Menurut Kusrini (2011) Normalisasi merupakan cara pendekatan dalam membangun desain logika basis data relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal. Sedangkan

Menurut Samiaji Sarosa (2010:5), Normalisasi adalah teknik yang dirancang untuk merancang tabel basis data relasional untuk meminimalkan duplikasi data dan menghindarkan basis data tersebut anomali. Pada dasarnya desain logika basis data relasional dapat menggunakan prinsip normalisasi maupun transformasi dari E-R ke bentuk fisik.

Dalam perpektif normalisasi sebuah database dikatakan baik jika setiap tabel yang membentuk basis data sudah berada dalam keadaan normal. Suatu tabel dikatakan normal jika :

a. Jika ada dekomposisi/penguraian tabel, maka dekomposisinya dijamin aman (*lossless-join decompisition*)
b. Terpeliharanya ketergantungan functional pada saat perubahan data (*dependency preservation*)
c. Tidak melanggar *Boyce Code Normal Form* (BCNF), jika tidak bisa minimal tidak melanggar bentuk normalisasi ketiga

Yang dimaksud dengan ketergantungan fungsional (*functional dependency*) adalah :

Diberikan sebuah tabel/relasi T, atribut B dari T bergantung secara fungsi pada atribut A dari T jika dan hanya jika setiap nilai B dari T punya hubungan dengan tepat satu nilai A dalam T (dalam setiap satu waktu).

Beberapa kondisi yang diujikan pada proses normalisasi :

a. Menambah data/insert
b. Mengedit/update
c. Menghapus/delete
d. Membaca/retrieve

### II.11.1. Bentuk-bentuk Normalisasi

#### a. Bentuk tidak normal

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi. Data dikumpulkan apa adanya sesuai keadaannya.

**b. Bentuk normal tahap pertama (1NF)**

Sebuah table disebut 1NF jika memenuhi syarat-syarat sebagai berikut :

1. Tidak ada baris yang duplikasi dalam tabel tersebut
2. Masing-masing cell bernilai tunggal

Catatan : permintaan yang menyatakan tidak ada baris yang duplikasi dalam sebuah tabel berarti tabel tersebut memiliki sebuah kunci, meskipun kunci tersebut merupakan kombinasi dari semua kolom.

Contoh :

- Tabel kuliah (kode_kul, nama_kul, sks, semester, nama_dos)
- Tabel jadwal (kode_kul, waktu, ruang)

**c. Bentuk normal tahap kedua (2NF)**

Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam primary key memiliki ketergantungan fungsional pada primary key secara utuh. Sebuah tabel dikatakan tidak memenuhi 2NF jika ketergantungannya hanya bersifat parsial (hanya tergantung pada sebagian dari *primary key*).

Contoh :

Misal tabel nilai terdiri dari atribut kode_kul, nim dan nilai. Jika pada tabel nilai, misalnya kita tambahkan sebuah atribut yang bersifat redundan, yaitu nama_mhs, maka tabel nilai ini dianggap melanggar

2NF. *Primary key* pada tabel nilai adalah (kode_kul, nim). Penambahan atribut baru (nama_mhs) akan menyebabkan adanya ketergantungan fungsional yang baru yaitu nim > nama_mhs. Karena atribut nama_mhs ini hanya memiliki ketergantungan parsial pada *primary key* secara utuh (hanya tergantung pada nim, padahal nim hanya bagian dari *primary key*).

Bentuk normal keduan ini dianggap belum memadai karena meninjau sifat ketergantungan atribut terhadap *primary key*.

**d. Bentuk normal tahap ketiga (3NF)**

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan relasi X > A, dimana A mewakili semua atribut unggal di dalam tabel yang tidak ada di dalam X, maka :

- X haruslah superkey pada tabel tersebut
- Atau A merupakan bagian dari primary key pada tabel tersebut

Misalkan pada tabel mahasiswa, atribut alamat_mhs dipecah kedalam alamat_jalan, alamat_kota dan kode_pos. Bentuk ini tidak memenuhi 3NF, karena terdapat ketergantungan fungsional baru yang muncul pada tabel tersebut, yaitu :

alamat_jalan nama_kota > kode_pos

dalam hal ini (alamat_jalan, nama_kota) bukan superkey sementara kode_pos juga bukan bagian dari primery key pada tabel mahasiswa. Jika tabel mahasiswa didekomposisi menjadi tabel mahasiswa dan tabel alamat,

maka telah memenuhi 3NF. Hal itu dapat dibuktikan dengan memeriksa dua ketergantungan fungsional pada tabel alamat tersebut, yaitu :

alamat_jalan nama_kota > kode_pos

kode_pos > nama_kota

ketergantungan fungsional yang pertama tidak melanggar 3NF, karena (alamat_jalan, nama_kota) merupakan superkey (sekaligus sebagai primary key) dari tabel alamat tersebut. Demikian juga dengan ketergantungan fungsional yang kedua meskipun (kode_pos) bukan merupakan superkey, tetapi nama_kota merupakan bagian dari primary key dari tabel alamat. Karena telah memenuhi 3NF, maka tabel tersebut tidak perlu didekomposisi lagi.

**e. Bentuk normal tahap keempat dan kelima**

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk 4NF berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergangungan fungsional. Adapun bentuk 5NF merupakan nama lain dari *project join normal form* (PJNF).

**f. Boyce Code Normal Form (BCNF)**

- Memenuhi 1NF
- Relasi harus bergantung fungsi pada atribut superkey.

Kesimpulan Normalisasi adalah sebagai berikut: untuk memperbaiki /membangun data dengan model data relasional dan secara umum lebih tepat dikoneksikan  dengan data model logika.Suatu pengelompokan data kedalam bentuk tbel atau relasi atau file untuk menyatakan entitas dan hubungan mereka,sehingga terwujud satu bentuk basis data yang mudah untuk dimodifikasi