BAB 2

LANDASAN TEORI

2.1 Logika *Fuzzy*

2.1.1 Pendahuluan

Titik awal dari konsep modern mengenai ketidakpastian adalah paper yang dibuat oleh Lofti A Zadeh, di mana Zadeh memperkenalkan teori yang memiliki objek-objek dari himpunan *Fuzzy* yang memiliki batasan yang tidak presisi dan keanggotaan dalam himpunan *Fuzzy*, dan bukan dalam bentuk logika benar (*true*) atau salah (false), tapi dinyatakan dalam derajat (degree). Konsep seperti ini disebut dengan *Fuzziness* dan teorinya dinamakan *Fuzzy Set Theory*.

Fuzziness dapat didefinisikan sebagai logika kabur berkenaan dengan semantik dari suatu kejadian, fenomena atau pernyataan itu sendiri. Seringkali ditemui dalam pernyataan yang dibuat oleh seseorang, evaluasi dan suatu pengambilan keputusan. Sebagai contoh:

1. Manajer pergudangan mengatakan pada manajer produksi seberapa banyak persediaan barang pada akhir minggu ini, kemudian manajer produksi akan menetapkan jumlah barang yang harus diproduksi esok hari.
2. Pelayan restoran memberikan pelayanan terhadap tamu, kemudian tamu akan memberikan tip yang sesuai atas baik tidaknya pelayanan yang diberikan.
3. Anda mengatakan pada saya seberapa sejuk ruangan yang anda inginkan, saya akan mengatur putaran kipas yang ada pada ruangan ini.

Ada beberapa alasan mengapa orang menggunakan logika *Fuzzy*,

antara lain:

1. Konsep logika F*uzzy* 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 pengalamanpengalaman 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.

2.1.2 Perbedaan Himpunan *Fuzzy* dengan Himpunan Pasti (*crisp*)

Pada himpunan pasti (*crisp*) nilai keanggotaan suatu item *x* dalam suatu himpunan A, yang sering ditulis dengan μA[*x*], memiliki 2 kemungkinan, yaitu:

- Satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan, atau
- Nol (0), yang berarti bahwa suatu item tidak menjadi anggota suatu himpunan.

Contoh :

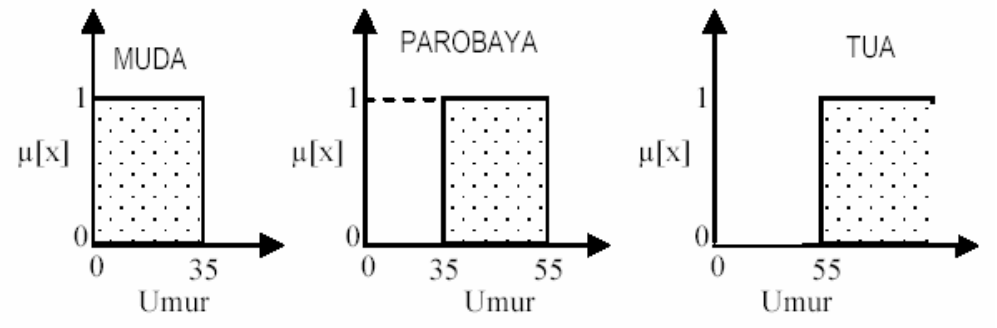
Misalkan variabel umur dibagi menjadi 3 kategori, yaitu:

MUDA umur < 35 tahun

PAROBAYA 35 ≤ umur ≤ 55 tahun

TUA umur > 55 tahun

Nilai keanggotaan secara grafis, himpunan MUDA, PAROBAYA, dan TUA ini dapat dilihat pada gambar 2.1



Gambar 2.1 Himpunan Muda, Parobaya, dan Tua

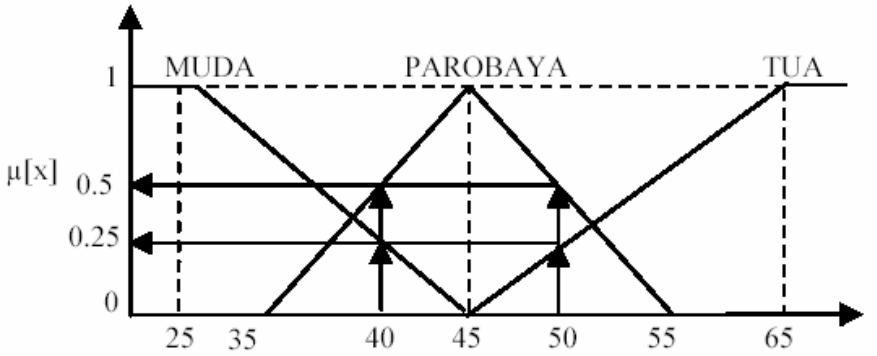
Pada Gambar 2.1, dapat dijelaskan bahwa:

- Apabila seseorang berusia 34 tahun, maka ia dikatakan MUDA (μMUDA[34] = 1);
- Apabila seseorang berusia 35 tahun, maka ia dikatakan TIDAK MUDA (μMUDA[35] = 0);
- Apabila seseorang berusia 35 tahun kurang 1 hari, maka ia dikatakan TIDAK PAROBAYA (μPAROBAYA[35 th – 1 hari] = 0).

Berdasarkan contoh diatas bisa dikatakan pemakaian himpunan crisp untuk menyatakan umur sangat tidak adil, adanya perubahan sedikit saja pada suatu nilai mengakibatkan perbedaan kategori yang cukup signifikan.

Himpunan *Fuzzy* digunakan untuk mengantisipasi hal tersebut. Seseorang dapat masuk dalam 2 himpunan yang berbeda, MUDA dan PAROBAYA,

PAROBAYA dan TUA, dsb. Seberapa besar eksistensinya dalam himpunan tersebut dapat dilihat pada nilai keanggotaan-nya. Gambar 2.2 menunjukkan himpunan *Fuzzy* untuk variable umur.



Gambar 2.2 Himpunan *Fuzzy* untuk variabel umur

Pada Gambar 2.2, dapat dilihat bahwa:

- Seseorang yang berumur 40 tahun, termasuk dalam himpunan MUDA dengan μMUDA[40]=0,25; namun dia juga termasuk dalam himpunan PAROBAYA dengan μPAROBAYA[40]=0,5.
- Seseorang yang berumur 50 tahun, termasuk dalam himpunan MUDA dengan μTUA[40]=0,25; namun dia juga termasuk dalam himpunan PAROBAYA dengan μPAROBAYA[50]=0,5.

Kalau pada himpunan *crisp*, nilai keanggotaan hanya ada 2 kemungkinan, yaitu 0 atau 1, pada himpunan *Fuzzy* nilai keanggotaan terletak pada rentang 0 sampai 1. Apabila *x* memiliki nilai keanggotaan *Fuzzy* μA[*x*]=0 berarti *x* tidak menjadi anggota himpunan A, demikian pula apabila *x* memiliki nilai keanggotaan *Fuzzy* μA[*x*]=1 berarti *x* menjadi anggota penuh pada himpunan A.

2.1.3 Beberapa Hal yang Perlu Diketahui dalam Sistem Fuzzy

Ada beberapa hal yang perlu diketahui dalam memahami system *Fuzzy*, yaitu:

a. Variable *Fuzzy*

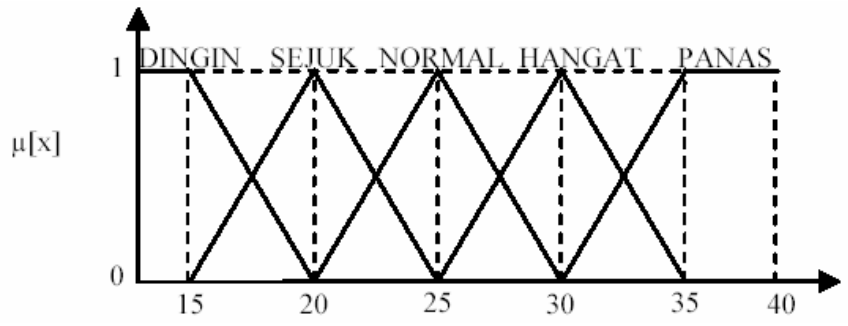
Variabel *Fuzzy* merupakan variabel yang hendak dibahas dalam suatu sistem *Fuzzy*. Contoh: umur, temperatur, permintaan, dsb.

b. Himpunan *Fuzzy*

Himpunan *Fuzzy* merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu variabel *Fuzzy*.

Contoh:

- Variabel umur, terbagi menjadi 3 himpunan *Fuzzy*, yaitu: MUDA, PAROBAYA, dan TUA.
- Variabel temperatur, terbagi menjadi 5 himpunan *Fuzzy*, yaitu: DINGIN, SEJUK, NORMAL, HANGAT, dan PANAS (Gambar 2.3).



Gambar 2.3 Himpunan *Fuzzy* pada variabel temperatur

Himpunan *Fuzzy* memiliki 2 atribut, yaitu:

1. Linguistik, yaitu penamaan grup yang mewakili suatu keadaan atau kondisi tertentu dengan menggunakan bahasa alami, seperti: MUDA, PAROBAYA, TUA.
2. Numeris, yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu variabel seperti: 40, 25, 50, dsb.

c. Semesta Pembicaraan

Semesta pembicaraan adalah keseluruhan nilai yang diperbolehkan untuk dioperasikan dalam suatu variabel *Fuzzy*. Semesta pembicaraan merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai semesta pembicaraan dapat berupa bilangan positif maupun negatif. Adakalanya nilai semesta pembicaraan ini tidak dibatasi batas atasnya

Contoh:

- Semesta pembicaraan untuk variabel umur: [0 +∞].
- Semesta pembicaraan untuk variabel temperatur: [0 40].

d. Domain

Domain himpunan *Fuzzy* adalah keseluruhan nilai yang diizinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *Fuzzy*. Seperti halnya dengan semesta pembicaraan, domain merupakan himpunan bilangan real yang senantiasa naik (bertambah) secara monoton dari kiri ke kanan. Nilai domain dapat berupa bilangan positif maupun negatif.

Contoh domain himpunan *Fuzzy*:

• MUDA = [0, 45]

• PAROBAYA = [35, 55]

• TUA = [45, +∞]

• DINGIN = [0, 20]

• SEJUK = [15, 25]

• NORMAL = [20, 30]

• HANGAT = [25, 35]

• PANAS = [30, 40]

2.1.4 Fungsi Keanggotaan

Fungsi keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke dalam nilai keanggotaannya (sering juga disebut 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. Ada beberapa fungsi yang dapat digunakan:

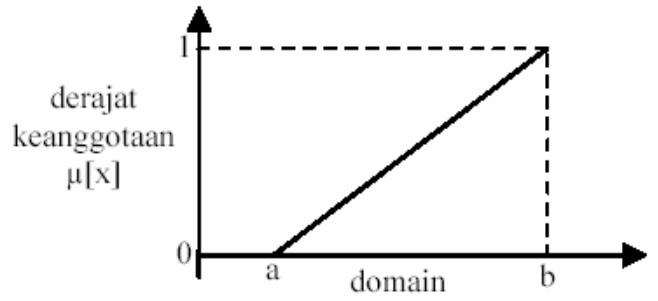
a. Representasi *Linear*

Pada representasi *linear*, pemetaan input ke derajat keanggotaannya digambarkan sebagai suatu garis lurus. Bentuk ini paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas. Ada 2 keadaan himpunan *Fuzzy linear*, yaitu:

1. Representasi *Linear* Naik

Kenaikan himpunan dimulai pada nilai domain yang memiliki derajat keanggotaan nol (0) bergerak ke kanan menuju

ke nilai domain yang memiliki derajat keanggotaan lebih tinggi (Gambar2.4).



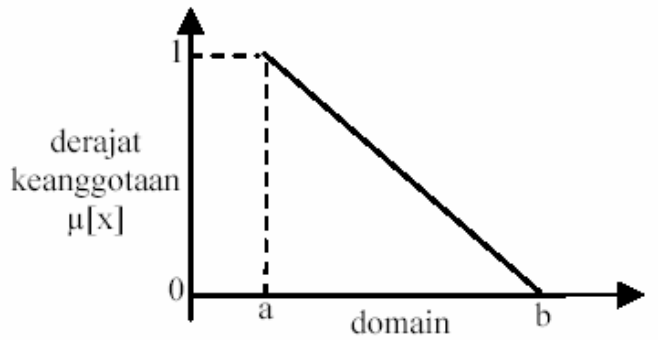
Gambar 2.4 Representasi *Linear* Naik

Fungsi keanggotaan:

$$\mu[x] = \begin{cases} 0; & x \leq a \\ (x-a)/(b-a) & a \leq x \leq b \\ 1; & x \geq b \end{cases}$$

2. Representasi *Linear* Turun

Representasi *linear* turun merupaka kebalikan dari *linear* naik. Garis lurus dimulai dari nilai domain dengan derajat keanggotaan tertinggi pada sisi kiri, kemudian bergerak menurun ke nilai domain yang memiliki derajat keanggotaan lebih rendah (Gambar 2.5).



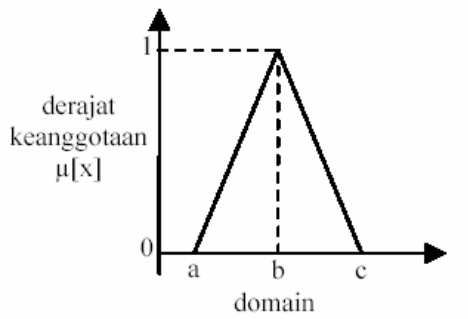
Gambar 2.5 Representasi *Linear* Turun

Fungsi keanggotaan:

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

b. Representasi Kurva Segitiga

Kurva segitiga pada dasarnya merupakan gabungan antara 2 garis linear seperti terlihat pada Gambar 2.6.



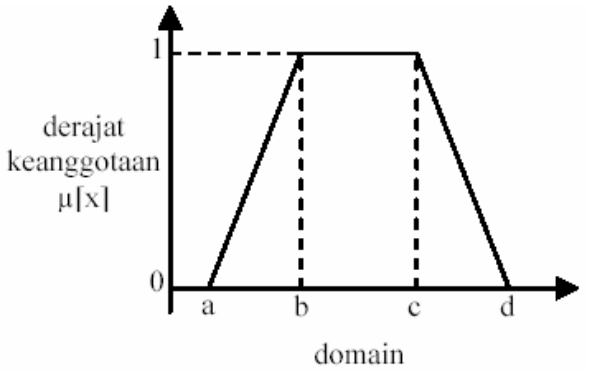
Gambar 2.6 Representasi Kurva Segitiga

Fungsi keanggotaan:

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

c. Representasi Kurva Trapesium

Kurva trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan 1 (Gambar 2.7).



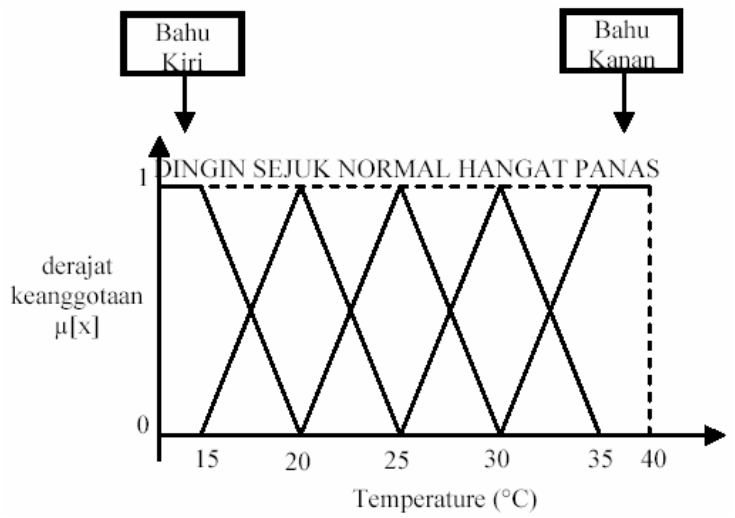
Gambar 2.7 Representasi Kurva Trapesium

Fungsi keanggotaan:

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

d. Representasi Kurva Bentuk Bahu

Daerah yang terletak di tengah-tengah suatu peubah yang direpresentasikan dalam bentuk segitiga, pada sisi kanan dan kirinya akan naik dan turun (misalkan: DINGIN bergerak ke SEJUK bergerak ke HANGAT dan bergerak ke PANAS). Tetapi terkadang salah satu sisi dari peubah tersebut tidak mengalami perubahan. Sebagai contoh, apabila telah mencapai kondisi PANAS, kenaikan temperatur akan tetap berada pada kondisi PANAS. Himpunan *Fuzzy* 'bahu', bukan segitiga, digunakan untuk mengakhiri peubah suatu daerah *Fuzzy*. Bahu kiri bergerak dari benar ke salah, sebaliknya bahu kanan bergerak dari salah ke benar. Gambar 5 menunjukkan peubah TEMPERATUR dengan daerah bahunya.



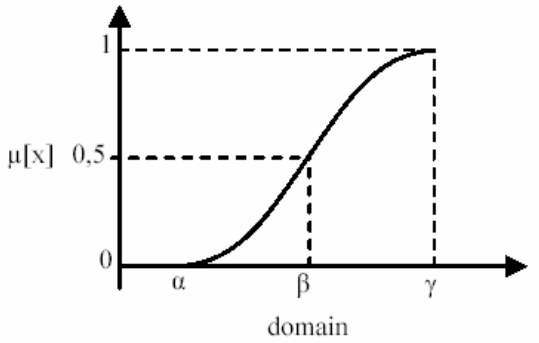
Gambar 2.8 Daerah 'bahu' pada variabel Temperatur

e. Representasi Kurva-S

Kurva-S memiliki nilai kenaikan atau penurunan yang tak *linear*. Ada dua representasi kurva-S, yaitu kurva PERTUMBUHAN dan PENYUSUTAN. Kurva-S didefinisikan menggunakan 3 parameter, yaitu: nilai keanggotaan nol (α), nilai keanggotaan lengkap (γ), dan titik *infleksi* atau *crossover* (β) yaitu titik yang memiliki domain 50% benar.

1. Representasi Kurva-S PERTUMBUHAN

Kurva-S PERTUMBUHAN akan bergerak dari sisi paling kiri dengan nilai keanggotaan nol (0) ke sisi paling kanan dengan nilai keanggotaan satu (1). Fungsi keanggotaannya akan bertumpu pada 50% nilai keanggotaannya yang sering disebut titik *infleksi* (Gambar 2.9).



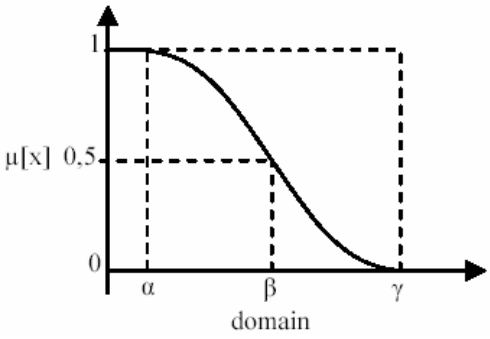
Gambar 2.9 Karakteristik fungsi kurva-S: PERTUMBUHAN

Fungsi keanggotaan:

$$S(x;\alpha,\beta,\gamma)=\begin{cases}0; & x\le\alpha\\ 2((x-\alpha)/(\gamma-\alpha))^2 & \alpha\le x\le\beta\\ 1-2((\gamma-x)/(\gamma-\alpha))^2 & \beta\le x\le\gamma\\ 1 & x\ge\gamma\end{cases}$$

2. Representasi Kurva-S PENYUSUTAN

Kurva-S PENYUSUTAN merupakan kebalikan dari Kurva-S PERTUMBUHAN. Nilai keanggotaannya akan bergerak dari sisi kiri dengan nilai keanggotaan satu (1) ke sisi kanan dengan nilai keanggotaan nol (0). Seperti (Gambar 2.10)



Gambar 2.10 Karakteristik fungsi kurva S- PENYUSUTAN

Fungsi keanggotaan:

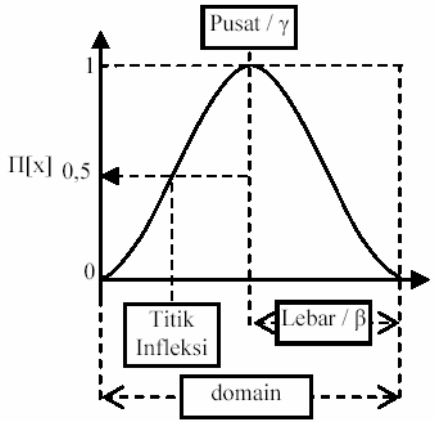
$$S(x;\alpha,\beta,\gamma)=\begin{cases}1 & x \le \alpha \\ 1-2((x-\alpha)/(\gamma-\alpha))^2 & \alpha \le x \le \beta \\ 2((\gamma-x)/(\gamma-\alpha))^2 & \beta \le x \le \gamma \\ 0 & x \ge \gamma\end{cases}$$

f. Representasi Kurva Bentuk Lonceng (*Bell Curve*)

Untuk merepresentasikan himpunan *Fuzzy*, biasanya digunakan kurva bentuk lonceng. Kurva bentuk lonceng ini terbagi atas 3 kelas, yaitu: Kurva π, BETA, dan GAUSS. Perbedaan ketiga kurva ini terletak pada gradiennya.

1. Kurva π

Kurva π berbentuk lonceng dengan derajat keangotaan 1 terletak pada pusat dengan *domain* (γ), dan lebar kurva (β). Seperti terlihat pada (Gambar 2.11)



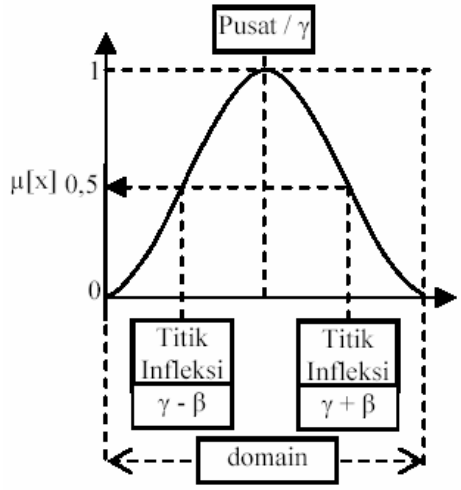
Gambar 2.11 Karakteristik fungsional kurva π

Fungsi keanggotaan:

$$\Pi(x,\beta,\gamma)= \begin{cases} S(x;\gamma-\beta,\gamma-\beta/2,\gamma) & x \le \gamma \\ 1-S(x;\gamma,\gamma+\beta/2,\gamma+\beta) & x > \gamma \end{cases}$$

2. Kurva BETA

Seperti halnya Kurva-π, kurva BETA juga berbentuk lonceng namun lebih rapat. Kurva ini didefinisikan dengan 2 parameter, yaitu nilai pada domain yang menunjukkan pusat kurva dengan *domain*(γ), dan setengah lebar kurva (β). Seperti terlihat pada (Gambar 2.12)



Gambar 2.12 karakteristik fungsional kurva BETA

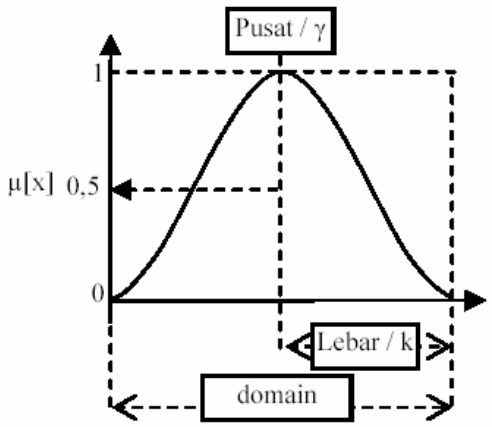
Fungsi keanggotaan:

$$B(x,\gamma,\beta)= \frac{1}{1+\left(\frac{x-\gamma}{\beta}\right)^2}$$

Salah satu perbedaan mencolok Kurva-BETA dari Kurva-π adalah fungsi keanggotaannya akan mendekati nol hanya jika nilai (β) sangat besar.

3. Kurva GAUSS

Kurva GAUSS menggunakan (γ) untuk menunjukkan nilai domain pada pusat kurva, dan (k) yang menunjukkan lebar kurva (Gambar 2.13).



Gambar 2.13 Karakteristik fungsional kurva GAUSS

Fungsi keanggotaan:

$$G(x;k,\gamma)=e^{-k(\gamma-x)^2}$$

2.1.5 Operator Dasar Zadeh untuk Operasi Himpunan *Fuzzy*

Seperti halnya himpunan konvensional, ada beberapa operasi yang didefinisikan secara khusus untuk mengkombinasi dan memodifikasi himpunan *fuzzy*. Nilai keanggotaan sebagai hasil dari operasi 2 himpunan sering dikenal dengan nama *fire strength* atau α- predikat.

Ada 3 operator dasar yang diciptakan oleh Zadeh, yaitu :

1. Operator *AND*

Operator ini berhubungan dengan operasi interseksi pada himpunan. α- predikat. Sebagai hasil operasi dengan operator *AND* diperoleh dengan mengambil nilai keanggotaan terkecil antar elemen pada himpunan-himpunan yang bersangkutan.

$$\mu_{A\cap B} = \min(\mu_A[x], \mu_B[y])$$

2. Operator *OR*

Operator ini berhubungan dengan operasi union pada himpunan. α- predikat sebagai hasil operasi dengan operator *OR* diperoleh dengan mengambil nilai keanggotaan terbesar antar elemen pada himpunan-himpunan yang bersangkutan.

$$\mu_{AUB} = \max(\mu_A[x], \mu_B[y])$$

3. Operator *NOT*

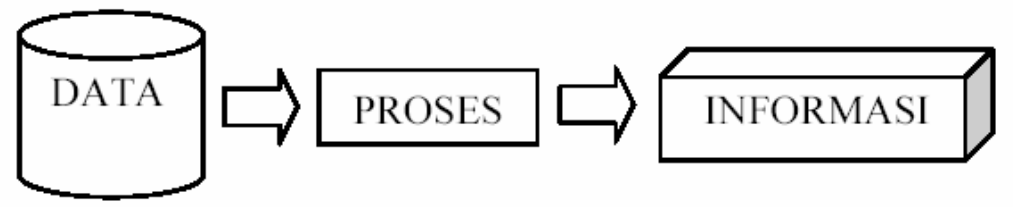
Operator ini berhubungan dengan operasi komplemen pada himpunan. μ-predikat sebagai hasil operasi dengan operator *NOT* diperoleh dengan mengurangkan nilai keanggotaan elemen pada himpunan yang bersangkutan dr 1.

$$\mu_{A'} = 1 - \mu_A[x]$$

2.2 Database

Dalam arti khusus database adalah sekumpulan informasi yang diatur agar mudah dicari. Dalam arti umum database adalah sekumpulan data yang di proses dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan tepat, yang dapat memungkinkan data dapat diakses dengan mudah dan tepat, yang dapat digambarkan sebagai aktivitas dari satu atau lebih organisasi yang berelasi.

Sebagaimana diketahui manajemen modern mengikutsertakan informasi sebagai sumber daya penting uang setara dengan sumber daya manusia, uang, mesin, dan material. Informasi adalah suatu bentuk penyajian data yang melalui mekanisme pemrosesan, berguna bagi pihak tertentu misalnya manajer. Bagi pihak manajer informasi merupakan bahan bagi pengambilan keputusan.



Gambar 2.14 Pengolahan data menjadi informasi

2.3 *Structured Query Language* (SQL)

SQL adalah bahasa standar untuk melakukan *query* kepada database. SQL *independen* terhadap arsitektur database sehingga bisa digunakan pada database apa saja.

Sebuah *query* adalah sebuah ekspresi bahasa yang menggambarkan data yang akan didapatkan kembali dari sebuah database. Dalam hubungannya dengan optimasi *query*, seringkali diasumsikan bahwa *query-query* tersebut

dinyatakan dalam sebuah dasar-dasar isi dan sekumpulan cara orientasi, yang memberikan optimizer pilihan-pilihan diantara *alternative* prosedur-prosedur evaluasi.

Query dapat digunakan pada beberapa keadaan kebanyakan aplikasi nyatanya adalah permintaan-permintaan secara langsung dari user yang memerlukan informasi tentang bentuk maupun isi dari database. Apabila permintaan user terbatas pada sekumpulan *query-query* standart, maka *query-query* tersebut dapat dioptimalisasi secara manual oleh pemrograman prosedur-prosedur pencaharian gabungan dan membatasi *input* dari user pada sebuah ukuran menu.

Tetapi bagaimanapun juga, sebuah sistem optimasi *query* otomatis menjadi penting apabila *query-query* khusus dinyatakan dengan menggunakan bahasa *query* yang digunakan secara umum seperti SQL.

SQL adalah bahasa pemrograman yang digunakan untuk akses ke database. SQL dibuat oleh perusahaan IBM sekitar tahun 1970, pada waktu yang bersamaan dengan diperkenalkannya konsep Relational Database. Setelah mengalami banyak perkembangan, pada masa kini SQL sudah merupakan bahasa yang lazim digunakan dalam dunia database. Bahasa dapat digolongkan bahwa generasi ke empat yang tidak berupa bahasa yang berstruktur dan beraturan seperti C dan Pascal (golongan bahasa generasi ke tiga). Oleh karena itu bahasa SQL mudah dipelajari.

Pernyataan (statement) SQL dapat digolongkan atas tiga golongan, yaitu :

- *Data Definition Language* (DDL) yang mendefinisikan struktur suatu data. Perintah-perintah SQL yang ternasuk DDL antara lain:

1. *CREATE*, untuk membuat tabel.
2. *ALTER*, untuk mengubah (modify) tabel yang telah dibuat, seperti :
    a. Menambah kolom baru.
    b. Mengubah ukuran kolom.
    c. Mengubah aturan-aturan yang berlaku pada suatu kolom.
3. *DROP*, untuk menghapus suatu tabel.

- *Data Manipulation Language* (DML) yang dapat mencari (*query*) dan mengubah (*modify*) suatu tabel. Perintah- perintah yang termasuk dalam DML antara lain :
    1. *SELECT*, untuk membaca (*query*) isi tabel.
    2. *INSERT*, untuk memasukkan data ke tabel.
    3. *UPDATE*, untuk mengubah isi tabel.
    4. *DELETE*, untuk menghapus isi dari tabel.
- *Data Control Language* (DCL) yang mengatur hak-hak (*privilege*) untuk seorang pemakai database.

Semua bahasa pemrograman mempunyai aturan gramatika. Beberapa aturan yang perlu diperhatikan pada saat menulis bahasa SQL adalah :

1. Semua pernyataan SQL ditutup dengan tanda titik-koma (;).
2. Perintah SQL dapat ditulis dalam suatu baris atau dipisah- pisah dalam beberapa baris agar mudah untuk dibaca.

3. SQL tidak membedakan huruf besar ataupun kecil (no case sensitive) untuk penulisan perintah-perintah SQL (seperti *CREATE*, *ALTER*, *SELECT*), nama tabel dan nama kolom.
4. Isi dari tabel peka terhadap huruf besar atau kecil (*case sensitive*), tergantung bagaimana isi tabel dimasukkan. Hal ini penting dalam perhatian waktu membuat pernyataan *query* SQL mencari isi tabel.
5. Pernyataan SQL harus ditulis menurut sintak tertentu.
6. Pernyataan SQL dapat diberikan baris komentar untuk dokumentasi serta menjelaskan maksud pemrograman.

2.4 Teknik untuk Mengimplementasikan *Fuziness* ke dalam Database

Pada umumnya ada dua cara untuk memasukkan *fuzziness* ke dalam suatu database, yaitu :

1. *Fuzzy Query Database.*
2. *Fuzzy Database.*

2.4.1 *Fuzzy Query Database*

Fuzzy query database adalah membuat suatu *fuzzy query* terhadap suatu database klasik. Artinya, kita membuat suatu aplikasi yang dapat menangani suatu *query* di mana dalam *query* tersebut terdapat variabel-variabel yang bernilai *fuzzy* atau dengan lain *query* tersebut memiliki variabel-variabel linguistik. Sedangkan data yang ada pada database yang diakses tersebut adalah data yang *crisp*.

2.4.2 *Fuzzy Database*

Fuzzy database adalah teknik untuk memasukkan informasi fuzzy ke dalam database. Maksudnya kita memasukkan informasi-informasi yang memiliki nilai fuzzy ke dalam database.

2.5 Rekayasa Perangkat Lunak

2.5.1 Pengertian Perangkat Lunak

Pengertian perangkat lunak adalah:

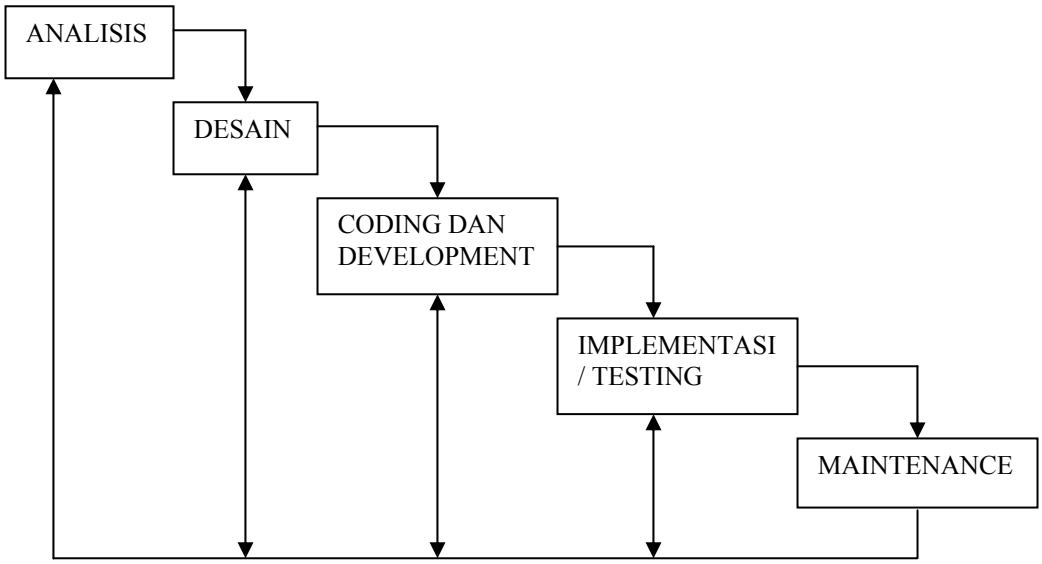
- Instruksi pada kompuer yang bila dijalankan akan memberikan fungsi dan hasil yang diinginkan.
- Dokumen yang menjelaskan operasi penggunaan program.
- Struktur data yang menjadikan program dapat memanipulasi suatu informasi. (Pressman 1992, 45)

Dengan demikian dapat disimpulkan bahwa rekayasa perangkat lunak adalah suatu instruksi, dokumen, dan struktur data yang menyediakan logika dan prosedur yang diinginkan.

2.5.2 Model *Waterfall*

Model *waterfall* merupakan model yang digunakan penulis dalam penelitian. Model ini merupakan sebuah pendekatan kepada pengembangan perangkat lunak yang sistematik dan sekuensial yang dimulai dengan analisis, desain, pengembangan, pengujian, dan pemeliharaan terhadap sistem yang akan dibuat.

Menurut pressman terdapat enam tahapan dalam model Waterfall, seperti yang terdapat pada gambar dibawah ini:



Gambar 2.15 Model Waterfall

Penjelasan tahapan dalam model *Waterfall* adalah sebagai berikut:

- Analisis Kebutuhan

  Proses pengumpulan kebutuhan diintensifkan dan difokuskan, khususnya pada perangkat lunak. Tujuan tahap ini adalah untuk mengetahui kebutuhan piranti lunak, sumber informasi piranti lunak, fungsi-fungsi yang dibutuhkan, kemampuan piranti lunak dan antarmuka piranti lunak tersebut.

- Desain

  Proses desain merupakan representasi kebutuhan ke bentuk perangkat lunak yang dapat dinilai kualitasnya sebelum dilakukan pengkodean. Tahap ini meliputi perancangan struktur data, perancangan

arsitektur piranti lunak, perancangan rincian prosedur dan perancangan user interface.

- Pengembangan

  Desain harus dapat diterjemahkan ke dalam bentuk bahasa pemrograman yang bisa dibaca.

- Implementasi dan Pengujian

  Setelah program aplikasi selesai dikode, program akan diujicobakan dan juga dilakukan pengujian. Pengujian dilakukan secara menyeluruh hingga semua perintah dan fungsi telah diuji sampai output yang dihasilkan oleh program sesuai dengan yang diharapkan.

- Pemeliharaan

  Pemeliharaan perangkat lunak dilakukan karena sering terjadinya perubahan atau peningkatan fungsi piranti lunak. Hal ini sesuai permintaan pemakai, maka pirati lunak yang telah selesai dibuat perlu dipelihara agar dapat mengantisipasi permintaan pemakai terhadap fungsi-fungsi baru. Bila terjadi perubahan berarti membalikkan tahapan ke tahapan yang lebih awal.