BAB 2
LANDASAN TEORI

2.1 *Artificial Intelligence* (AI)

Definisi kecerdasan buatan atau *artificial intelligent* yang paling tepat untuk saat ini adalah *Acting rationally* dengan pendekatan *Rational agent*. Hal ini berdasarkan pemikiran bahwa komputer bisa melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional berdasarkan hasil penalaran tersebut [8].

2.2 Kecerdasan buatan pada *Game*

Kecerdasan buatan atau AI merupakan kegiatan membuat komputer agar dapat berpikir dan mengerjakan kegiatan yang dapat dilakukan oleh manusia maupun binatang. Saat ini dapat ditemukan program komputer yang memiliki kemampuan menangani masalah seperti *aritmatic*, *sorting, searching*[8]. Bahkan komputer juga dapat bermain beberapa *board game* seperti catur lebih baik dari pada manusia. Namun masih banyak hal yang tidak dapat dilakukan dengan baik oleh komputer. Seperti, mengenali wajah, berbicara bahasa manusia, menentukan sendiri apa yang harus dilakukan, dan bertingkah kreatif. Hal itu semua merupakan domain dari AI untuk mencoba menentukan algoritma apa yang dibutuhkan untuk memenuhi kebutuhan diatas.

Dalam bidang akademik, beberapa peneliti AI termotivasi oleh filosofi, yaitu memahami alam pikiran dan alam kecerdasan dan membangun program untuk memodelkan bagaimana proses berpikir. Beberapa juga termotivasi oleh *psychology*, bertujuan untuk memahami mekanisme otak manusia dan proses mental. Dan lainya termotivasi oleh *engineering*, dengan tujuan membangun algoritma untuk melakukan kegiatan seperti manusia atau hewan. Dalam pembangunan *Game*, umumnya akan cenderung hanya pada sisi *engineering* yang bertujuan membangun algoritma yang dapat membuat *Game* karakter mengerjakan kegiatan seperti yang dilakukan manusia atau binatang [8].

Kecerdasan buatan dapat dibangun dengan menggunakan metode-metode

tertentu, salah satu metode dalam kecerdasan buatan adalah metode logika *fuzzy*.

2.3 Logika *Fuzzy*

Fuzzy secara bahasa diartikan sebagai kabur atau samar-samar. Suatu nilai dapat bernilai besar atau salah secara bersamaan. Dalam *fuzzy* dikenal derajat keanggotaan yang memiliki rentang nilai 0 (nol) hingga 1(satu). Berbeda dengan himpunan tegas yang memiliki nilai 1 atau 0 (ya atau tidak).

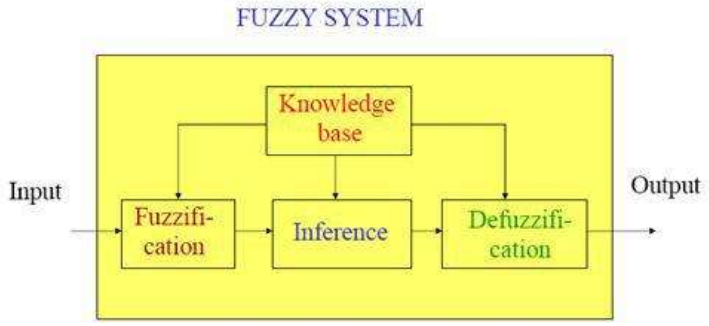
Logika *Fuzzy* merupakan suatu logika yang memiliki nilai kekaburan atau kesamaran (*fuzzyness*) antara benar atau salah. Dalam teori logika *fuzzy* suatu nilai bisa bernilai benar atau salah secara bersama. Namun berapa besar keberadaan dan kesalahan suatu tergantung pada bobot keanggotaan yang dimilikinya. Logika *fuzzy* memiliki derajat keanggotaan dalam rentang 0 hingga 1 [7]. Berbeda dengan logika digital yang hanya memiliki dua nilai 1 atau 0. Logika *fuzzy* digunakan untuk menterjemahkan suatu besaran yang diekspresikan menggunakan bahasa (*Linguistic*), misalkan besaran kecepatan laju kendaraan yang diekspresikan dengan pelan, agak cepat, cepat, dan sangat cepat. Dan logika *fuzzy* menunjukan sejauh mana suatu nilai itu benar dan sejauh mana suatu nilai itu salah. Tidak seperti logika klasik (*crisp*)/tegas, suatu nilai hanya mempunyai 2 kemungkinan yaitu merupakan suatu anggotan himpunan atau tidak. Derajat keanggotaan 0 (nol) artinya nilai bukan merupakan anggota himpunan dan 1 (satu) berarti nilai tersebut adalah anggota himpunan.

Logika *fuzzy* adalah suatu cara yang tepat untuk memetakan suatu ruang *input* kedalam suatu ruang *output*, yang mempunyai nilai *continued*. *Fuzzy* dinyatakan dalam derajat dari suatu keanggotaan dan derajat dari kebenaran. Oleh sebab itu sesuatu dapat dikatakan sebagian benar dan sebagian salah pada waktu yang sama [9]. Logika *Fuzzy* memungkinkan nilai keanggotaan antara 0 dan 1, tingkat keabuan dan juga hitam dan putih, dan dalam bentuk *linguistic*, konsep tidak pasti seperti "sedikit", "lumayan" dan "sangat" [1]. Kelebihan dari teori logika *fuzzy* adalah kemampuan dalam proses penalaran secara bahasa (*Linguistic reasoning*). Sehingga dalam perancangannya tidak memerlukan persamaan matematik dari objek yang akan dikendalikan.

Suatu sistem berbasis aturan *fuzzy* yang lengkap terdiri dari tiga komponen utama yaitu:

1. *Fuzzification*
2. *Inference*
3. *Defuzzification*

Pada gambar 2.1 di bawah ini akan dijelaskan langkah-langkah pengambilan keputusan menggunakan logika *fuzzy.*



Gambar 2. 1 langkah-langkah *fuzzy* [3].

1. *Fuzzification* yaitu suatu proses untuk mengubah suatu masukan dari bentuk tegas (*Crisp*) menjadi *fuzzy* (variabel *linguistic*) yang biasanya disajikan dalam bentuk himpunan-himpunan *fuzzy* dengan suatu fungsi kenggotaannya masing-masing,
2. *Inference* yaitu proses melakukan penalaran menggunakan *fuzzy input* dan *fuzzy rules* yang telah ditentukan sehingga menghasilkan *fuzzy output*. pada umumnya aturan-aturan fuzzy dinyatakan dalam bentuk “IF THEN” yang merupakan inti dari relasi fuzzy.
3. *Defuzzification* yaitu mengubah fuzzy *output* menjadi *Crisp* value (nilai pasti) berdasarkan fungsi keanggotaan yang telah ditentukan. proses pengubahan data-data fuzzy tersebut menjadi data-data numerik yang dapat dikirimkan ke peralatan pengendalian.

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*.

Contoh: umur, temperatur, permintaan, dsb.

2. Himpunan *fuzzy*

Himpunan fuzzy merupakan suatu grup yang mewakili suatu kondisi atau keadaan tertentu dalam suatu *variable fuzzy* [11].

Himpunan *fuzzy* memiliki 2 atribut, yaitu:

1) *Linguistic*, yaitu penamaan suatu 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.

Himpunan *fuzzy* merupakan suatu pengembangan lebih lanjut tentang konsep himpunan dalam matematika. Himpunan *fuzzy* adalah rentang nilai-nilai, masing-masing nilai mempunyai derajat keanggotaan antara 0 sampai dengan 1. Suatu himpunan *fuzzy* $\tilde{A}$ dalam semesta pembicaraan U dinyatakan dengan fungsi keanggotaan $\mu\tilde{A}$, yang nilainya berada dalam interval [0,1], dapat dinyatakan dengan:

$$\mu\tilde{A} : U \rightarrow [0,1] \qquad \textbf{(2.1)}$$

Himpunan *fuzzy* $\tilde{A}$ dalam semesta pembicaraan U biasa dinyatakan sebagai sekumpulan pasangan elemen u (u anggota U) dan derajat keanggotaannya dinyatakan sebagai berikut:

$$\tilde{A} = \{(u, \mu\tilde{A}(u) \mid u \in U\}. \qquad \textbf{(2.2)}$$

Cara tradisional untuk menyatakan apakah sebuah obyek merupakan anggota sebuah himpunan atau tidak adalah dengan menggunakan fungsi karakteristik (kadang-kadang disebut juga dengan fungsi diskriminasi). Jika sebuah obyek merupakan anggota dari sebuah himpunan maka fungsi karakteristiknya 1. Jika sebuah obyek bukan merupakan anggota dari sebuah

himpunan maka fungsi karakteristiknya 0. Fungsi karakteristik dapat didefinisikan menggunakan pemetaan fungsional sebagai berikut:

$$\mu a(X) : X \rightarrow \{0.1\} \quad \textbf{(2.3)}$$

yang menyatakan bahwa fungsi karakteristik memetakan himpunan *universal* X ke himpunan yang terdiri dari 0 dan 1. Himpunan dimana fungsi ini berlaku disebut himpunan *Crisp*.

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

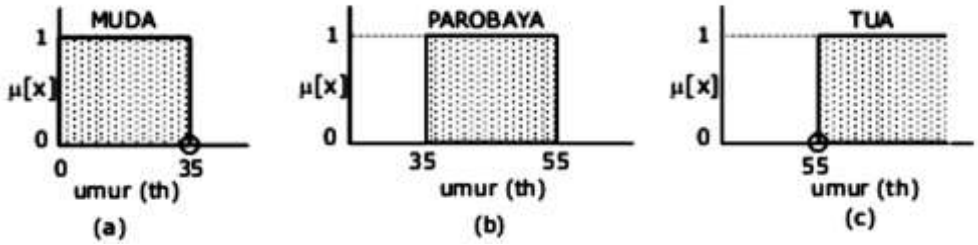
1) satu (1), yang berarti bahwa suatu item menjadi anggota dalam suatu himpunan, atau
2) nol (0), yang berarti bahwa suatu item tidak menjadi anggota dalam suatu himpunan.

Contoh :

Misalkan variabel umur dibagi menjadi 3 kategori, yaitu:

1) MUDA umur < 35 tahun
2) PAROBAYA 35 ≤ umur ≤ 55 tahun
3) TUA umur > 55 tahun

Nilai keanggotaan secara grafis, himpunan MUDA, PAROBAYA dan TUA ini dapat. dilihat pada Gambar 2.2.



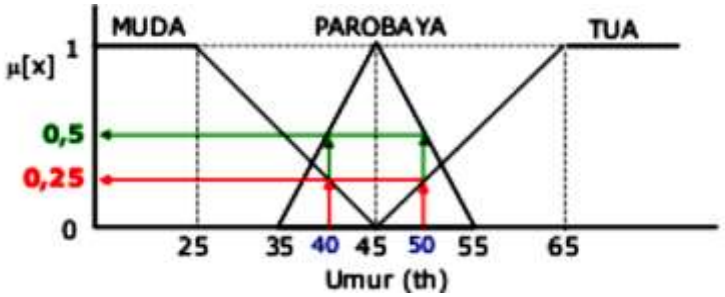
Gambar 2. 2 Himpunan: MUDA, PAROBAYA, dan TUA

Pada Gambar 2.2, dapat dilihat bahwa:

1) Apabila seseorang berusia 30 tahun, maka ia dikatakan MUDA (μMUDA[30] =1);

2) Apabila seseorang berusia 40 tahun, maka ia dikatakan PAROBAYA (μPAROBAYA[40]=1);
3) Apabila seseorang berusia 60 tahun, maka ia dikatakan TUA (μTUA[60]=1);

Dari sini bisa dikatakan bahwa pemakaian himpunan *Crisp* untuk menyatakan umur sangat tidak adil, adanya perubahan kecil 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 keanggotaannya.



Gambar 2. 3 Menunjukkan himpunan *fuzzy* untuk variabel umur

Pada Gambar 2.3 dapat dilihat bahwa:

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

Bila 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. Terkadang kemiripan antara keanggotaan *fuzzy* dengan probabilitas menimbulkan kerancuan. Keduanya memiliki nilai pada interval [0,1], namun interpretasi nilainya sangat berbeda antara kedua kasus tersebut.

Keanggotaan *fuzzy* memberikan suatu ukuran terhadap pendapat atau keputusan, sedangkan probabilitas mengindikasikan proporsi terhadap keseringan suatu hasil bernilai benar dalam jangka panjang. Misalnya, jika nilai keanggotaan suatu himpunan *fuzzy* MUDA adalah 0,9; maka tidak perlu dipermasalahkan berapa seringnya nilai itu diulang secara individual untuk mengharapkan suatu hasil yang hampir pasti muda. Di lain pihak, nilai probabilitas 0,9 muda berarti 10% dari himpunan tersebut diharapkan tidak muda.

3. 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: $[0 + \infty)$

4. Domain himpunan fuzzy

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

Contoh domain himpunan *fuzzy*:

1) MUDA = [0 45]
2) PABOBAYA = [35 55]
3) TUA = $[45 +\infty)$

5. Fungsi Keanggotaan

Fungsi Keanggotaan (*membership function*) adalah suatu kurva yang menunjukkan pemetaan titik-titik input data ke kedalam nilai keanggotaannya (sering juga disebut dengan derajat keanggotaan) yang memiliki interval antara 0 sampai 1. Salah satu cara yang dapat digunakan untuk mendapatkan mendapatkan nilai keanggotaan adalah dengan melalui melalui pendekatan fungsi.

Ada beberapa fungsi yang bisa digunakan, yaitu [11]:

1) Representasi Linear
2) Representasi Kurva Segitiga
3) Representasi Kurva Trapesium

Selanjutnya akan dijelaskan mengenai fungsi-fungsi yang dapat digunakan untuk mencari nilai keanggotaan *fuzzy* tersebut.

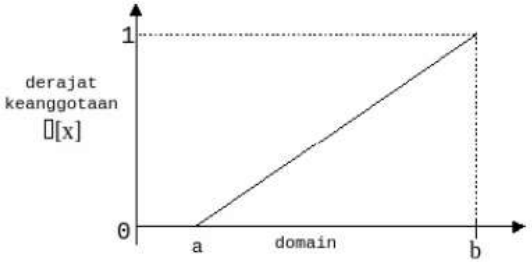
1) Representasi Linear

Pada representasi linear, pemetaan input ke derajat keanggotannya digambarkan sebagai suatu garis lurus. Bentuk ini adalah yang paling sederhana dan menjadi pilihan yang baik untuk mendekati suatu konsep yang kurang jelas.

Terdapat 2 keadaan himpunan *fuzzy* yang linear, yaitu :

A. 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.



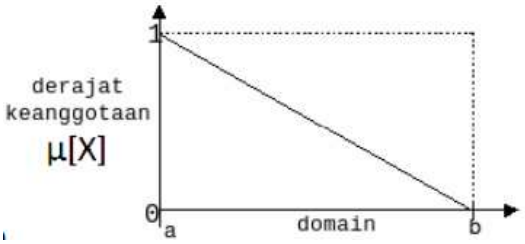
Gambar 2. 4 Representasi linear naik

Fungsi Keanggotaannya dapat di tulis sebagai :

$$\mu[x] = \begin{cases} 0; x \leq a \\ \frac{(x-a)}{b-a}; a \leq x \leq b \\ 1; x \geq b \end{cases} \quad \textbf{(2.4)}$$

B. Representasi Linear Turun

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.



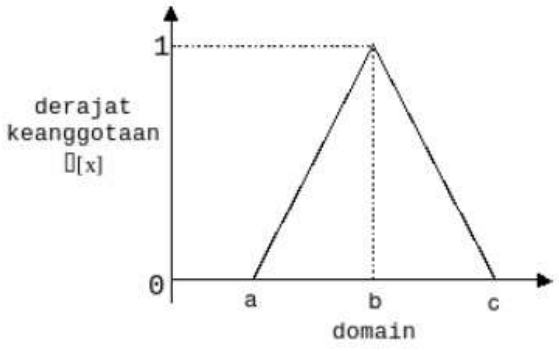
Gambar 2. 5 Representasi linear turun

Fungsi Keanggotaannya dapat di tulis sebagai :

$$\mu[x] = \begin{cases} \frac{b-x}{b-a}; a \leq x \leq b \\ 0; x \geq b \end{cases} \quad \textbf{(2.5)}$$

2) Representasi Kurva Segitiga

Kurva Segitiga pada dasarnya merupakan gabungan antara 2 garis (linear) seperti terlihat pada Gambar 2.6.



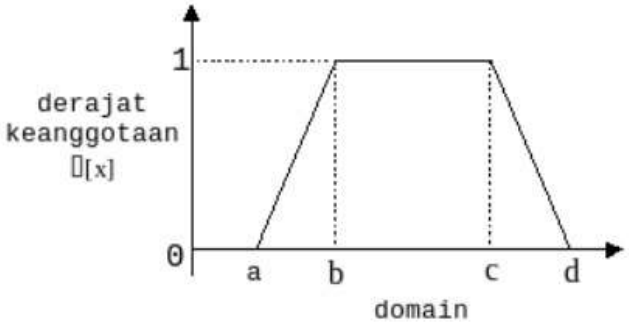
Gambar 2. 6 Representasi kurva segitiga

Fungsi Keanggotaannya dapat di tulis sebagai :

$$\mu[x] = \begin{cases} 0; x \leq a, OR, x \geq c \\ (x-a)/(b-a)\ ; a \leq x \leq b \\ (c-x)/(c-b)\ ; b \leq x \leq c \end{cases} \quad \textbf{(2.6)}$$

3) Representasi Kurva Trapesium

Kurva Trapesium pada dasarnya seperti bentuk segitiga, hanya saja ada beberapa titik yang memiliki nilai keanggotaan1.



Gambar 2. 7 Representasi kurva segitiga

Fungsi Keanggotaannya dapat di tulis sebagai :

$$\mu[x] = \begin{cases} 0; x \leq a, OR, x \geq d \\ (x-a)/(b-a)\ ; a \leq x \leq b \\ 1; b \leq x \leq c \\ (d-x)/(d-c)\ ; c \leq x \leq d \end{cases} \quad \textbf{(2.7)}$$

2.3.1 Sistem Inferensi *fuzzy / (Fuzzy Inference System/FIS)*

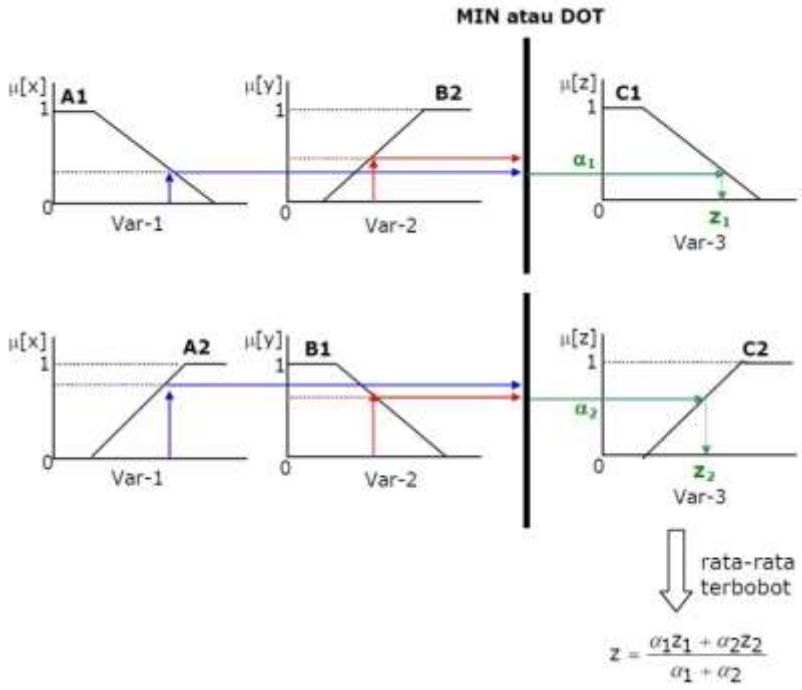
Sistem Inferensi *Fuzzy* (*Fuzzy Inference System*/FIS) disebut juga *fuzzy inference engine* adalah sistem yang dapat melakukan penalaran dengan prinsip serupa seperti manusia melakukan penalaran dengan nalurinya.

Terdapat beberapa jenis FIS yang dikenal yaitu Tsukamoto, Mamdani, dan Sugeno. FIS yang paling mudah dimengerti, karena paling sesuai dengan naluri manusia adalah FIS Mamdani. FIS tersebut bekerja berdasarkan kaidah-kaidah *linguistic* dan memiliki algoritma *fuzzy* yang menyediakan sebuah aproksimasi untuk dimasuki analisa matematik.

Berikut ini akan dijelaskan beberapa jenis FIS yang umum di gunakan dalam *fuzzy logic* :

2.3.1.1 Metode Tsukamoto

Pada Metode Tsukamoto, setiap konsekuen pada aturan yang berbentuk IF-Then harus direpresentasikan dengan suatu himpunan *fuzzy* dengan fungsi keanggotaan yang monoton (Gambar 2.8). Sebagai hasilnya, *output* hasil inferensi dari tiap-tiap aturan diberikan secara tegas (*Crisp*) berdasarkan α-predikat (fire strength). Hasil akhirnya diperoleh dengan menggunakan rata-rata terbobot.



Gambar 2. 8 Inferensi dengan menggunakan Metode Tsukamoto [11].

2.3.1.2 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 4 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 3 metode yang digunakan dalam melakukan inferensi sistem *fuzzy*, yaitu:

1) *Max*.
2) *Additive.*
3) *probabilistic*.

1) 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*). Jika semua proposisi telah dievaluasi, maka *output* akan berisi suatu himpunan *fuzzy* yang merefleksikan konstribusi dari tiap-tiap proposisi. Secara umum dapat dituliskan:

$$\mu_{sf}[x_i] \leftarrow \max(\mu_{sf}[x_i], \mu_{kf}[x_i]) \quad \textbf{(2.8)}$$

dengan:

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

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

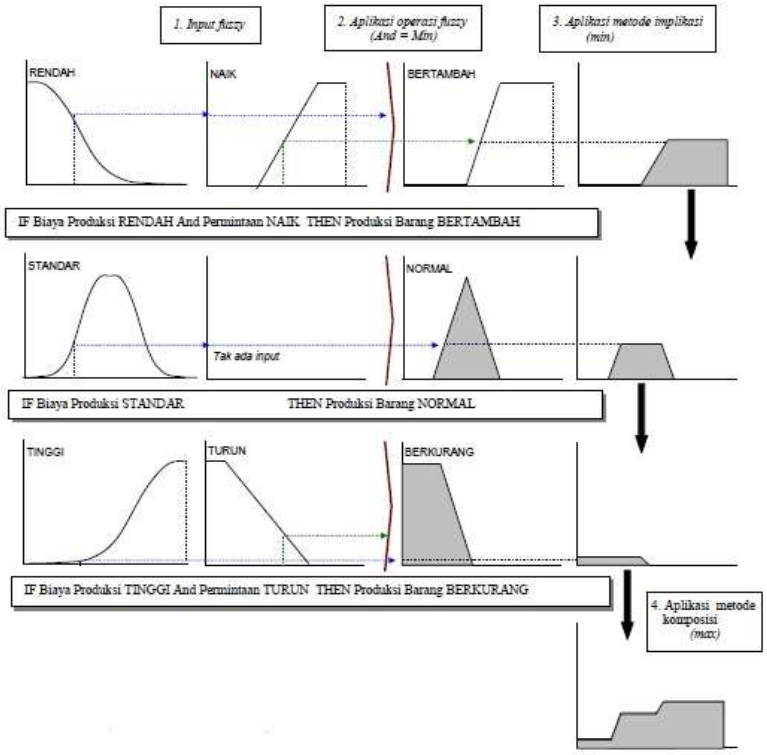
Misalkan ada 3 aturan (proposisi) sebagai berikut:

[R1] IF Biaya Produksi RENDAH And Permintaan NAIK
THEN Produksi Barang BERTAMBAH;

[R2] IF Biaya Produksi STANDAR
THEN Produksi Barang NORMAL;

[R3] IF Biaya Produksi TINGGI And Permintaan TURUN
THEN Produksi Barang BERKURANG;

Apabila digunakan fungsi implikasi *MIN*, maka metode komposisi ini sering disebut dengan nama MAX-MIN atau MIN-MAX atau MAMDANI. Proses inferensi dengan menggunakan metode Max dalam melakukan komposisi aturan seperti terlihat pada Gambar 2.9.



Gambar 2. 9 Komposi Aturan Metode MIN-MAX [11].

2) Metode *Additive* (Sum)

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

$$\mu_{sf}[x_i] \leftarrow \min(1, \mu_{sf}[x_i] + \mu_{kf}[x_i]) \quad \textbf{(2.9)}$$

dengan:

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

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

3) Metode *Probabilistic* (probor)

Pada metode ini, solusi himpunan *fuzzy* diperoleh dengan cara melakukan

product terhadap semua *output* daerah *fuzzy*. Secara umum dituliskan:

$$\mu_{sf}[x_i] \leftarrow (\mu_{sf}[x_i] + \mu_{kf}[x_i]) - (\mu_{sf}[x_i] * \mu_{kf}[x_i]) \quad \textbf{(2.10)}$$

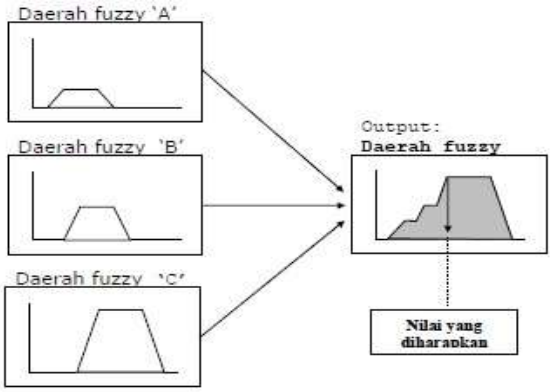
dengan:

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

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

4. Penegasan (de*fuzzification*)

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 2.10.



Gambar 2. 10 Proses Defuzzifikasi [11].

Terdapat beberapa metode defuzzifikasi pada komposisi aturan MAMDANI, antara lain:

1) Metode *Centroid (Composite Moment)*

Pada metode ini, solusi *Crisp* diperoleh dengan cara mengambil titik pusat (z*) daerah *fuzzy*. Secara umum dirumuskan:

$$z^* = \frac{\int_z z\mu(z)dz}{\int_z \mu(z)dz} \rightarrow Untuk\ Semesta\ Kontinu \quad \textbf{(2.11)}$$

$$z^* = \frac{\sum_{j=1}^{n} z_j \mu(z_j)}{\sum_{j=1}^{n} \mu(z_j)} \rightarrow Untuk\ Semesta\ Diskret$$

Metode Defuzzifikasi inilah yang di gunakan dalam pengimplementasian logika fuzzy pada game pasar jajanan ini karena dirasa paling cocok untuk mencari nilai *crisp* untuk kemudian dikonversi kedalam perilaku NPC.

2) Metode Bisektor

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

3) Metode *Mean of Maximum* (MOM)

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

4) Metode *Largest of Maximum* (LOM)

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

5) Metode *Smallest of Maximum* (SOM)

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

2.3.1.3 Metode Sugeno

Penalaran dengan metode SUGENO hampir sama dengan penalaran MAMDANI, hanya saja *output* (konsekuen) sistem tidak berupa himpunan *fuzzy*, melainkan berupa konstanta atau persamaan linear. Metode ini diperkenalkan oleh Takagi- Sugeno pada tahun 1985.

1. Model *Fuzzy* Sugeno Orde-Nol

Secara umum bentuk model *fuzzy* SUGENO Orde-Nol adalah:

IF (x1 is A1) . (x2 is A2) .(x3 is A3) ….(xN is AN) THEN z = k **(2.12)**

dengan Ai adalah himpunan *fuzzy* ke-i sebagai anteseden, dan k adalah suatu konstanta (tegas) sebagai konsekuen.

2. Model *Fuzzy* Sugeno Orde-Satu

Secara umum bentuk model *fuzzy* SUGENO Orde-Satu adalah:

IF (x1 is A1) …(xN is AN) THEN z = p1*x1 + … + pN*xN + q **(2.13)**

dengan Ai adalah himpunan *fuzzy* ke-i sebagai anteseden, dan pi adalah suatu konstanta (tegas) ke-i dan q juga merupakan konstanta dalam konsekuen.

2.4 Game

Game merupakan kata dalam bahasa inggris yang berarti permainan. Permainan adalah sesuatu yang dapat dimainkan dengan aturan tertentu sehingga ada yang menang dan ada yang kalah, biasanya dalam konteks tidak serius atau dengan tujuan *refreshing*. Suatu cara belajar yang digunakan dalam menganalisa interaksi antara sejumlah pemain maupun perorangan yang menunjukkan strategi strategi yang rasional. Teori permainan pertama kali ditemukan oleh sekelompok ahli Matematika pada tahun 1944. Teori itu dikemukakan oleh John von Neumann and Oskar Morgenstern yang berisi:

"Permainan terdiri atas sekumpulan peraturan yang membangun situasi bersaing dari dua sampai beberapa orang atau kelompok dengan memilih strategi yang dibangun untuk memaksimalkan kemenangan sendiri atau pun untuk meminimalkan kemenangan lawan. Peraturan-peraturan menentukan kemungkinan tindakan untuk setiap pemain, sejumlah keterangan diterima setiap pemain sebagai kemajuan bermain, dan sejumlah kemenangan atau kekalahan dalam berbagai situasi"[8].

2.4.1 Pengertian *Game*

Pengertian *Game* menurut beberapa ahli :

1. Menurut Agustinus Nilwan dalam bukunya Pemrograman Animasi dan *Game* Profesional terbitan Elex Media Komputindo, *Game* merupakan permainan komputer yang dibuat dengan teknik dan metode animasi. Jika ingin mendalami pengunaan animasi haruslah memahami pembuatan *Game*. Atau jika ingin membuat *Game*, maka haruslah memahami teknik dan metode animasi, sebab keduanya saling berkaitan.
2. Menurut Clark C. Abt, *Game* adalah kegiatan yang melibatkan keputusan pemain, berupaya mencapai tujuan dengan dibatasi oleh konteks tertentu (misalnya, dibatasi oleh peraturan).

3. Menurut Chris Crawford, seorang computer *Game* designer mengemukakan bahwa *Game*, pada intinya adalah sebuah interaktif, aktivitas yang berpusat pada sebuah pencapaian, ada pelaku aktif *(player),* ada pelaku pasif *(NPC*).
4. Menurut David Parlett, *Game* adalah sesuatu yang memiliki "akhir dan cara mencapainya" : artinya ada tujuan, hasil dan serangkaian peraturan untuk mencapai keduanya.
5. Menurut Roger Caillois, seorang sosiolog Perancis, dalam bukunya yang berjudul Les jeux et les hommes menyatakan *Game* adalah aktivitas yang mencakup karakteristik berikut: *fun* (bebas bermain adalah pilihan bukan kewajiban), *separate* (terpisah), *uncertain, non-productive, governed by rules* (ada aturan), *fictitious* (pura-pura).
6. Menurut Greg Costikyan, *Game* adalah ―sebentuk karya seni di mana peserta, yang disebut Pemain, membuat keputusan untuk mengelola sumberdaya yang dimilikinya melalui benda di dalam *Game* demi mencapai tujuan.

2.4.2 Klasifikasi *Game*

Klasifikasi *game* adalah *genre game* yang didasari interaksi pemainnya, Visualisasi juga menjadi ukuran dari Klasifikasi *game* berikut ini adalah pembagian *genre game* menurut *platform* permainannya [15].

1. *Arcade Games*

   yaitu yang sering disebut *ding-dong* di Indonesia, biasanya berada di daerah / tempat khusus dan memiliki box atau mesin yang memang khusus di design untuk jenis video *Game*s tertentu dan tidak jarang bahkan memiliki fitur yang dapat membuat pemainnya lebih merasa masuk dan menikmati , seperti pistol, kursi khusus, sensor gerakan, sensor injakkan dan stir mobil (beserta transmisinya tentunya).
2. *PC Games*

   yaitu *video Game* yang dimainkan menggunakan *Personal Computers*.
3. *Console Games*

yaitu *video Games* yang dimainkan menggunakan *console* tertentu, seperti *Playstation* 2*, Playstation* 3*, XBOX* 360*,* dan *Nintendo* Wii.

4. *Handheld Games*

   yaitu yang dimainkan di *console* khusus *video Game* yang dapat dibawa kemana-mana, contoh *Nintendo* DS dan Sony PSP.

5. *Mobile Games*

   yaitu yang dapat dimainkan atau khusus untuk *mobile phone* atau *PDA*.

2.5 *Game* berbasis *Desktop*

Desktop Game adalah aplikasi *Game* yang di buat untuk di jalankan oleh desktop komputer dan dikembangkan untuk dijalankan di masing klien. *Database* diletakkan di *server* sedangkan aplikasinya di instal di masing-masing klien. Bahasa pemograman yang digunakan untuk aplikasi tipe ini adalah C#, *javascript* dsb.

Pada *Game* berbasis desktop, aplikasi di bangun dengan menggunakan tool tertentu contohnya unity , kemudian dikompilasi dan Hasilnya dapat langsung digunakan dalam *computer.*

2.6 *Game life-simulation*

Life-Simulation Game adalah cabang dari *genre Game* simulator yaitu, sebuah *genre Games* yang dimana pemainnya bertanggung jawab atas sebuah tokoh atau karakter dalam memenuhi kebutuhan tokoh selayaknya kehidupan nyata, namun dalam ranah *virtual*. Karakter memiliki kebutuhan dan kehidupan layaknya manusia, seperti kegiatan bekerja, bersosialisasi, makan, belanja, dan sebagainya. Biasanya karakter ini hidup dalam sebuah dunia *virtual* yang dipenuhi oleh karakter-karakter yang dimainkan pemain lainnya. Contoh *Game* yang ber-genre *life-simulation* adalah *The Sims* dan *Second Life.*

2.6.1 Kriteria *Game* simulasi

Dari semua jenis permainan yang ada, masing-masing memiliki tingkat kesulitan dan kemudahannya, jika bukan algoritmanya maka akan mudah dalam hal animasinya, akan tetapi *Game*s simulasi bisa disebut sebagai jenis permainan

yang paling sulit, baik algoritma pembuatannya maupun animasinya. Permainan jenis ini juga yang paling membuat pusing dibandingkan dengan permainan jenis lainnya. Algoritmanya sangat sulit sebab harus memperhitungkan semua kejadian dalam kondisi sebenarnya. Berbagai efek animasi yang dibuat tidak cukup bermodalkan ahli grafik dan algoritma saja, tetapi sedikitnya harus mengerti persoalan matematika, teknik dan fisika. Contoh permainan jenis ini adalah Stellar7, F-15 Strike Eagle, Flight Simulator 98, F-14 Tomcat, F-16 Falcon, Jet Fighter [12].

Construction and manajement simulation Game, disingkat CMSs, adalah tipe *Game* simulasi dimana player harus mendirikan, memperluas atau mengelola komunitas atau proyek fiksi.

1. *City-building Game* , merupakan bagian dari economic simulation *Game.* contoh dari *Game* ini adalah *sims city series.*
2. *Economic simulation Game*, biasanya berupa simulasi keadaan ekonomi atau bisnis , dimana *player* mengontrol keadaan ekomomi dari *Game* tersebut contoh *Game* ini adalah *monopoly tycoon*
3. *god Game*, sering tidak mempunyai tujuan akir yang membuat *player* memenangkan *Game* contoh *the sims series*
4. *government simulation Game* , biasa disebut juga *political Game*, masukkan unsur kepolisian, pemerintahan atau politik sebuah negara, bahkan juga peperangan.

2.7 Pasar

Pasar adalah salah satu dari berbagai sistem, institusi, prosedur, hubungan sosial dan infrastruktur dimana usaha menjual barang, jasa dan tenaga kerja untuk orang-orang dengan imbalan uang [4]. Barang dan jasa yang dijual menggunakan alat pembayaran yang sah seperti uang fiat. Kegiatan ini merupakan bagian dari perekonomian. Ini adalah pengaturan yang memungkinkan pembeli dan penjual untuk item pertukaran. Persaingan sangat penting dalam pasar, dan memisahkan pasar dari perdagangan. Dua orang mungkin melakukan perdagangan, tetapi dibutuhkan setidaknya tiga orang untuk memiliki pasar, sehingga ada persaingan

pada setidaknya satu dari dua belah pihak. Pasar bervariasi dalam ukuran, jangkauan, skala geografis, lokasi jenis dan berbagai komunitas manusia, serta jenis barang dan jasa yang diperdagangkan. Beberapa contoh termasuk pasar petani lokal yang diadakan di alun-alun kota atau tempat parkir, pusat perbelanjaan dan pusat perbelanjaan, mata uang internasional dan pasar komoditas, hukum menciptakan pasar seperti untuk izin polusi, dan pasar ilegal seperti pasar untuk obat-obatan terlarang.

Dalam ilmu ekonomi mainstream, konsep pasar adalah setiap struktur yang memungkinkan pembeli dan penjual untuk menukar jenis barang, jasa dan informasi. Pertukaran barang atau jasa untuk uang adalah transaksi. Pasar peserta terdiri dari semua pembeli dan penjual yang baik yang memengaruhi harga nya. Pengaruh ini merupakan studi utama ekonomi dan telah melahirkan beberapa teori dan model tentang kekuatan pasar dasar penawaran dan permintaan. Ada dua peran di pasar, pembeli dan penjual. Pasar memfasilitasi perdagangan dan memungkinkan distribusi dan alokasi sumber daya dalam masyarakat. Pasar mengizinkan semua item yang diperdagangkan untuk dievaluasi dan harga. Sebuah pasar muncul lebih atau kurang spontan atau sengaja dibangun oleh interaksi manusia untuk memungkinkan pertukaran hak (kepemilikan) jasa dan barang.

2.8 Pedagang

Perdagangan adalah semua tindakan yang tujuannya menyampaikan barang untuk tujuan hidup sehari-hari, prosesnya berlangsung dari produsen kepada konsumen. Orang yang pekerjaannya memperjualbelikan barang atas prakarsa dan resiko dinamakan pedagang .

Perdagangan dibedakan atas perdagangan besar dan perdagangan kecil. Dalam perdagangan besar jual beli berlangsung secara besar-besaran. Dalam perdagangan besar, barang tidak dijual/disampaikan langsung kepada konsumen atau pengguna, sedangkan dalam perdagangan kecil, jual beli berlangsung secara kecil-kecilan dan barang dijual langsung kepada konsumen.

Sementara itu, pedagang sendiri jenisnya bermacam-macam. Ada

pedagang keliling, pedagang asongan, pedagang dari pintu ke pintu (door to door), pedangang kios, pedangang kaki lima, grosir (pedagang besar), pedagang supermarket dan sebagainya. Jenis-jenis pedagang ini lazim dibedakan berdasarkan pada cara menawarkan barang dagangannya masing-masing yaitu :

1. Pedagang keliling

   Pedagang keliling adalah pedagang yang menawarkan barang dagangannya dengan cara berkeliling. Berkeliling di sini biasanya dilakukan dari RT ke RT, dari RW ke RW, dari kampung ke kampung, atau dari desa ke desa. Barang yang mereka tawarkan biasanya digendong, dipikul.

2. Pedagang Asongan

   Pedagang asongan adalah pedagang yang menawarkan barang dagangannya dengan cara menempatkannya di kotak kecil yang mudah dibawa dan dipindah-pindahkan. Kotak tersebut biasanya mereka kalungkan di leher seperti tas, dan barang-barang yang mereka tawarkan biasanya berupa rokok, korek api, kembang gula, kertas tisu, kacang, kuaci, buah, dan barang-barang ringan lainnya

3. Pedagang Kaki Lima

   Pedagang kaki lima adalah pedagang yang menawarkan barang dagangannya dengan cara menggelarnya di trotoar atau di tepi jalan yang ramai. Untuk menggelar dagangannya, mereka menggunakan tikar, terpal atau semacam balai-balai. Barang-barang yang mereka tawarkan umumnya berupa sepatu, pakaian, makanan, buah-buahan dan lain – lain.

4. Pedagang Grosir

   Grosir adalah pedagang yang dalam menawarkan barang tidak langsung berhadapan dengan calon pembeli. Pedagang grosir tidak langsung menawarkan barang kepada calon pembeli sebagaimana pedagang eceran, melainkan calon pembelilah yang mendatangi pedagang grosir.

2.9 *Software* pengimplementasian algoritma *fuzzy logic* pada *Game*

Untuk mengiplementasikan *fuzzy logic* pada *game life-simulation* dibutuhkan beberapa software pendukung, Berikut adalah *software's* tersebut.

2.9.1 Unity

Unity *Game* Engine adalah software atau *Game* engine yang digunakan untuk membuat video *Game* berbasis dua atau tiga dimensi dan dapat digunakan secara gratis. Selain untuk membuat *Game*, unity juga dapat digunakan untuk membuat konten yang interaktif lainnya seperti, visual arsitektur dan real-time 3D animasi.

karena unity memiliki berbagai fitur yang sangat berguna bagi perancangan game 3d seperti yang di deskripsikan diatas dan bisa meng-*import* format FBX dari software lainya seperti google skechup dan lain-lain, game engine ini di pilih untuk di gunakan sebagai *software* utama dalam membuat game pasar jajanan ini.

Unity adalah sebuah sebuah tool yang terintegrasi untuk membuat *Game*, arsitektur bangunan dan simulasi. Unity bisa digunakan untuk *Game*s PC dan *Game*s *online*. Untuk *Game*s *online* diperlukan sebuah *plugin,* yaitu Unity Web *Player*, yang sama halnya dengan flash *player* pada *browser.* Bahasa pemrograman yang digunakan bermacam-macam, mulai dari javascript, C#, dan boo. Unity tidak bisa melakukan desain atau modelling, dikarenakan unity bukan merupakan tools untuk mendesain. Banyak hal yang bisa di lakukan di unity, ada fitur audio *reverb zone , particle effect* , *sky box* untuk menambahkan langit, dan masih banyak lagi, dan juga bisa langsung edit texture dari e*ditor* seperti photoshop dll.

Features (Scripting) di dalam unity adalah sebagai berikut (Unity Technologies, 2013): Mendukung 3 bahasa pemrograman, JavaScript, C#, dan Boo. Flexible and EasyMoving, rotating, dan scaling objects hanya perlu sebaris kode. Begitu juga dengan duplicating, removing, dan changing properties. Multi *Platform Game* bisa di deploy di PC, Mac, Wii, iPhone, iPad dan *browser,* android. Visual Properties *Variable*s yang di definisikan dengan *scripts*

ditampilkan pada editor. Bisa digeser, di drag and drop, bisa memilih warna dengan color picker. Berbasis .NET, penjalanan program dilakukan dengan *Open Source .NET platform, Mono.*

2.9.2 Matlab *fuzzy toolkit*

Fuzzy toolkit adalah suatu *tools khusus* yang di sediakan oleh *Software Matlab (matrix laboratory)* dengan *tools* ini kita bisa mensimulasikan perhitungan *fuzzy logic* dengan segala aturanya dengan hasil yang cepat dan akurat, serta bisa langsung mendapatkan keluaran berupa *graphic* hasil perhitungan *fuzzy* dan lain lain.

Matlab adalah sebuah lingkungan komputasi numerikal dan bahasa pemrograman komputer generasi keempat. Dikembangkan oleh The MathWorks, MATLAB memungkinkan manipulasi *matriks,* pem-plot-an fungsi dan data, implementasi algoritma, pembuatan antarmuka pengguna, dan peng-antarmuka-an dengan program dalam bahasa lainnya. Meskipun hanya bernuansa numerik, sebuah kotak kakas (*toolbox*) yang menggunakan mesin simbolik MuPAD, memungkinkan akses terhadap kemampuan aljabar komputer. Sebuah paket tambahan, *Simulink*, menambahkan simulasi grafis multiranah dan Desain Berdasar-Model untuk sistem terlekat dan dinamik.

2.9.3 Google sketchup

Google sketchup sebuah software buatan google yang berfungsi untuk desain grafis,yang dapat menghasilkan berupa gambar 3D. Selain itu software ini sangat ringan daripada software-software lainnya. Walaupun dengan tampilannya yang sederhana, Google SketchUp memungkinkan kita untuk menggambar lebih cepat dan akurat. Program ini merupakan suatu program aplikasi pemodelan 3D yang fleksibel cepat dan dan praktis. Google SketchUp juga Biasa digunakan Untuk mendisain bangunan serta detail-detailnya dengan penampilan 3D yang mudah.

Google SketchUp ini di gunakan untuk mendesain model 3d dari pasar serta seluruh isinya di dalam game Pasar jajanan ini, software ini digunakan karena mudah digunakan dan dapat meng-*export file* yang di buat menjadi *format*

FBX sehingga memungkinkan untuk di gunakan di *software UNITY*.

2.9.4 Iclone

iClone adalah sebuah *software* yang di buat dan di kembangkan oleh reallusion yaitu sebuah *software* real-time animasi 3D untuk aktor digital, lingkungan & efek visual dengan editing drag & drop, yang dirancang untuk design kreativitas dalam siaran, pendidikan produksi & *previzualization*.

Profesional dan pemula dapat mengambil keuntungan dari konten dan interaktivitas cerdas antara aktor, kendaraan dan alat peraga yang membuat animasi dan logic lebih menyenangkan.

Software ini di gunakan untuk membuat model 3d, dan seluruh animasi dari karakter utama dan NPC yang di gunakan sebagai penjual dan pembeli di dalam game Pasar jajanan ini.

2.10 Pemrograman Berorientasi Objek (*Object Oriented Programing*)

OOP adalah konsep yang membagi program menjadi objek-objek yang saling berinteraksi satu sama lain. Objek adalah benda, baik benda yang berwujud nyata maupun benda yang tidak nyata (konsep). Objek adalah kesatuan entitas yang memiliki sifat dan tingkah laku [13].

Dalam kehidupan sehari-hari, objek adalah benda, baik benda berwujud nyata seperti manusia, hewan, mobil, komputer, handphone, pena, ataupun benda yang tidak nyata atau konsep, seperti halnya tabungan bank, sistem antrian, sistem internet banking, dan sebagainya.

Ada 6 keuntungan dalam menggunakan oop yaitu :

1. Alami (*Natural*).
2. Dapat diandalkan (*Reliable*).
3. Dapat digunakan kembali (*Reusable*).
4. Mudah untuk dalam perawatan (*Maintainable*).
5. Dapat diperluas (Extendable).
6. Efisiensi waktu.

Berikut ini beberapa bahasa pemrograman yang sudah menggunakan konsep OOP, adalah :

1. C++.
2. C#.
3. Visual Basic.
4. Java script.

2.10.1 Analisis dan Desain Berorientasi Objek

Pemrograman berorientasi objek bekerja dengan baik ketika dibarengi dengan *object-oriented analysis and design process (OOAD).* Jika membuat program berorientasi objek tanpa OOAD, ibarat membangun rumah tanpa terlebih dahulu menganalisa apa saja yang dibutuhkan oleh rumah itu, tanpa perencanaan tanpa *blueprint*, tanpa menganalisis ruangan apa saja yang diperlukan, berapa besar rumah yang akan dibangun dan sebagainya.

2.10.2 Objek (*Object*)

Orientasi objek merupakan teknik dalam menyelesaikan masalah yang kerap muncul dalam pengembangan perangkat lunak. Teknik ini merupakan titik kulminasi dalam menemukan cara yang efektif dalam membangun sistem dan menjadi metode yang paling banyak dipakai oleh para pengembang perangkat lunak saat ini. Orientasi objek merupakan teknik pemodelan sistem ril yang berbasis objek. Inti dari konsep ini adalah objek yang merupakan model dari sistem nyata.

Objek adalah entitas yang memiliki atribut, karakter dan kadangkala disertai kondisi. Objek merepresentasikan sesuatu sistem nyata seperti siswa, sistem kontrol permukaan sayap pesawat, sensor atau mesin. Objek juga merepresentasikan sesuatu dalam bentuk konsep seperti nasabah bank, merek dagang, pernikahan atau sekedar *listing*. Bahkan bisa juga mengatakan visualisasi seperti, bentuk huruf, histogram, poligon, garis atau lingkaran. Semuanya memiliki fitur atribut (untuk data), *behavior* (*operation* atau *method*), keadaan (memori), identitas dan tanggung jawab. Proses menjabarkan sistem nyata menjadi objek dinamakan abstraksi *(abstraction)*. Abstraksi mengeliminir aspek

yang tidak perlu dalam suatu objek.

2.10.3 Kelas (*Class)*

Kelas adalah penggambaran satu set objek yang memiliki atribut dan *behaviour* yang sama. Kelas mirip tipe data pada pemrograman non objek, tapi lebih komprehensif karena terdapat struktur sekaligus karakteristiknya. Programmer dapat membentuk kelas baru yang lebih spesifik dari kelas *general*-nya. Kelas dan objek merupakan jantung dari pemrograman berorientasi objek. Untuk menghasilkan program jenis ini sangat penting untuk selalu berfikir dalam bentuk objek.

2.10.4 Pembungkusan (*Encapsulation*)

Pembungkusan sebagai penggabungan potongan-potongan informasi dan perilaku-perilaku spesifik yang bekerja pada informasi tersebut, kemudian mengemasnya menjadi apa yang disebut sebagai objek. Dalam perbankan dikenal objek rekening yang memiliki perilaku-perilaku misalnya buka, tutup, penarikan, penyimpanan, ubah nama, ubah alamat, dan sebagainya. Akibatnya, perubahan-perubahan pada sistem perbankan yang berkaitan dengan rekening-rekening dapat secara sederhana diimplementasikan satu kali saja pada objek rekening. Keuntungan lainnya adalah membatasi efek-efek perubahan pada sistem. Misalnya, saat manajemen bank menentukan jika seseorang memiliki rekening pinjaman di bank yang bersangkutan, rekening pinjaman itu harus dapat juga digunakan sebagai sarana bagi penarikan rekening.

2.10.5 Pewarisan *(Inheritance)* dan *Generalization*

Konsep dimana metode dan atau atribut yang ditentukan di dalam sebuah objek kelas dapat diwariskan atau digunakan lagi atau digunakan lagi oleh objek kelas lainnya. Sedangkan *generalization* merupakan teknik dimana atribut dan perilaku yang umum pada beberapa tipe kelas objek, dikelompokkan (atau diabstraksi) ke dalam kelasnya sendiri (dinamakan *supertype*). Atribut dan metode kelas objek *supertype* kemudian diwariskan oleh kelas objek tersebut (dinamakan *subtype*).

2.10.6 Polimorfisme

Polimorfisme berarti suatu fungsionalitas yang diimplementasikan dengan berbagai cara yang berbeda. Pada terminologi berorientasi objek, ini berarti dapat memiliki berbagai implementasi untuk sebagian fungsionalitas tertentu. Sebagai contoh, misalkan pengembang akan mengembangkan sistem berbasis grafis. Saat pengguna mau menggambar sesuatu, misalnya garis atau lingkaran, sistem akan memunculkan perintah gambar. Sistem akan mengenali berbagai bentuk gambar, masing-masing dengan perilakunya sendiri-sendiri. Manfaat dari *polimorfisme* adalah kemudahan pemeliharaannya. Jika perlu menambahkan gambar baru (misalnya segitiga) maka cukup menambahkan fungsi baru (fungsi menggambar segitiga) sedangkan fungsi umumnya (fungsi gambar) tidak mengalami perubahan.

2.11 Bahasa Pemprograman C#

C# merupakan sebuah bahasa pemrograman yang berorientasi objek yang dikembangkan oleh Microsoft sebagai bagian dari inisiatif kerangka *.NET Framework*. C# adalah *Java versi Microsoft*, sebuah bahasa multi flatform yang didesain untuk bisa berjalan di berbagai mesin. C# adalah pemrograman berorientasi Object (OOP). C# memiliki kekuatan bahasa C++ dan portabilitas seperti Java. Fitur-fitur yang diambilnya dari bahasa C++ dan Java adalah desain berorientasi objek, seperti *garbage collection, reflection,* akar kelas (*root class*), dan juga penyederhanaan terhadap pewarisan jamak (multiple *inheritance*).

Bahasa pemrograman C# dibuat sebagai bahasa pemrograman yang bersifat *general-purpose* (untuk tujuan jamak), berorientasi objek, modern, dan sederhana. C# ditujukan agar cocok digunakan untuk menulis program aplikasi baik dalam sistem *klien-server* (*hosted system*) maupun sistem *embedded (embedded system),* mulai dari program aplikasi yang sangat besar yang menggunakan sistem operasi yang canggih hingga kepada program aplikasi yang sangat kecil.

2.12 UML (*Unified Modeling Language*)

UML adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek. Definisi ini merupakan definisi sederhana. Pada kenyataannya, pendapat orang-orang tentang UML berbeda satu sama lain. Hal ini dikarenakan oleh sejarahnya sendiri dan oleh perbedaan persepsi tentang apa yang membuat sebuah proses rancang bangun perangkat lunak efektif [14].

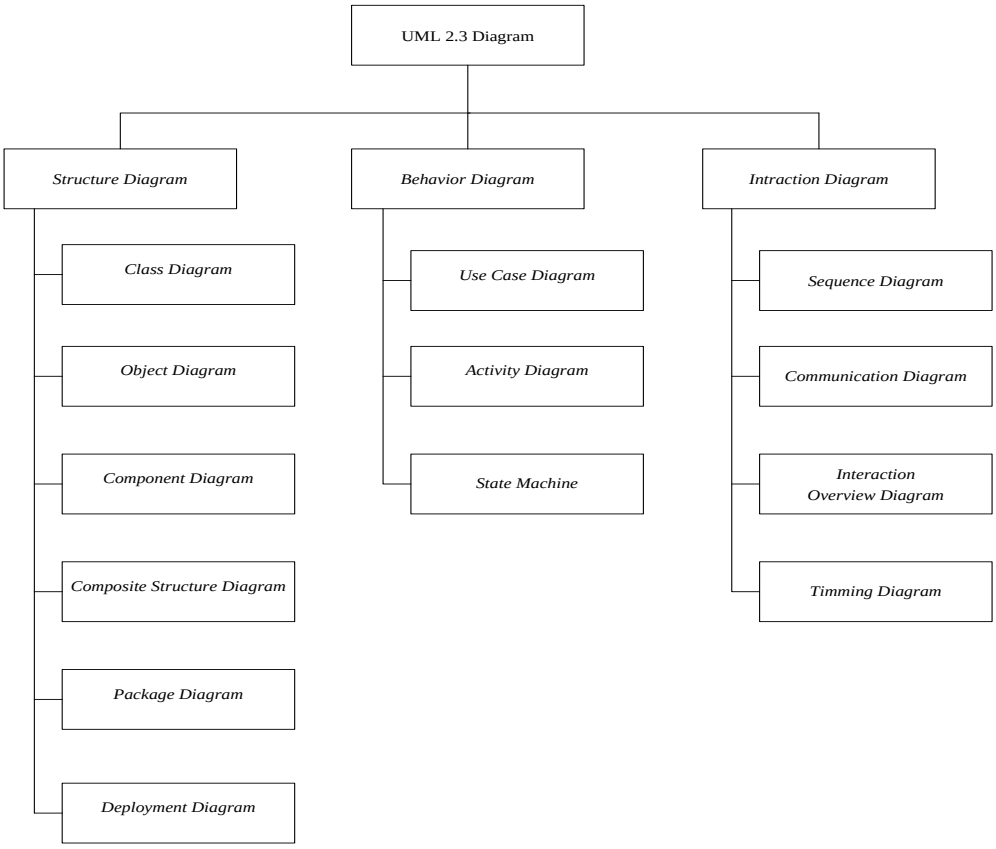
UML merupakan standar yang relatif terbuka yang dikontrol oleh O*bject Management Group (OMG),* sebuah konsorsium terbuka yang terdiri dari banyak perusahaan. OMG dibentuk untuk membuat standar-standar yang mendukung interoperabilitas, khususnya interoperabilitas sistem berorientasi objek. OMG mungkin lebih dikenal dengan standar-standar *COBRA (Common Object Request Broker Architecture)*.

UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi objek yang berkembang pesat pada akhir 1980-an dan awal 1990-an. Sejak kehadirannya pada tahun 1997, UML menggantikan menara Babel yang telah menjadi sejarah. UML merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari IBM.

Bagian-bagian utama dari UML adalah view, diagram, model element, dan general mechanism . Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sistem. Sebuah diagram merupakan bagian dari suatu view tertentu dan ketika digambarkan biasanya.

2.12.1 Diagram UML

UML terdiri dari 13 macam diagram yang dikelompokkan dalam 3 kategori [13]. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar 2.11 di bawah ini.



Gambar 2. 11 Diagram UML

Berikut ini penjelasan singkat dari pembagian kategori tersebut.

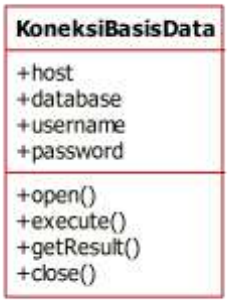
1. *Structure diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkain perubahan yang terjadi pada sebuah sistem.
3. *Interaction diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan interaksi sistem dengan sistem lain maupun interaksi antarsubsistem pada suatu sistem.

2.11.1.1 Diagram Kelas (*Class Diagram*)

Diagram kelas menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
2. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuat program atau *programmer* membuat kelas-kelas sesuai rancangan di dalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron. Berikut adalah contoh dari diagram kelas.



Gambar 2. 12 Contoh *class diagram* [13]

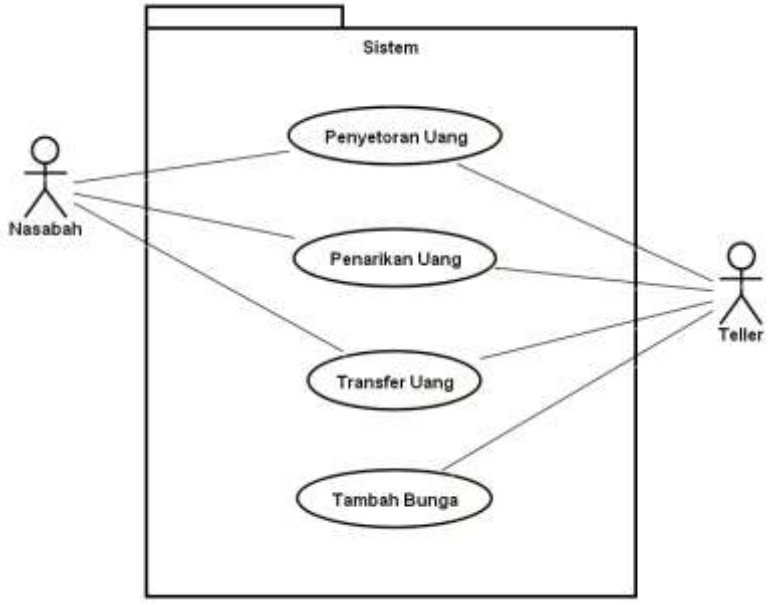
Pada gambar 2.12 diatas terdapat sebuah contoh dari *class diagram* yang bernama *class* KoneksiBasisData yang terdiri dari tiga bagian, bagian pertama menjelaskan nama dari class tersebut yaitu class KoneksiBasisData, bagian kedua adalah atribut dari *class* tersebut yaitu *host,database, username, password* dan bagian ke tiga adalah proses yang berjalan dari *class* tersebut yaitu proses *open, execute, getresult,*dan *close.*

2.11.1.2 *Diagram Use Case*

Diagram *use case* merupakan pemodelan untuk kelakuan *(behaviour)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi

itu. Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut aktor dan *use case*.

1. Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2. *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.



Gambar 2. 13 Contoh *use case diagram* [13]

Pada gambar 2.13 di atas di jelaskan bahwa terdapat 2 *actor yaitu* Nasabah dan teller di dalam *use case diagram* tersebut, Nasabah hanya dapat memasuki *system* untuk melakukan Penyetoran uang, penarikan Uang, dan Transfer uang sedangkan Teller dapat melakukan semua hal yang nasabah bisa ditambah dengan menambah bunga.

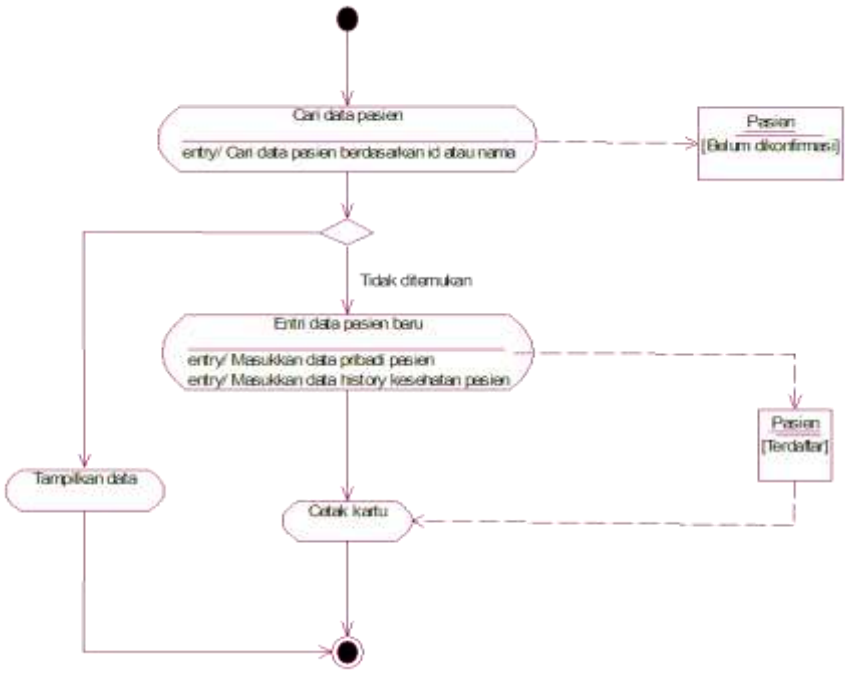
2.11.1.3 Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas atau activity diagram adalah sebuah diagram yang menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Dalam diagram aktivitas yang perlu diperhatikan adalah bahwa diagram aktivitas menggambarkan aktivitas sistem, bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut :

1. Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkanmerupakan proses bisnis sistem yang didefinisikan.
2. Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.
3. Rancangan proses bisnis di mana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

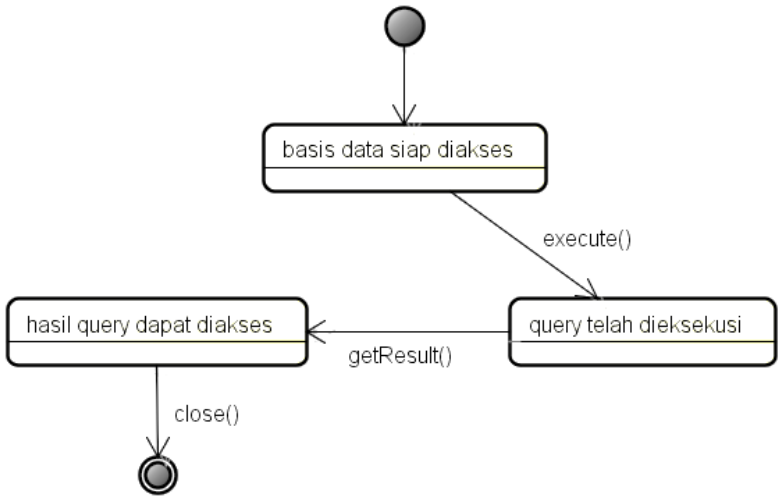
Contoh dari activity diagram dapat dilihat pada gambar 2.14.



Gambar 2. 14 Contoh dari Activity Diagram

2.11.1.4 *Diagram State Machine*

State machine diagram atau *statechart diagram* atau dalam bahasa Indonesia disebut diagram mesin status atau sering juga disebut diagram status digunakan untuk menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek. Jika diagram sekuen digunakan untuk interaksi antar objek maka diagram status digunakan untuk interaksi di dalam sebuah objek. Perubahan tersebut digambarkan dalam suatu graf berarah.



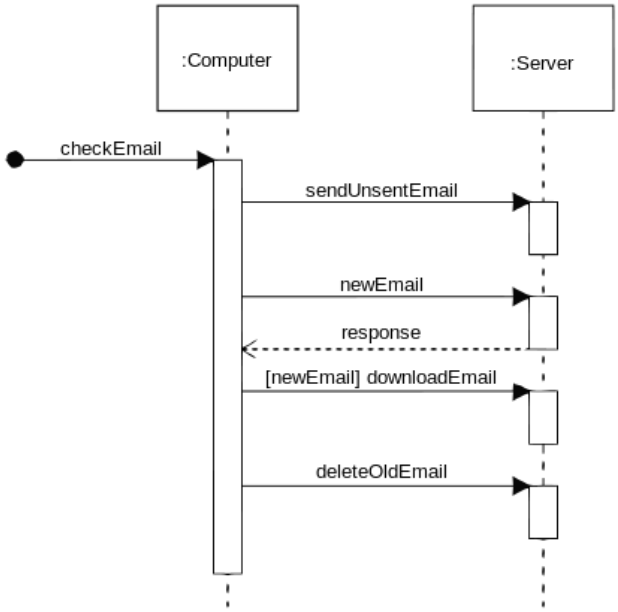
Gambar 2. 15 Contoh *statechart diagram* [13].

Pada gambar 2.15 di atas di jelaskan sebuah contoh *state chart diagram* pada sebuah *database* langkah pertama menyiapkan basisdata agar dapat diakses kemudian mengeksekusi *query* dan mengakses hasil *query* kemudian menutup database.

2.11.1.5 Diagram Sekuen (*Sequence Diagram*)

Diagram sekuen menggambarkan kelakuan pada objek *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat skenario yang ada pada *use case*.

Banyaknya diagram sekuen yang harus digambar adalah minimal sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak.



Gambar 2. 16 Contoh sequence diagram [13].

Pada gambar 2.16 di gambarkan sebuah contoh *sequence diagram* yang mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek antara computer dan server pada proses pengecekan *email*. Pada langkah pertama *user* akan mengakses komputer kemudian komputer mengirim informasi kepada server yaitu mengirim email baru dan email yang belum terkirim, dan setelah itu server merespon informasi tersebut selanjutnya computer men-*download* email baru dari server dan mendelete email lama.