# BAB II

# Tinjauan Pustaka

## II.1. Pengertian Sistem

Sistem merupakan kumpulan dari unsur atau elemen – elemen yang saling berkaitan/berinteraksi dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu (Asbon Hendra, 2012).

## II.2. Sistem Pendukung Keputusan

Sistem Pendukung Keputusan dapat didefenisikan sebagai sistem berbasis komputer interaktif yang membantu para pengambilan keputusan untuk menggunakan data dan berbagai model untuk memecahkan masalah tidak terstruktur. SPK dirancang untuk menunjang seluruh tahapan pembuatan keputusan yang dimulai dari tahap mengidentifikasi masalah, memilih data yang relevan, menentukan pendekatan yang digunakan dalam proses pembuatan keputusan, sampai pada kegiatan mengevaluasi pemilihan alternatif (Dwi, dkk ; 2013 : 548)

## II.3. Metode SAW

Konsep dasar metode SAW adalah mencari penjumlahan terbobot dari rating kinerja pada setiap alternatif pada semua atribut. Metode SAW membutuhkan proses normalisasi matriks keputusan (X) kesuatu skala yang dapat

diperbandingan dengan semua rating alternatif yang ada (Youllia Indrawaty, 2011).

$$r_{ij} = \begin{cases} \dfrac{X_{ij}}{\underset{i}{Max}\, x_{ij}} & \text{Jika j atribut keuntungan } \textit{(benefit)} \\ \dfrac{\underset{i}{Min}\, x_{ij}}{x_{ij}} & \text{Jika j atribut biaya } \textit{(cost)} \end{cases} \quad \text{.....(II.1)}$$

Keterangan :

a. Rij = nilai rating kinerja ternormalisasi
b. Xij = nilai atribut yang dimiliki dari setiap kriteria
c. Max xij = nilai terbesar dari setiap kriteria
d. Min xij = nilai terkecil dari setiap kriteria
e. *benefit* = jika nilai terbesar adalah terbaik
f. *cost* = jika nilai terkecil adalah terbaik

dimana Rij adalah rating kinerja ternormalisasi dari alternatif Ai pada atribut Cj ; i=1,2,...,m dan j=1,2,...,n. Nilai preferensi untuk setiap alternatif ( Vi) diberikan sebagai:

$$V_i = \sum_{j=1}^{n} w_j r_{ij} \quad \text{.....(II.2)}$$

Keterangan :

a. Vi = rangking untuk setiap alternative
b. Wj= nilai bobot dari setiap kriteria
c. rij= nilai rating kinerja ternormalisasi

Nilai $V_i$ yang lebih besar mengindikasikan bahwa alternatif $A_i$ lebih (Destriyana Darmastuti, 2012).

### II.3.1. Langkah *Simple Additive Weight (SAW)*

Langkah – langkah dari metode SAW adalah :

1. Menentukan kriteria-kriteria yang akan dijadikan acuan dalam pengambilan keputusan, yaitu Ci.
2. Menentukan rating kecocokan setiap alternatif pada setiap kriteria.
3. Membuat matriks keputusan berdasarkan kriteria (Ci), kemudian melakukan normalisasi matriks Berdasarkan persamaan yang disesuaikan dengan jenis atribut (atribut keuntungan ataupun atribut biaya) sehingga diperoleh matriks ternormalisai R.
4. Hasil akhir diperoleh dari proses perankingan yaitu penjumlahan dari perkalian matriks ternormalisasi R dengan vektor bobot sehingga diperoleh nilai terbesar yang dipilih sebagai alternatif terbaik (Ai) sebagai solusi.(Youlia Indrawati, 2011)

## II.4. Visual Studio 2010

*Visual* Basic 2010 adalah inkarnasi dari bahasa visual basic yang sangat populer dan telah dilengkapi dengan fitur serta fungsi yang setara dengan bahasa tingkat lainnya sseperti C++. Anda dapat menggunakan visual basic 2010 untuk membuat aplikasi windows, mobile, web, dan office atau kode yang telah ditulis oleh orang lain dan kemudian dimasukkan ke program lainnya. Visual basic menyediakan berbagai tools dan fitur canggih yang memungkinkan dapat menulis kode, menguji dan menjalankan program tunggal ataau terkadang serangkaian program yang terkait dengan satu aplikasi (Christopher Lee, 2014).

Bahasa pemrograman visual basic merupakan bahasa pemrograman utama dari perusahaan *Microsoft Inc* yang paling sukses hingga 12 tahun. Bahasa pemrograman ini menjadi contoh semua pemrograman *RAD*. Hingga saat ini kepopuleran bahasa pemrograman visual basic masih bertahan kuat karena kemudahan, ringan dan andal. *Visual Studio 2010* merupakan *IDE* bahasa pemrograman visual basic menggunakan teknologi .Net versi 3.5. *visual studio 2010* hadir dengan edisi *Team System, Professional Edition*, *Standart Edition* dan *Express Edition.*

## II.5. SQL Server

*SQL Server 2008* adalah sebuah RDBMS (*Relational Database Management System*) yang di-develop oleh Microsoft, yang digunakan untuk menyimpan dan mengolah data. Pada *SQL Server 2008*, kita bisa melakukan pengambilan dan modifikasi data yang ada dengan cepat dan efisien. Pada *SQL*

*Server 2008*, kita bisa membuat objek-objek yang sering digunakan pada aplikasi bisnis, seperti membuat database, *table, function, stored database, trigger* dan *view*. Selain objek, kita juga menjalankan perintah Sql (*Structured Query Language*) untuk mengambil data (Elex Media Competindo, 2010).

## II.6. UML (Unified Modelling Language)

*Unified Modelling Language* (UML) menyediakan beberapa notasi dan artifak standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. Artifak dalam UML adalah informasi dalam berbagai bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat lunak. Contohnya adalah *source code* yang dihasikan oleh pemrograman. Yang perlu diperhatikan untuk menjaga konsistensi antar artifak selama proses analisis dan desain adalah bahwa setiap perubahan terjadi pada suatu artifak yang harus juga dilakukan pada artifak lainnya. (Sumber: Evi Triandini Dan I gede Suardika : 2012:15)

### II.6.1. Jenis-Jenis Diagram UML

Berikut ini adalah mengenai berbagai diagram UML serta tujuan dan simbol-simbolnya (Mhd.Alpan;2015), yaitu :

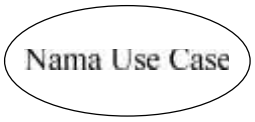
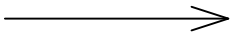
1. *Use Case Diagram* (Diagram *Use Case*)

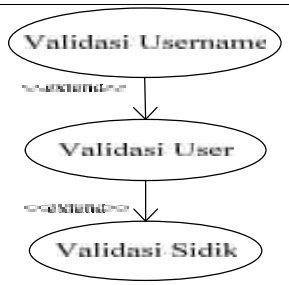
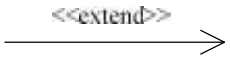
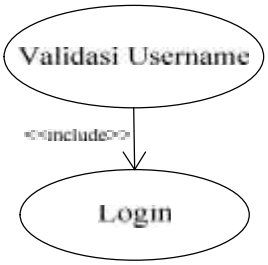
   *Use Case Diagram* merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara

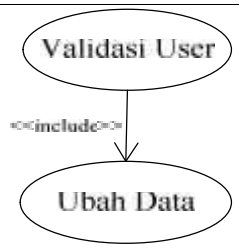
dasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Berikut ini adalah simbol-simbol yang ada pada diagram *use case* :

**Tabel II.1. Simbol-simbol *Use Case Diagram***

| **Simbol** | **Deskripsi** |
|---|---|
| *Use case*<br>  | Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling tertukar pesan antar unit atau aktor, biasanya dengan menggunakan kata kerja di awal *frase* nama *use case.* |
| Aktor/*actor*<br>  | Orang, proses, atau lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah orang, biasanya dinyatakan menggunakan kata benda di awal *fase* nama aktor. |
| Asosiasi/*association*<br> | Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* miliki interaksi dengan aktor. |
| Ekstensi/*extend*<br>  | Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, misal : |

| | |
|---|---|
| |  <br>Arah panah mengarah pada *use case* yang ditambahkan; biasanya *use case* yang menjadi *extend*-nya merupakan jenis yang sama dengan *usecase* yang menjadi induknya. |
| *Include*<br>  | Relasi *use case* tambahan sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. Ada dua sudut pandang yang cukup besar mengenai *include* di *use case* :<br>• *Include* berarti *use case* yang ditambahkan akan selalu di panggil saat *use case* tambahan dijalankan. Contoh :<br>  <br>• *Include* berarti *use case* yang tambahan akan selalu melakukan pengecekan apakah *use case* yang ditambahkan telah dijalankan sebelum *use case* tambahan dijalankan, misal pada kasus berikut : |

| | |
|---|---|
| |  <br>Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan. |

**(Sumber : Mhd.Alpan, 2015)**

2. *Class Diagram* (Diagram Kelas)

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

a. Atribut merupakan variabel-variabel yang dimiliki suatu kelas.

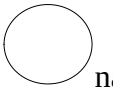
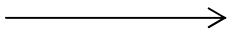
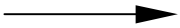
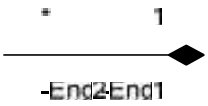
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

*Class diagram* dibuat agar pembuat program atau programmer membuat kelas-kelas sesuai rancangan didalam *class diagram* agar antara dokumentasi, perancangan, dan perangkat lunak sinkron.

Berikut ini adalah simbol-simbol yang ada pada *class diagram* :

**Tabel II.2. Simbol-simbol *Class Diagram***

| **Simbol** | **Deskripsi** |
|---|---|

| | |
|---|---|
| Kelas<br>**Nama Kelas**<br>+ Attribute1<br>+ Attribute2<br>+ Operation 1 ( ) | Kelas pada struktur sistem. |
| Antarmuka/*interface*<br>  <br>nama_interface | Sama dengan konsep *interface* dalam pemrograman berorientasi objek. |
| Asosiasi/*association*<br> | Relasi antarkelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*. |
| Asosiasi berarah/*directed asosiasi*<br>  | Relasi antarkelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicaty*. |
| *Generalisasi*<br>  | Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus). |
| Kebergantungan/*dependency*<br>  | Relasi antarkelas dengan makna kebergantungan antarkelas. |
| Agregasi/*aggregiation*<br>  <br>* 1<br>End2 End1 | Relasi antarkelas dengan makna semua-bagian (*whole-part*). |

**(Sumber : Mhd.Alpan, 2015)**

3. *Activity Diagram* (Diagram Aktivitas)

*Activity Diagram* adalah versi UML untuk sebuah *flowchart*. *Activity diagram* digunakan untuk menganalisa proses. Sebuah activity diagram bukan sebuah tool yang sempurna untuk menganalisis masalah dari sistem. Sebagai tool untuk menganalisis, pemrogram tidak ingin untuk mulai memecahkan masalah dilevel teknis dengan membuat class, tetapi dengan menggunakan *activity diagram* untuk mengerti masalah dan menyaring proses yang terdapat dalam sistem. Setiap *activity diagram* selalu mempunyai satu *initial state. Initial node* yang digambarkan dengan simbol lingkaran padat merupakan titik yang mengawali *activity diagram*. *Activity diagram* dapat diakhiri dengan memberikan *activity final* diagram yang digambarkan dengan lingkaran padat dengan mempunyai cincin dibagian luarnya.

**Tabel II.3. Simbol – Simbol Activity Diagram**

| **Simbol** | **Keterangan** |
|---|---|
|  | Titik awal |
|  | Titik akhir |
|  | *Activity* |
| | Pilihan untuk mengambil keputusan |
|  | *Fork*; Digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua |

| | |
|---|---|
| | kegiatan paralel menjadi satu. |
|  | *Rake*; Menunjukkan adanya dekomposisi |
|  | Tanda waktu |
|  | Tanda pengiriman |
|  | Tanda penerimaan |
|  | Aliran akhir (*Flow Final*) |

***(Sumber : Munawar, 2005)***

4. *Sequence Diagram* (Diagram Rangkaian)

   *Sequence diagram* menggambarkan kelakukan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu, untuk menggambar diagram *sequence* maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu. Membuat *Sequence diagram* juga dibutuhkan untuk melihat skenario yang ada pada *use case* (Mhd.Alpan, 2015).

## II.8. Normalisasi

*Kroenke* mendefinisikan normalisasi sebagai proses untuk mengubah suatu relasi yang memiliki masalah tertentu ke dalam dua buah relasi atau lebih yang tida memiliki masalah tersebut. Masalah yang dimaksud oleh kroenke ini sering disebut dengan istilah anomali.

Normalisasi merupakan sebuah teknik dalam logical desain sebuah basis data / database, teknik pengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik (tanpa redudansi).

Normalisasi adalah suatu proses memperbaiki / membangun dengan model data relasional, dan secara umum lebih tepat dikoneksikan dengan model data logika. (I Wayan Susena ; 2011 : 06)

Proses normalisasi adalah proses pengelompokan data elemen menjadi tabel-tabel yang menunjukkan entity dan relasinya. Pada proses normalisasi dilakukan pengujian pada beberapa kondisi apakah ada kesulitan pada saat menambah/menyisipkan, menghapus, mengubah dan mengakses pada suatu basis data. Bila terdapat kesulitan pada pengujian tersebut maka perlu dipecahkan relasi pada beberapa tabel lagi atau dengan kata lain perancangan basis data belum optimal.

Ada lima bentuk normal yang telah ditemukan yaitu :

1. Bentuk Normal Pertama (1 NF / First Normal Form)

   Bentuk Bentuk Normal Kesatu mempunyai ciri yaitu setiap data dibentuk dalam file flat, data dibentuk dalam satu record demi satu record dan nilai dari field

berupa “atomic value”. Tidak ada set atribut yang berulang ulang atau atribut bernilai ganda (multi value). Tiap field hanya satu pengertian, bukan merupakan kumpulan data yang mempunyai arti mendua. Hanya satu arti saja dan juga bukanlah pecahan kata kata sehingga artinya lain.

2. Bentuk Normal Kedua (2NF)

   Bentuk Normal kedua mempunyai syarat yaitu bentuk data telah memenuhi kriteria bentuk Normal Kesatu. Atribut bukan kunci haruslah bergantung secara fungsi pada kunci utama, sehingga untuk membentuk Normal Kedua haruslah sudah ditentukan kunci-kunci field. Kunci field harus unik dan dapat mewakili atribut lain yang menjadi anggotanya.

3. Bentuk Normal Ketiga (3NF)

   Untuk menjadi bentuk Normal Ketiga maka relasi haruslah dalam bentuk Normal Kedua dan semua atribut bukan primer tidak punya hubungan yang transitif. Artinya setiap atribut bukan kunci harus bergantung hanya pada kunci primer secara menyeluruh.

4. Boyce-Codd Normal Form (BNCF)

   Boyce-Codd Normal Form mempunyai paksaan yang lebih kuat dari bentuk Normal ketiga. Untuk menjadi BNCF, relasi harus dalam bentuk Normal Kesatu dan setiap atribut dipaksa bergantung pada fungsi pada atribut super key.

5. Bentuk Normal Keempat (4NF)

   Semua anomali yang bersal dari ketergantungan banyak-nilai (multivalues dependeny) telah diilangkan. (Adi Nugroho ; 2010 : 34)

Tujuan Normalisasi adalah

1. Memudahkan user dalam akses data
2. Optimalisasi struktur tabel
3. Optimalisasi *storage*
4. Mengurangi redundansi
5. Menghindari anomali (*insert, delete, update*)
6. Peningkatan integritas data