BAB II

LANDASAN TEORI

II.1 Sistem

II.1.1 Konsep Dasar Sistem

Terdapat dua kelompok pendekatan di dalam pendefinisian sistem, yaitu kelompok yang menekankan pada prosedur dan kelompok yang menekankan pada elemen atau komponennya. Pendekatan yang menekankan pada prosedur mendefinisikan sistem sebagai suatu jaringan kerja prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. Sedangkan pendekatan sistem yang lebih menekankan pada elemen atau komponen mendefinisikan sistem sebagai kumpulan elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. Kedua kelompok ini adalah benar dan tidak bertentangan, yang beda adalah cara pendekatannya (Tata Sutabri, 2012 : 2).

II.1.2 Pengertian Sistem

Secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisasi, saling berinteraksi, saling bergantung satu sama lain dan terpadu (Tata Sutabri, 2012 : 3).

Gordon B. Davis dalam bukunya menyatakan bahwa sistem bisa berupa abstrak atau fisik.Sistem yang abstrak adalah susunan gagasan-gagasan atau konsepsi yang teratur yang saling bergantung. Sedangkan sistem yang bersifat

fisik adalah serangkaian unsur yang bekerja sama untuk mencapai suatu tujuan (Tata Sutabri, 2012 : 6).

Sedangkan Norman L. Enger menyatakan bahwa suatu sistem dapat terdiri atas kegiatan-kegiatan yang berhubungan guna mencapai tujuan-tujuan perusahaan seperti pengendalian inventaris atau penjadwalan produksi (Tata Sutabri, 2012 : 7).

II.2 Sistem Pendukung Keputusan (*Decision Support System*)

II.2.1 Pengertian Sistem Pendukung Keputusan

Sistem Pendukung Keputusan (*Decision Support System*) adalah sistem informasi yang mendukung bisnis atau kegiatan organisasi yang melibatkan pengambilan keputusan. *Decision Support System* sangat berguna dalam situasi yang cepat berubah dan sulit menentukan suatu kondisi yang belum pernah ditemui sebelumnya (Krzywicki D., et al., 2014 : 4).

Konsep sistem pendukung keputusan ditandai dengan sistem interaktif berbasis komputer yang membantu membentuk keputusan, memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang tidak terstruktur. Pada dasarnya sistem pendukung keputusan dirancang untuk mendukung seluruh tahap pengambilan keputusan mulai dari mengidentifikasi masalah, memilih data yang relevan, menentukan pendekatan yang digunakan dalam pengambilan keputusan, sampai mengevaluasi pemilihan interaktif (Maulidia Indapuri, 2014 : 86).

II.2.2 Karakteristik Sistem Pendukung Keputusan

Konsep Sistem Pendukung Keputusan pertama kali diperkenalkan pada tahun 1970-an oleh Michael S. Scott Morton dengan istilah *Management Decision Model*. Konsep sistem pendukung keputusan ditandai dengan sistem interaktif berbasis komputer yang membantu membentuk keputusan, memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang tidak terstruktur. Pada dasarnya sistem pendukung keputusan dirancang untuk mendukung seluruh tahap pengambilan keputusan mulai dari mengidentifikasi masalah, memilih data yang relevan, menentukan pendekatan yang digunakan dalam pengambilan keputusan, sampai mengevaluasi pemilihan interaktif. Peranan sistem pendukung keputusan dalam konteks keseluruhan sistem informasi ditujukan untuk memperbaiki kinerja melalui aplikasi teknologi informasi (Maulidia Indapuri, 2014 : 86).

Terdapat sepuluh karakteristik dasar sistem pendukung keputusan yang efektif yaitu (Maulidia Indapuri, 2014 : 86):

1. Mendukung proses pengambilan keputusan, menitikberatkan pada *management by perception.*
2. Adanya *interface* manusia atau mesin di mana manusia (*user*) tetap mengontrol proses pengambilan keputusan.
3. Menggunakan model-model matematis dan statistik yang sesuai.
4. Memiliki kapabilitas dialog untuk memperoleh informasi sesuai dengan kebutuhan model interaktif.
5. *Output* ditunjukkan untuk personil organisasi dalam semua tingkatan.

6. Memiliki subsistem-subsistem yang terintegrasi sedemikian rupa sehingga dapat berfungsi sebagai kesatuan sistem.
7. Membutuhkan struktur data komprehensif yang dapat melayani kebutuhan informasi keseluruhan tingkatan manajemen.
8. Pendekatan *easy to use.* Ciri suatu sistem pendukung keputusan yang efektif adalah kemudahan untuk digunakan dan memungkinkan keleluasaan pemakai untuk memilih atau mengembangkan pendekatan-pendekatan baru dalam membahas masalah yang dihadapi.
9. Kemampuan sistem beradaptasi secara tepat, di mana pengambil keputusan dapat menghadapi masalah baru pada saat yang sama dapat menangani dengan cara mengadaptasi sistem terhadap kondisi-kondisi dan perubahan yang terjadi.
10. Mendukung pengambilan keputusan untuk membahas masalah-masalah terstruktur, semiterstruktur, dan tidak terstruktur.

II.2.3 Komponen-komponen Sistem Pendukung Keputusan

Suatu sistem pendukung keputusan memiliki tiga subsistem utama yang menentukan kapabilitas teknis sistem pendukung keputusan tersebut, yaitu (Maulidia Indapuri, 2014 : 86-87):

1. Subsistem Manajemen Basis Data (*Database Management Subsystem*)
   Sistem pendukung keputusan membutuhkan proses ekstraksi dan *Database Management Subsystem* (DBMS) yang dalam pengelolaannya harus cukup fleksibel untuk memungkinkan penambahan dan pengurangan secara cepat. Dalam hal ini,

kemampuan yang dibutuhkan dari manajemen *database* dapat diringkas sebagai berikut:

a. Kemampuan untuk mengkombinasikan berbagai variasi data melalui pengambilan dan ekstraksi data.
b. Kemampuan untuk menambahkan sumber data secara cepat dan mudah.
c. Kemampuan untuk menangani data secara personil sehingga pemakai dapat mencoba berbagai alternatif pertimbangan personil.
d. Kemampuan untuk menggambarkan struktur data logikal sesuai dengan pengertian pemakai sehingga pemakai mengetahui apa yang tersedia dan dapat menentukan kebutuhan penambahan dan pengurangan.

2. Subsistem Manajemen Basis Model (*Model Basis Management Subsystem*)

   Salah satu keunggulan dalam sistem pendukung keputusan adalah kemampuan untuk mengintegrasikan akses data dan model-model keputusan. Hal ini dapat dilakukan dengan menambahkan model-model keputusan ke dalam sistem informasi yang menggunakan *database* sebagai mekanisme integrasi dan komunikasi di antara model-model. Kemampuan yang dimiliki subsistem berbasis model meliputi:

a. Kemampuan untuk menciptakan model-model baru secara cepat dan mudah.
b. Kemampuan untuk mengakses dan mengintegrasikan model-model keputusan.
c. Kemampuan untuk mengelola basis data dengan fungsi manajemen yang analog dan manajemen basis data (seperti mekanisme untuk menyimpan, membuat dialog, menghubungkan, dan mengakses model).

3. Subsistem Perangkat Lunak Penyelenggara Dialog (*Dialog Generation and Management Software*)

   Fleksibilitas dan kekuatan karakteristik sistem pendukung keputusan timbul dari kemampuan interaksi antara sistem dan pemakai, yang dinamakan subsistem dialog. Bennet mendefinisikan pemakai, terminal, dan sistem perangkat lunak sebagai komponen-komponen dari sistem dialog. Ia membagi subsistem dialog menjadi tiga bagian yaitu:

   a. Bahasa aksi, meliputi apa yang dapat digunakan pemakai dalam berkomunikasi dengan sistem.
   b. Bahasa tampilan atau presentasi, meliputi apa yang harus diketahui oleh pemakai.
   c. Basis pengetahuan, meliputi apa yang harus diketahui oleh pemakai.

Kombinasi dari kemampuan-kemampuan di atas terdiri dari apa yang disebut gaya dialog, misalnya pendekatan tanya jawab dan perintah. Kemampuan yang harus dimiliki oleh sistem pendukung keputusan untuk mendukung dialog atau sistem meliputi:

a. Kemampuan untuk memberikan dukungan dan mengetahui basis pengetahuan pemakai.
b. Kemampuan untuk menangani berbagai variasi gaya dialog, bahkan juga mungkin untuk mengkombinasikan berbagai gaya dialog sesuai dengan pilihan pemakai.
c. Kemampuan untuk mengakomodasi tindakan pemakai dengan berbagai peralatan masukan.
d. Kemampuan untuk menampilkan data dengan berbagai variasi format peralatan keluaran.
e. Kemampuan untuk memberikan dukungan yang fleksibel dan mengetahui basis pengetahuan pemakai.

II.3 Logika *Fuzzy*

II.3.1 Pengertian Logika *Fuzzy*

Konsep tentang logika *fuzzy* diperkenalkan oleh Prof. Lotfi Astor Zadeh pada 1962. Logika *fuzzy* adalah metodologi sistem kontrol pemecahan masalah yang cocok untuk diimplementasikan pada sistem, mulai dari sistem yang sederhana, sistem kecil, *embedded system*, jaringan PC, *multi-channel* atau *workstation* berbasis akuisisi data, dan sistem kontrol metodologi ini dapat

diterapkan pada perangkat keras, perangkat lunak, atau kombinasi keduanya (T. Sutojo, et al., 2011 : 211).

II.3.2 Dasar-dasar Logika *Fuzzy*

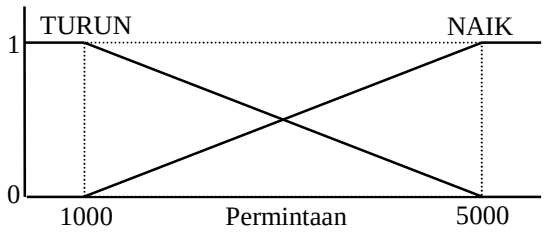
Untuk memahami logika *fuzzy*, sebelumnya perhatikan dahulu tentang konsep himpunan *fuzzy*. Himpunan *fuzzy* memiliki 2 atribut, yaitu (T. Sutojo, et al., 2011 : 212-213):

1. Linguistik, yaitu nama suatu kelompok yang mewakili suatu keadaan tertentu dengan menggunakan bahasa alami, misalnya DINGIN, SEJUK, PANAS mewakili variabel temperatur. Contoh lain misalnya MUDA, PAROBAYA, TUA mewakili variabel umur.
2. Numeris, yaitu suatu nilai yang menunjukkan ukuran dari suatu variabel, misalnya 10, 35, 40, dan sebagainya.

Di samping itu, ada beberapa hal yang harus dipahami dalam memahami logika *fuzzy*, yaitu:

1. Variabel *fuzzy*, yaitu variabel yang akan dibahas dalam suatu sistem *fuzzy*. Contoh: penghasilan, temperatur, permintaan, umur, dan sebagainya.
2. Himpunan *fuzzy*, yaitu suatu kelompok yang mewakili suatu keadaan tertentu dalam suatu variabel *fuzzy*. Contoh (Gambar II.1):

   Variabel permintaan terbagi menjadi 2 himpunan *fuzzy*, yaitu NAIK dan TURUN.



Gambar II.1 Variabel Permintaan Terbagi Menjadi 2 Himpunan *Fuzzy*, yaitu Himpunan NAIK dan Himpunan TURUN (Sumber: T. Sutojo, et al., 2011 : 213)

3. Semesta pembicaraan, yaitu seluruh nilai yang diizinkan untuk dioperasikan dalam suatu variabel *fuzzy*. Contoh:

   Semesta pembicaraan untuk variabel permintaan: [0 +∞].

   Semesta pembicaraan untuk variabel temperatur: [-10 90].

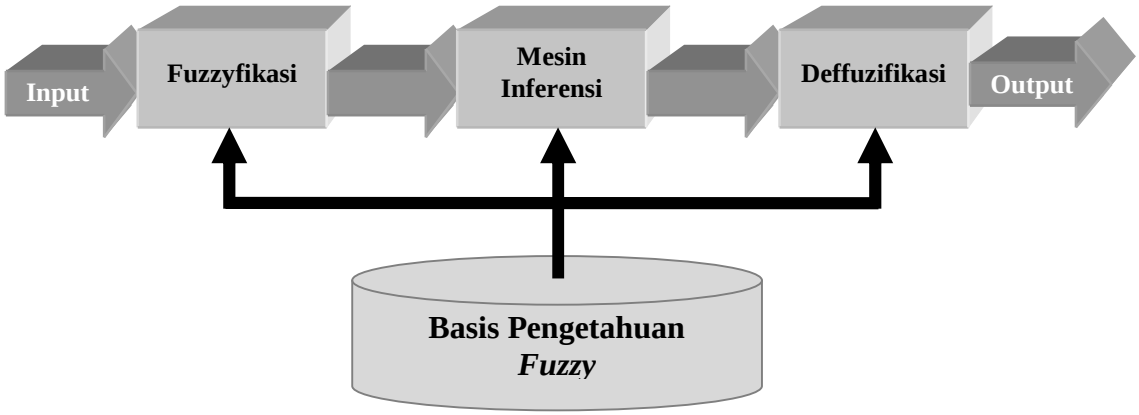
4. Domain himpunan *fuzzy*, yaitu seluruh nilai yang diizinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *fuzzy*. Pada Gambar II.1 domain untuk himpunan TURUN dan himpunan NAIK masing-masing adalah:

   Domain himpunan TURUN = [0 5000]

   Domain himpunan NAIK = [1000 +∞]

II.3.3 Cara Kerja Logika *Fuzzy*

Untuk memahami cara kerja logika *fuzzy*, perhatikan struktur elemen dasar sistem inferensi *fuzzy* pada Gambar II.2.



Gambar II.2 Struktur Sistem Inferensi *Fuzzy*
(Sumber: T. Sutojo, et al., 2011 : 232)

Keterangan:

1. Basis Pengetahuan *Fuzzy*: kumpulan *rule-rulefuzzy* dalam bentuk pernyataan *IF...THEN*.
2. Fuzzyfikasi: proses untuk mengubah *input* sistem yang mempunyai nilai tegas menjadi variabel linguistik menggunakan fungsi keanggotaan yang disimpan dalam basis pengetahuan *fuzzy*.
3. Mesin Inferensi: proses untuk mengubah *input fuzzy* menjadi *output fuzzy* dengan cara mengikuti aturan-aturan (*IF...THEN Rules*) yang telah ditetapkan pada basis pengetahuan *fuzzy*.
4. Defuzzyfikasi: mengubah *output fuzzy* yang diperoleh dari mesin inferensi menjadi nilai tegas menggunakan fungsi keanggotaan yang sesuai dengan saat dilakukan fuzzyfikasi.

II.3.4 Metode *Tsukamoto*

Pada metode *Tsukamoto,*setiap konsekuen pada aturan yang terbentuk *IF-THEN* harus direpresentasikan dengan suatu himpunan *fuzzy* dengan fungsi keanggotaan yang monoton. Sebagai hasilnya, *output* hasil inferensi dari tiap-tiap

aturan diberikan dengan tegas *(crisp)* berdasarkan α-predikat *(fire strength)*.Hasil akhirnya diperoleh dengan menggunakan rata-rata terbobot. Misalkan ada dua variabel *input*, yaitu x dan y serta satu variabel *output* z. Variabel x terbagi atas dua himpunan yaitu A_1 dan A_2, sedangkan variabel y terbagi atas himpunan B_1 dan B_2. Variabel z juga terbagi atas dua himpunan yaitu C_1 dan C_2 (Arkham Zahri Rakhman, et al., 2012).

Secara umum bentuk model *Fuzzy Tsukamoto* adalah: *IF (X is A) AND (Y is B) THEN (Z is C)*. Di mana A, B, dan C adalah himpunan *fuzzy*. Misalkan diketahui 2 rule berikut (T. Sutojo, 2011 : 233-234):

IF (X is A_1) AND (Y is B_1) THEN (Z is C_1)

IF (X is A_2) AND (Y is B_2) THEN (Z is C_2)

Dalam inferensinya, metode *Fuzzy Tsukamoto* menggunakan tahapan berikut:

1. Fuzzyfikasi
2. Pembentukan basis pengetahuan *Fuzzy* (*Rule* dalam bentuk *IF...THEN*).
3. Mesin inferensi

   Menggunakan fungsi implikasi *MIN* untuk mendapatkan nilai α-predikat tiap-tiap *rule* (α_1,α_2,α_3,...α_n). kemudian masing-masing nilai α-predikat ini digunakan untuk menghitung keluaran hasil inferensi secara tegas (*crisp*) masing-masing *rule* (z_1, z_2, z_3, ...z_n).

4. Defuzzyfikasi

   Menggunakan model rata-rata (*average*), ditunjukkan dengan persamaan (2.1).

$$z^* = \frac{\sum \alpha_i z_i}{\sum \alpha_i} \quad \text{.....(2.1)}$$

II.4 Metode *Profile Matching*

Metode *Profile Matching* secara garis besar merupakan proses membandingkan antara kompetensi individu kedalam kompetensi prestasi sehingga dapat diketahui perbedaan kompetensinya (disebut juga *Gap*), semakin kecil *Gap* yang dihasilkan maka bobot nilainya semakin besar yang berarti memiliki peluang lebih besar untuk seseorang individu itu mendapat nilai prestasi yang ditentukan (Asfan Muqtadir dan Irwan Purdianto, 2013 : 49).

Metode *Profile Matching* cukup efektif dalam menyederhanakan dan mempercepat proses pengambilan keputusan dengan memecahkan persolahan tersebut ke dalam bagian-bagiannya (Maulidia Indapuri, 2014 : 85).

Langkah-langkah pada metode *Profile Matching* yaitu (Arief Soma Darmawan, 2012 : 2-3):

1. Menentukan variabel-variabel pemetaan *Gap* kompetensi dan menentukan aspek-aspek yang akan digunakan dalam memproses nilai karyawan.
2. Menghitung hasil pemetaan *Gap* kompetensi. Yang dimaksud dengan *Gap* di sini adalah beda antara profile karyawan dengan profile standar yang diharapkan, dapat ditunjukkan dengan persamaan (2.2)

Gap = Profile karyawan – Profile standar .....(2.2)

Profile karyawan yaitu nilai-nilai yang diperoleh dari karyawan sedangkan profile standar yaitu nilai standar yang ditentukan terlebih dahulu. Setelah diperoleh *Gap* pada masing-masing karyawan, setiap profile karyawan diberi bobot nilai dengan patokan.

3. Kemudian setiap aspek dikelompokkan menjadi 2 kelompok, yaitu kelompok *Core Factor* dan *Secondary Factor*. Perhitungan *core factor* ditunjukkan dengan persamaan (2.3).

$$NCF = \frac{\sum Nc}{\sum Ic} \quad \text{.....(2.3)}$$

Di mana:

NCF : Nilai rata-rata *core factor*

Nc : Jumlah total nilai *core factor*

Ic : Jumlah item *core factor*

Sementara untuk perhitungan *secondary factor* ditunjukkan dengan persamaan (2.4).

$$NSF = \frac{\sum Ns}{\sum Is} \quad \text{.....(2.4)}$$

Di mana:

NSF : Nilai rata-rata *secondary factor*

Ns : Jumlah total nilai *secondary factor*

Is : Jumlah item *secondary factor*

4. Setelah perhitungan *Core factor* dan *Secondary factor*, kemudian menghitung nilai total berdasarkan dari persentase dari *core* dan

secondary yang diperkirakan berpengaruh terhadap kinerja tiap-tiap profile. Contoh perhitungan ditunjukkan dengan persamaan (2.5).

$(x)\%NCF+(x)\%NSF=N_{total}$ .....(2.5)

Di mana:

(x)% : Nilai persen yang diinputkan

N_{total} : Nilai total dari aspek

5. Proses terakhir adalah perhitungan Nilai Akhir, perhitungan tersebut bisa ditunjukkan dengan persamaan (2.6).

Nilai Akhir = (x)%N1 + (x)%N2 + (x)%N3 .....(2.6)

Di mana:

N1, N2, N3 : Nilai aspek yang sudah dihitung total

(x)% : Nilai persen yang diinputkan

II.5 Basis Data (*Database*)

II.5.1 Definisi Basis Data

Basis data dapat didefinisikan sebagai himpunan kelompok data yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah. Prinsip utamanya adalah pengaturan data. Tujuan utamanya kemudahan dan kecepatan dalam pengambilan kembali data (Priyanto Hidayatullah, 2012 : 137).

Sebuah basis data adalah sebuah kumpulan data yang saling berhubungan secara logis, dan merupakan sebuah penjelasan dari data tersebut, yang didesain untuk menemukan data yang dibutuhkan oleh sebuah organisasi. Di dalam basis

data, semua data diintegrasikan dengan menghindari duplikasi data. Basis data juga merupakan sekumpulan elemen data terintegrasi yang secara logika saling berhubungan. Basis data mengonsolidasikan berbagai catatan yang terlebih dahulu disimpan dalam file-file terpisah ke dalam satu gabungan umum elemen data yang menyediakan data untuk banyak aplikasi. Elemen data mendeskripsikan entitas-entitas dan hubungan antara entitas-entitas tersebut (Indrajani, 2015 : 70).

II.5.2 Tujuan Basis Data

Secara lebih lengkap pemanfaatan basis data dilakukan untuk memenuhi tujuan berikut ini (Priyanto Hidayatullah, 2012 : 138):

1. Kecepatan dan kemudahan (*Speed*).
2. Efisiensi ruang penyimpanan (*Space*).
3. Keakuratan (*Accuracy*).
4. Ketersediaan (*Availability*).
5. Kelengkapan (*Completeness*).
6. Keamanan (*Security*).
7. Pemakaian bersama (*Share ability*).

II.6 Kamus Data (*Data Dictionary*)

Kamus data adalah katalog fakta tentang data dan kebutuhan informasi suatu sistem informasi. Kamus data terdapat pada tahapan analisis dan perancangan. Pada tahap analisis, kamus data berfungsi untuk mendefinisikan data yang mengalir pada sistem. Sedangkan pada tahap perancangan, kamus data ini

digunakan untuk merancang masukan dan keluaran seperti laporan serta basis data (Indrajani, 2015 : 30-31).

Berikut notasi-notasi yang digunakan dalam kamus data terlihat pada Tabel II.1.

Tabel II.1 Notasi Kamus Data

Notasi	Keterangan
=	*Is composed of*
+	*And*
()	*Optional (may be present or absent)*
{}	*Iteration*
[]	*Select one of several alternative choices*
**	*Comment*
@	*Identifier (key field) for a store*
|	*Separates alternative choices in the [] construct*

(Sumber: Indrajani, 2015 : 31)

Contoh kamus data, antara lain:

name= courtesy-title + first-name + (*middle-name*) + *last-name*

courtesy-title=[*Mr.* | *Miss* | *Mrs.* | *Ms.* | *Dr.* | *Professor*]

first-name={*legal-character*}

middle-name={*legal-character*}

last-name= {*legal-character*}

legal-character=[A-Z | a-z | 0-9 | ` | - | |]

II.7 Normalisasi

II.7.1 Pengertian Normalisasi

Normalisasi adalah suatu teknik dengan pendekatan *bottom-up* yang digunakan untuk membantu mengidentifikasikan hubungan. Dimulai dari menguji hubungan, yaitu *functional dependencies* antara atribut. Pengertian lainnya adalah suatu teknik yang menghasilkan sekumpulan hubungan dengan sifat-sifat yang diinginkan dan memenuhi kebutuhan pada perusahaan (Indrajani, 2015 : 7).

Proses normalisasi merupakan proses pengelompokkan elemen data menjadi tabel-tabel yang menunjukkan entitas dan relasinya. Proses ini selalu diuji pada beberapa kondisi. Apakah ada kesulitan pada saat menambah (*insert*), menghapus (*delete*), mengubah (*update*), atau membaca (*retrieve*) pada satu *database*. Bila ada kesulitan pada pengujian tersebut maka relasi dapat dipecah dalam beberapa tabel lagi (Tata Sutabri, 2012 : 138).

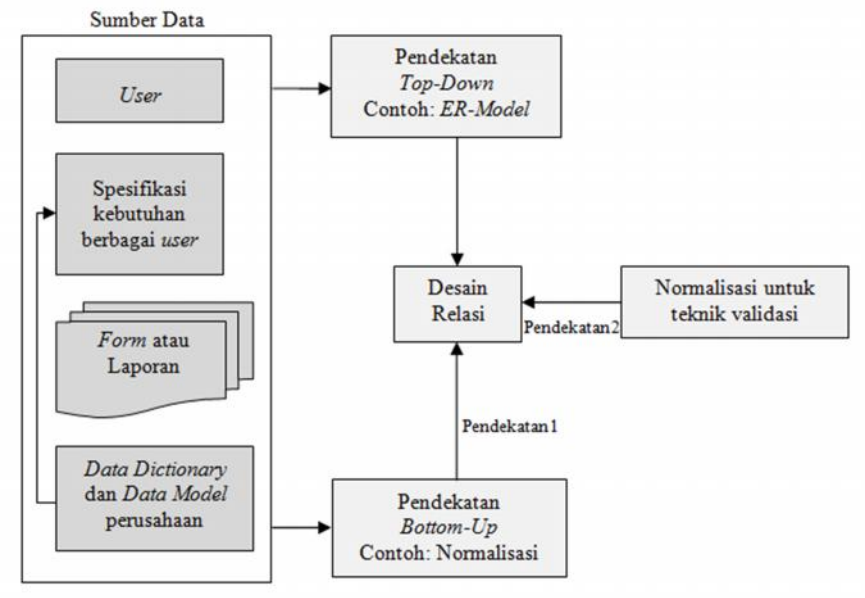
II.7.2 Tujuan Normalisasi

Tujuan utama normalisasi adalah mengidentifikasikan kesesuaian hubungan yang mendukung data untuk memenuhi kebutuhan perusahaan. Adapun karakteristik hubungan tersebut mencakup (Indrajani, 2015 : 7):

1. Minimal jumlah atribut yang diperlukan untuk mendukung kebutuhan perusahaan.
2. Atribut dengan hubungan logika yang menjelaskan mengenai *functional dependencies*.
3. Minimal duplikasi untuk tiap atribut.

II.7.3 Peranan Normalisasi dalam Perancangan Basis Data

Normalisasi adalah suatu teknik formal yang dapat digunakan dalam perancangan basis data. Peranan normalisasi dalam hal ini adalah dalam penggunaan pendekatan *bottom-up* dan teknik validasi. Teknik validasi digunakan untuk memeriksa, apakah struktur relasi yang dihasilkan oleh *ER-modeling* itu baik atau tidak baik (Indrajani, 2015 : 7). Peranan normalisasi dalam perancangan basis data dapat lebih jelas terlihat pada Gambar II.3.



Gambar II.3 Peranan Normalisasi dalam Perancangan Basis Data (Sumber: Indrajani, 2015 : 7)

Dari Gambar II.3 dapat dilihat sumber data terdiri dari *user*, spesifikasi kebutuhan berbagai *user*, berbagai *form* atau laporan, *data dictionary*, dan *data model* perusahaan. Kemudian terdapat pendekatan *top-down* dan *bottom-up*, di mana pendekatan tersebut nantinya menghasilkan desain relasi. Lalu peranan normalisasi pada *bottom-up* dan teknik validasi.

Adapun beberapa bentuk normalisasi yang biasa digunakan yaitu (Indrajani, 2015 : 8-10):

1. *Un-normalized Form* (UNF)

   Merupakan suatu tabel yang berisikan satu atau lebih grup yang berulang. Membuat tabel yang *un-normalized*, yaitu dengan memindahkan data dari sumber informasi. Contoh: nota penjualan yang disimpan ke dalam format tabel dengan baris dan kolom.

2. *First Normal Form* (1NF) atau Normalisasi Tingkat 1

   Merupakan sebuah relasi di mana setiap baris dan kolom berisikan satu dan hanya satu nilai. Adapun proses UNF ke 1NF yaitu:

   a. Tentukan satu atau kumpulan atribut sebagai kunci untuk tabel *un-normalized*.
   b. Identifikasikan grup yang berulang dalam tabel *un-normalized* yang berulang untuk kunci atribut.
   c. Hapus grup yang berulang dengan cara:
      1) Masukkan data yang semestinya ke dalam kolom yang kosong pada baris yang berisikan data yang berulang (*flattening the table*).
      2) Menggantikan data yang ada dengan menulis ulang dari kunci atribut yang sesungguhnya ke dalam relasi terpisah.

3. *Second Normal Form* (2NF) atau Normalisasi Tingkat 2

   Berdasarkan pada konsep *full functionaldependency*, yaitu A dan B merupakan atribut sebuah relasi. B dikatakan *fully dependent* terhadap

A jika B *functionally dependent* pada A tetapi tidak pada *proper subset* dari A. 2NF merupakan sebuah relasi dalam 1NF dan setiap atribut *non primary key* bersifat *fully functionally dependent* pada *primary key.* Adapun proses 1NF ke 2NF yaitu:

a. Identifikasi *primary key* untuk relasi 1NF.
b. Identifikasi *functional dependencies* dalam relasi.
c. Jika terdapat *partial dependencies* terhadap *primary key*, maka hapus dengan menempatkan dalam relasi yang beru bersama dengan salinan determinannya.

4. *Third Normal Form* (3NF) atau Normalisasi Tingkat 3

Berdasarkan pada konsep *transitive dependency*, yaitu suatu kondisi di mana A, B, dan C merupakan atribut sebuah relasi, maka A → B dan B → C, maka *transitively dependent* pada A melalui B (jika A tidak *functionally dependent* pada B atau C). 3NF adalah sebuat relasi dalam 1NF dan 2NF, di mana tidak terdapat atribut *non primary key* yang bersifat *transitively dependent* pada *primary key*. Adapun proses 2NF ke 3NF yaitu:

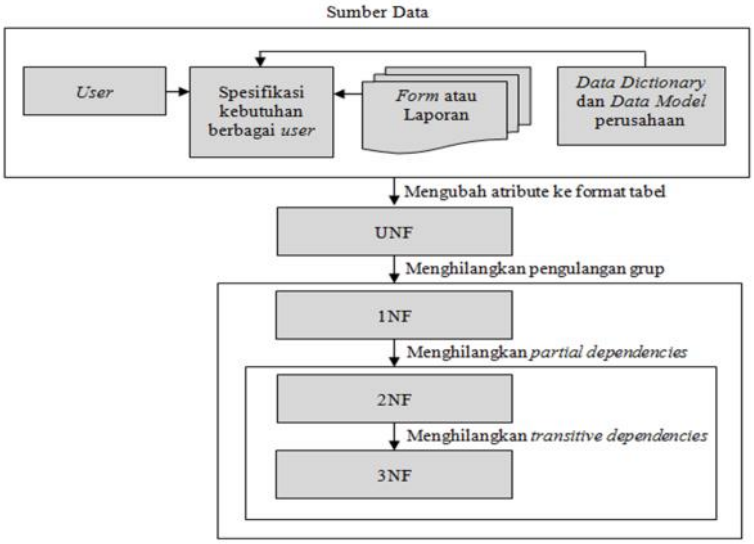
a. Identifikasikan *primary key* dalam relasi 2NF.
b. Identifikasikan *functional dependencies* dalam relasi.
c. Jika terdapat *transitive dependencies* terhadap *primary key*, hapus dengan menempatkannya dalam relasi yang baru bersama dengan salinan determinannya.

5. *Boyce-Code Normal Form* (BCNF)

   Berdasarkan pada *functional dependencies* yang dimasukkan ke dalam hitungan seluruh *candidate key* dalam suatu relasi. Bagaimana pun BCNF juga memiliki batasan-batasan tambahan disamakan dengan definisi umum dari 3NF. Suatu relasi dikatakan BCNF, jika dan hanya jika setiap determinan merupakan *candidate key*. Perbedaan antara 3NF dan BCNF yaitu untuk *functional dependency* A → B, 3NF memungkinkan *dependency* ini dalam satu relasi jika B adalah atribut *primary key* dan A bukan merupakan *candidate key*. Sedangkan BCNF menetapkan dengan jelas bahwa untuk *dependency* ini agar ditetapkan dalam relasi A, maka A harus merupakan *candidate key*. Setiap relasi dalam BCNF juga merupakan 3NF, tetapi relasi dalam 3NF belum tentu termasuk ke dalam BCNF. Dalam BCNF kesalahan jarang sekali terjadi, kesalahan dapat terjadi pada relasi yang:

   a. Terdiri atas 2 atau lebih *composite candidate key*.
   b. *Candidate key overlap*, sedikitnya satu atribut.

Adapun bentuk diagram proses normalisasi dapat lebih jelas dilihat pada Gambar II.4.



Gambar II.4 Diagram Proses Normalisasi
(Sumber: Indrajani, 2015 : 8)

II.8 *Entity Relationship Diagram* (ERD)

Entity Relational (ER) *Modeling* adalah sebuah pendekatan *top-bottom* dalam perancangan basis data yang dimulai dengan mengidentifikasikan data-data terpenting yang disebut dengan entitas dan hubungan antara entitas-entitas tersebut yang digambarkan dalam suatu model. Karena terdapat keterbatasan pada ER Model, maka terdapat pengembangan penambahan konsep semantik pada ER yang disebut *Enhanced Entity Relational* (EER) Model (Indrajani, 2015 : 17).

Entity Relationship Diagram (ERD) memiliki dua komponen utama yaitu Entitas (*Entity*) dan Relasi (*Relation*). Kedua komponen ini masing-masing dilengkapi dengan sejumlah atribut yang mempresentasikan seluruh fakta yang ada di dunia nyata (Eka, 2014 : 30). Entitas adalah sesuatu atau objek dalam dunia nyata yang dibedakan dari objek lain, misalnya: mahasiswa dan matakuliah. Entitas digambarkan dalam basis data dengan kumpulan atribut, misalnya: NIM,

nama, alamat, dan kota. Relasi adalah hubungan antara beberapa entitas, misalnya: relasi menghubungkan mahasiswa dengan matakuliah yang diambilnya (D. Tri Octafian, 2011 : 150).

Adapun simbol daripada *Entity Relationship Diagram* dapat dilihat pada Tabel II.2.

Tabel II.2 *Entity Relationship Diagram* (ERD)

No.	Simbol	Keterangan Fungsi
1.		Persegi panjang menyatakan himpunan entitas adalah orang, kejadian, atau berada di mana data akan dikumpulkan.
2.		Atribut merupakan informasi yang diambil tentang sebuah entitas.
3.		Belah ketupat menyatakan himpunan relasi merupakan hubungan antar entitas.
4.		Garis sebagai penghubung antara himpunan, relasi, dan himpunan entitas dengan atributnya.

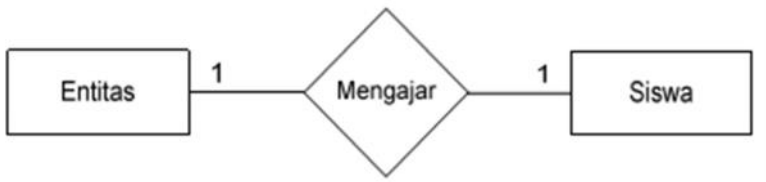
(Sumber: Ibnu Aqil, 2010 : 6)

Pemetaan kardinalitas menyatakan jumlah entitas di mana entitas lain dapat dihubungkan ke entitas tersebut melalui sebuah himpunan relasi. Kardinalitas relasi yang terjadi di antara dua himpunan entitas di antaranya adalah (D. Tri Octafian, 2011 : 151-152):

1. *One to One*

   Sebuah entitas pada A berhubungan dengan paling banyak satu entitas pada B dan sebuah entitas pada B berhubungan paling banyak satu

entitas pada A. Contoh: pada pengajaran privat, satu guru satu siswa. Seorang guru mengajar seorang siswa, seorang siswa diajar oleh seorang guru. Hubungan *One to One* dapat dilihat pada Gambar II.5.



Gambar II.5 Hubungan *One to One*
(Sumber: D. Tri Octafian, 2011 : 151)

2. *One to Many* / *Many to One*

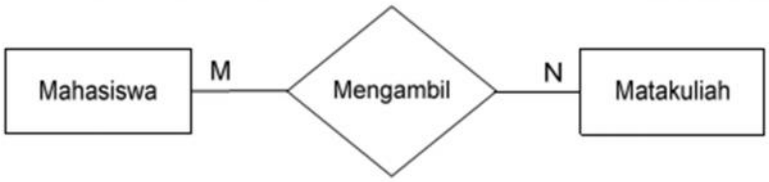
Sebuah entitas pada A berhubungan dengan lebih dari satu entitas pada B dan sebuah entitas pada B berhubungan dengan paling banyak satu entitas pada A, atau sebaliknya (*Many to One*). Contoh: dalam satu perusahaan, satu bagian mempekerjakan banyak pegawai. Satu bagian mempekerjakan banyak pegawai, satu pegawai kerja dalam satu bagian. Hubungan *One to Many* dapat dilihat pada Gambar II.6.

Gambar II.6 Hubungan *One to Many*
(Sumber: D. Tri Octafian, 2011 :152)

3. *Many to Many*

Sebuah entitas pada A berhubungan dengan lebih dari satu entitas pada B dan sebuah entitas pada B berhubungan dengan lebih dari satu entitas pada A. Contoh: dalam satu universitas, seorang mahasiswa

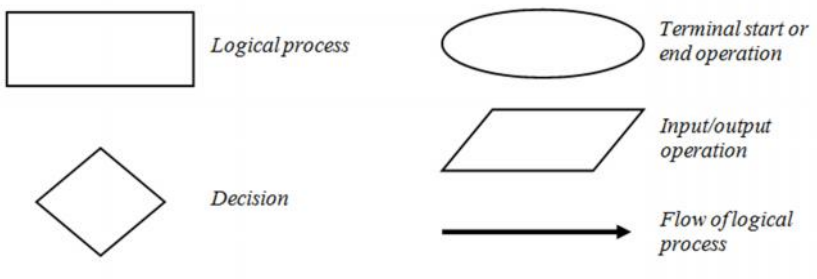
dapat mengambil banyak matakuliah. Satu mahasiswa mengambil banyak matakuliah dan satu matakuliah diambil banyak mahasiswa. Hubungan *Many to Many* dapat dilihat pada Gambar II.7.



Gambar II.7 Hubungan *Many to Many*
(Sumber: D. Tri Octafian, 2011 : 152)

II.9 *Flowchart*

Flowchart merupakan urutan-urutan langkah kerja suatu proses yang digambarkan dengan menggunakan simbol-simbol yang disusun secara sistematis (Eka Iswandy, 2014 : 30). *Flowchart* merupakan gambaran secara grafik dari langkah-langkah dan urutan prosedur suatu program. Biasanya mempermudah penyelesaian masalah, khususnya yang perlu dipelajari dan dievaluasi lebih lanjut (Indrajani, 2015 : 36). Adapun bentuk atau notasi program *flowchart* dapat dilihat pada Gambar II.8.



Gambar II.8 Notasi Program *Flowchart*
(Sumber: Indrajani, 2015 : 38)

II.10 *Unified Modeling Language* (UML)

II.10.1 Pengertian UML

Unified Modeling Language (UML) adalah standar *de-facto* untuk pemodelan objek dan sistem perangkat lunak berorientasi. UML menyediakan diagram untuk mewakili sifat statis serta perilaku dinamis dari suatu sistem (Sabharwal, Sangeeta, et al., 2011 : 433).

UML dianggap sebagai industri bahasa pemodelan standar dengan notasi grafis yang kaya dengan seperangkat diagram dan elemen. Hal ini digunakan untuk menentukan, memvisualisasikan, memodifikasi, membangun, dan mendokumentasikan bentuk dari perangkat lunak-intensif berorientasi objek dalam membangun sistem (Lee, Sunguk, 2012 : 157).

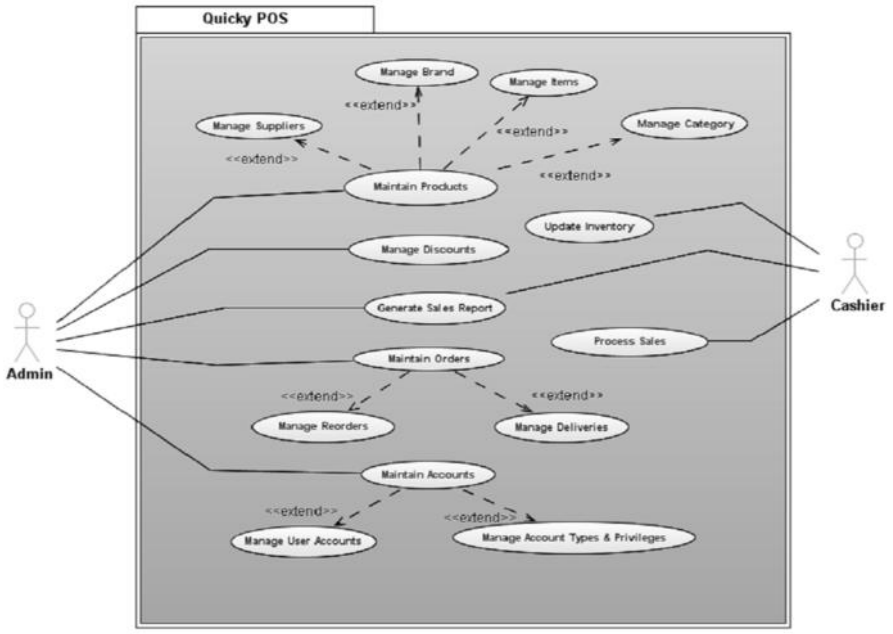
II.10.2 Diagram UML

UML memiliki beberapa jenis diagram, namun dalam penulisan skripsi ini penulis hanya menggunakan 4 jenis diagram UML yaitu: *Use Case Diagram*, *Class Diagram*, *Activity Diagram*, dan *Sequence Diagram*. Adapun penjelasan dari diagram-diagram tersebut adalah sebagai berikut:

1. *Use Case Diagram*

   Use case diagram merupakan suatu diagram yang berisi *use case, actor,* serta *relationship* diantaranya. *Use case diagram* merupakan titik awal yang baik dalam memahami dan menganalisis kebutuhan sistem pada saat perancangan (Indrajani, 2015 : 45). Fungsi yang disediakan oleh sistem *database* atau aplikasi komputer dapat digambarkan dengan *diagram use case*. Tujuan utamanya adalah

untuk memvisualisasikan persyaratan fungsional dari suatu sistem, termasuk hubungan *actor* (manusia yang akan berinteraksi dengan sistem) untuk proses yang penting, serta hubungan antara kasus penggunaan yang berbeda untuk mencapai suatu tujuan (Lee, Sunguk, 2012 : 159). Adapun contoh penggunaan *use case diagram* dapat dilihat pada Gambar II.9.

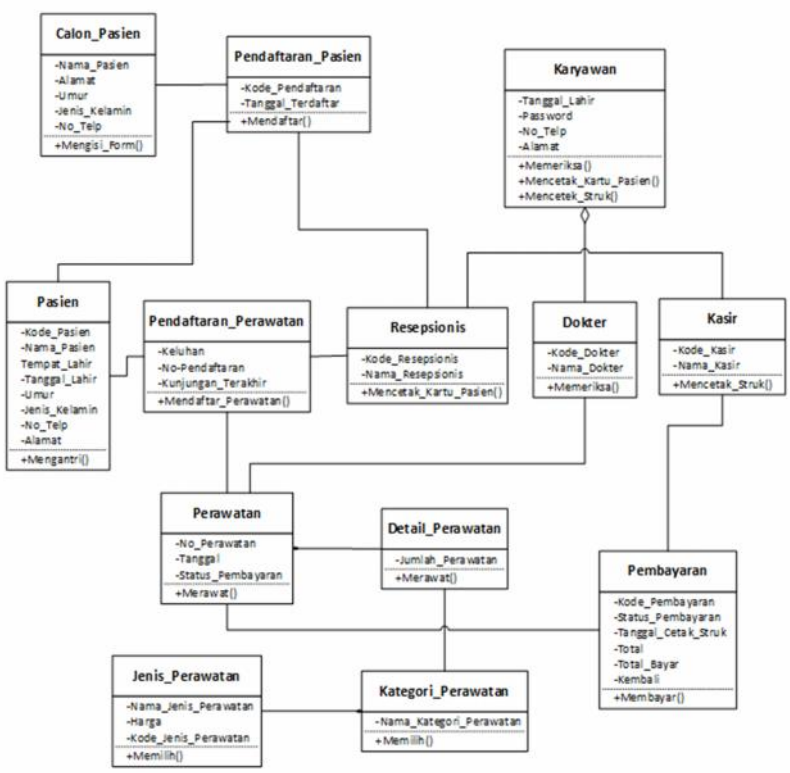


Gambar II.9 *Sample Use Case Diagram of a Point of Sale*
(Sumber: Lee, Sunguk, 2012 : 160)

2. *Class Diagram*

Diagram ini digunakan untuk menggambarkan perbedaan yang mendasar antara *class-class,* hubungan antar *class*, dan di mana sub-sistem *class* tersebut. Pada *class diagram* terdapat nama *class*, *attributes*, *operations*, serta *association* (hubungan antar *class*) (Indrajani, 2015 : 49). *Class diagram* merupakan struktur statis aplikasi komputer atau stasiun *database* yang ditampilkan dalam

diagram class. Hal ini juga menunjukkan bagaimana entitas yang berbeda (orang, benda, dan data) berhubungan satu sama lain (Lee, Sunguk, 2012 : 160). Adapun contoh penggunaan *class diagram* dapat dilihat pada Gambar II.10.

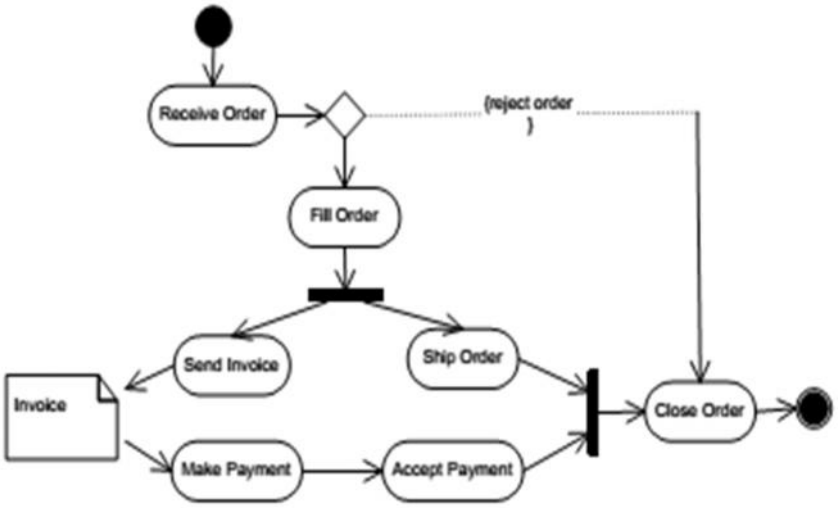


Gambar II.10 *Sample Class Diagram of Hospital Management*
(Sumber: Indrajani, 2015 : 50)

3. *Activity Diagram*

Diagram ini digunakan untuk menganalisis *behavior* dengan *use case* yang lebih kompleks dan menunjukkan interaksi-interaksi di antara mereka satu sama lain (Indrajani, 2015 : 46). Aliran prosedural kontrol antara dua atau lebih objek kelas saat memproses suatu kegiatan dapat ditampilkan dengan *activity diagram*. Hal ini dapat digunakan untuk model yang lebih tinggi seperti proses ditingkat unit bisnis atau untuk

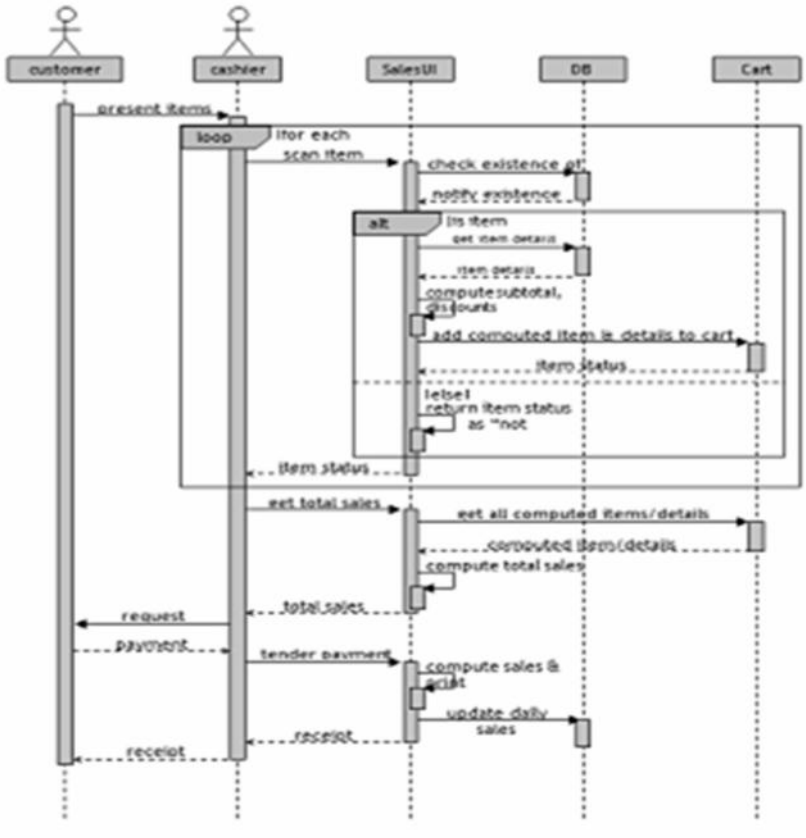
model tingkat rendah seperti tindakan kelas internal (Lee, Sunguk, 2012 : 161-162). Adapun contoh penggunaan *activity diagram* dapat dilihat pada Gambar II.11.



Gambar II.11 ***Sample Activity Diagram of a Point of Sale (Processing Order)***
(Sumber: Lee, Sunguk, 2012 : 162)

4. *Sequence Diagram*

Diagram ini merupakan suatu diagram interaksi yang menggambarkan bagaimana objek-objek berpartisipasi dalam bagian interaksi (*particular interaction*) dan pesan yang ditukar dalam urutan waktu (Indrajani, 2015 : 50). *Sequence diagram* menunjukkan aliran rinci untuk kasus tertentu atau bahkan hanya bagian dari penggunaan kasus tertentu. Ini menunjukkan panggilan antara objek yang berbeda secara berurutan yang dapat ditampilkan, dan pada tingkat rinci panggilan yang berbeda untuk objek yang berbeda pula (Lee, Sunguk, 2015 : 160). Adapun contoh penggunaan *sequence diagram* dapat dilihat pada Gambar II.12.



Gambar II.12 ***Sample Sequence Diagram of a Point of Sale (Processing Sales)***
(Sumber: Lee, Sunguk, 2012 : 161)

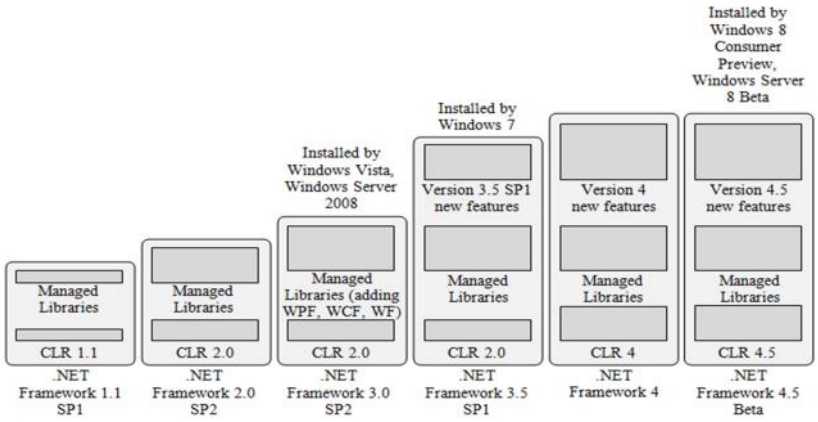
II.11 *Microsoft Visual Studio 2010*

II.11.1 Definisi *Microsoft Visual Studio 2010*

Microsoft Visual Studio 2010 merupakan sebuah perangkat lunak yang dapat digunakan untuk melakukan pengembangan aplikasi, yang di dalamnya terdapat beberapa kumpulan *tools* pemrograman seperti *Visual Basic .NET, Visual C++ .NET, Visual C# .NET,* dan *Visual J# .NET*. Namun yang paling umum digunakan yaitu *Visual Basic .NET*. *Visual Basic .NET* adalah *Visual Basic* yang direkayasa kembali untuk digunakan pada *platform .NET* sehingga aplikasi yang

dibuat menggunakan *Visual Basic .NET* dapat berjalan pada sistem komputer apapun, dan dapat mengambil data dari *server* dengan tipe apapun asalkan terinstal *.NET Framework* (Priyanto Hidayatullah, 2012 : 5-8).

Pemrograman *Visual Basic .NET* adalah bahasa pemrograman terpopuler. Ini merupakan pemrograman yang berjalan di atas *platform NET Framework*. Karena itu setiap kali pemrograma *VB .NET* ini merilis versi barunya, tentu saja akan diikuti atau berbarengan dengan perkembangan *NET Framework* terbaru (Edy Winarno, et al., 2013 : 141). Perkembangan teknologi *VB .NET* untuk lebih jelasnya dapat dilihat pada Gambar II.13.

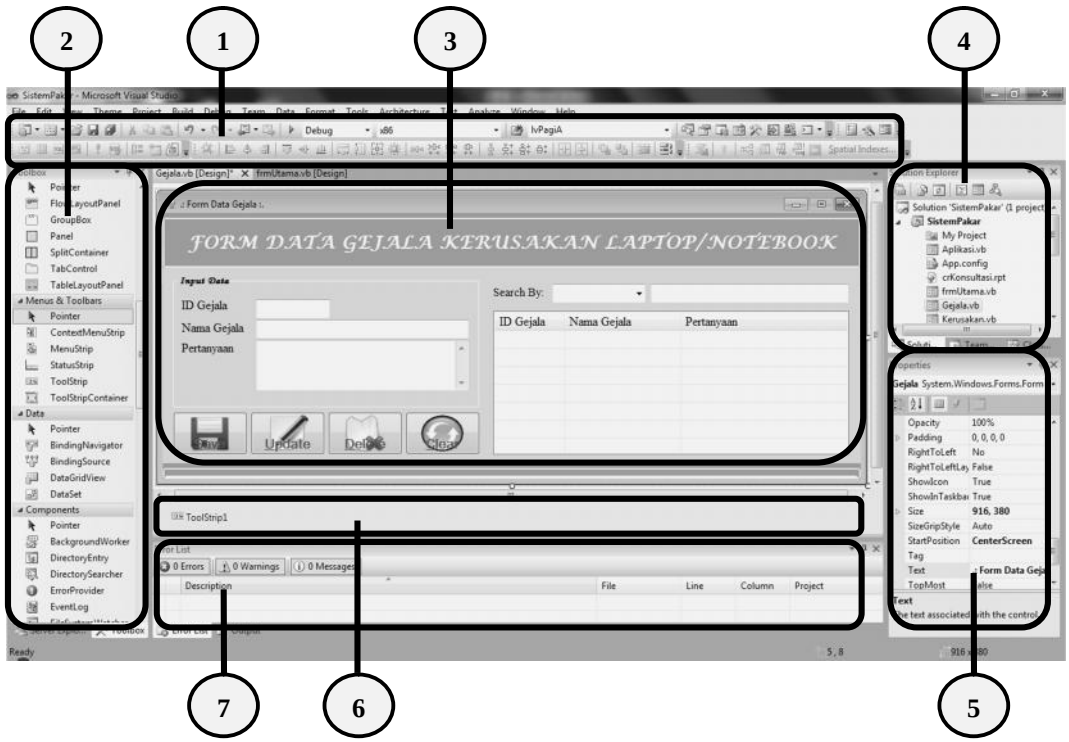


Gambar II.13 Perkembangan Teknologi *VB .NET*
(Sumber: Edy Winarno, et al., 2013 : 142)

II.11.2 Mengenal IDE *Visual Studio 2010*

Pada IDE (*Integrated Developing Environment*) *Visual Studio 2010* untuk *Windows Application* secara *default* telah terdapat sebuah *form*. *Form* tersebut bernama *Form1*. Pada *form* inilah tempat meletakkan kontrol-kontrol atau komponen-komponen untuk membuat sebuah aplikasi *Windows Form* dan kontrol-kontrol dari aplikasi inilah yang biasanya disebut dengan GUI (*Graphical*

User Interface). Jadi *user* akan berinteraksi dengan sebuah program aplikasi melalui GUI (Priyanto Hidayatullah, 2012 : 24). Adapun tampilan daripada *Visual Basic 2010* dapat dilihat pada Gambar II.14.



Gambar II.14 *Integrated Developing Environment of Visual Basic 2010*
(Sumber: Dodit Suprianto, 2010 : 15)

Dari Gambar II.14 dapat dilihat IDE daripada *Visual Basic 2010,* adapun keterangan dari gambar tersebut adalah sebagai berikut:

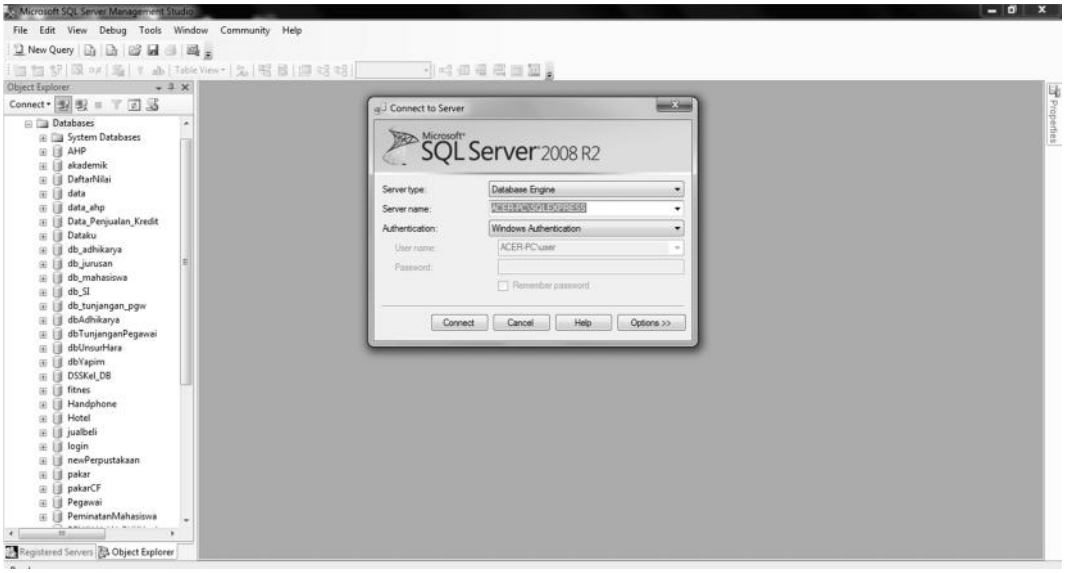
1. ***Toolbar,*** terdiri dari ikon-ikon sebagai jalan pintas (*shortcut*) pengganti menu *pulldown*. Ikon *toolbar* berisi perintah-perintah yang sering digunakan oleh *programmer* sehingga lebih cepat mengoperasikan perintah-perintah yang ada.
2. ***Toolbox,*** berisi kontrol *object* terutama pembentuk antarmuka (*interface*) *windows*, seperti *textbox*, *panel*, *button*, *groupbox*, dan lain sebagainya.

3. ***Form Design,*** sebuah *Form* akan terbentuk dengan sendirinya saat *project* dibuat pertama kali. *Form design* ini merupakan *Graphical User Interface* yang dirancang.
4. ***Solution Explorer,*** terdiri dari *object-object* pendukung *project*, seperti *form, module, report, data control,* dan lain sebagainya.
5. ***Properties,*** menampilkan *property* setiap *object* yang terpilih. Dari *Windows Property* dapat mengubah atau mengatur nilai masing-masing *properties object*.
6. ***Status Object,*** menampilkan daftar kontrol yang telah digunakan oleh *Form* bersangkutan. Dari kasus di atas, kontrol yang digunakan adalah *ToolStrip1*.
7. ***Error List,*** merupakan *window* yang akan menampilkan pesan tertentu jika terjadi kesalahan atau *error* pada saat proses pengembangan program atau kompilasi *program*.

II.12 *Microsoft SQL Server 2008 R2*

Microsoft SQL Server merupakan produk *Relational Database Management System* (RDBMS) yang dibuat oleh *Microsoft*. Pada tahun 2008 *Microsoft* mengeluarkan *SQL Server 2008 R2* yang merupakan versi yang banyak digunakan. *SQL* (*Structured Query Language*) adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Saat ini hampir semua *server* basis data yang ada mendukung bahasa

ini untuk melakukan manajemen datanya. *SQL* terdiri dari dua bahasa, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML). Implementasi DDL dan DML berbeda untuk tiap Sistem Manajemen Basis Data (SMBD), namun secara umum implementasi setiap bahasa ini memiliki bentuk standar yang ditetapkan oleh ANSI (Adelia dan Jimmy Setiawan, 2011 : 115). Adapun tampilan daripada *Microsoft SQL Server 2008 R2* dapat dilihat pada Gambar II.15.



Gambar II.15 ***Integrated Developing Environment of Microsoft SQL Server 2008 R2***
(Sumber: Priyanto Hidayatullah, 2012 : 187)