BAB II

LANDASAN TEORI

II.1. Kecerdasan Buatan

Kecerdasan buatan berasal dari bahasa inggris "*Artificial Intelligence"* atau disebut AI, yaitu *Intelligence* adalah kata sifat yang berarti cerdas, sedangkan *Artificial* yang berarti buatan. Kecerdasan buatan yang dimaksud di sini merujuk pada mesin yang mampu berfikir, menimbang tindakan yang akan di ambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia.

Menurut Herbertb Alexsander Simon (June, 15, 1916-February 9,1001) "kecerdasan buatan (articial intellilagence) merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemograman komputer untuk melakukan sesuatu hal yang dalam pandangan manusia adalah cerdas".
(T. Sutojo; 2011: 1-2).

II.2. Sistem Pakar

Bidang sistem pakar merupakan penyelesaian pendekatan yang sangat berhasil dan bagus untuk permasalahan AI klasik dari pemprograman *intelligent* (cerdas). Sistem Pakar *(expert system)* merupakan solusi AI *(Artificial Intelligence)* bagi permasalahan pemrograman pintar. Profesor Edward Feigenbaum dari Stanford University yang merupakan pionir dalam teknologi sistem pakar mendefinisikan sistem pakar sebagai sebuah program komputer pintar (*Intelligent Computer Program*) yang memanfaatkan pengetahuan

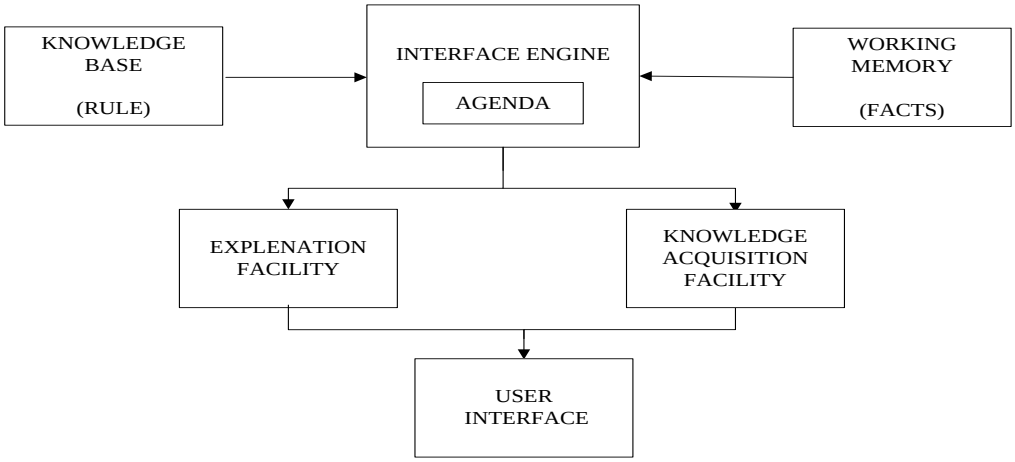
(*knowledge)* dan prosedur inferensi (*inference procedure*) untuk memecahkan masalah yang cukup sulit sehingga membutuhkan keahlian khusus dari manusia. (Rika Rosnelly; 2012: 2).

Secara umum, sistem pakar (*Expert System*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer, agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Dengan sistem pakar ini, orang awam pun dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli. Tujuan pengembangan sistem pakar sebenarnya tidak untuk menggantikan peran para pakar, namun untuk mengimplementasikan pengetahuan para pakar ke dalam bentuk perangkat lunak, sehingga dapat digunakan oleh banyak orang dan tanpa biaya yang besar. Untuk membangun sistem yang difungsikan untuk menirukan seorang pakar manusia harus bisa melakukan hal-hal yang dapat dikerjakan oleh para pakar. Untuk pembangun sistem yang seperti itu maka komponen-komponen dasar yang minimal harus dimiliki adalah sebagai berikut:

1. Antar muka (user interface).
2. Basis pengetahuan (knowledge base).
3. Mesin inferensi (Inference Engine). (Yasidar Nur Istiqomah; 2013 : 34).

II.2.1. Struktur Sistem Pakar

Adapun struktur sistem pakar dapat dilihat pada Gambar II.I.



Gambar II.1 : Struktur Sistem Pakar
(Sumber : Rika Rosnelly; 2012: 12)

Komponen yang terdapat dalam struktur sistem pakar ini adalah *knowledge base (rules), inference engine, working memory, explanation facility, knowledge acquisition facility, dan user interface*. (Rika Rosnelly; 2012: 13).

II.2.2. Pakar

Pakar adalah seorang individu yang memiliki pengetahuan khusus, pemahaman, pengalaman, dan metode-metode yang digunakan untuk memecahkan persoalan dalam bidang tertentu.

Seorang pakar memiliki kemampuan kepakaran, yaitu :

1. Dapat mengenali dan merumuskan suatu masalah.
2. Menyelesaikan masalah dengan cepat dan tepat.
3. Menjelaskan solusi dari suatu masalah.
4. Restrukturisasi pengetahuan.
5. Belajar dari pengalaman.

6. Memahami batas kemampuan. (Rika Rosnelly; 2012: 10).

II.2.3. Manfaat Sistem Pakar

Sistem pakar menjadi sangat populer karena sangat banyak manfaat dari sistem pakar, diantaranya :

1. Meningkatkan produktifitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seseorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Dapat beroperasi di lingkungan yang berbahaya.
6. Memudahkan akses pengetahuan seorang pakar.
7. Andal. Sistem pakar tidak pernah menjadi bosan dan kelelahan atau sakit.
8. Meningkatkan kapabilitas sistem komputer. Integrasi sistem pakar dengan sistem komputer lain membuat sistem lebih efektif dan mencakup lebih banyak aplikasi.
9. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti.
10. Bisa digunakan sebagai media pelengkap dalam penelitian.
11. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar.

    (T. Sutojo; 2010: 160).

II.2.4. Kekurangan Sistem Pakar

Selain manfaat, ada juga beberapa kekurangan yang ada pada sistem pakar, di antaranya :

1. Biaya yang sangat mahal untuk membuat dan memeliharanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan memeliharanya.
3. Sistem pakar tidak 100% bernilai benar.

II.3. Demam Bedarah Dengue (DBD)

Penyakit demam dengue atau demam berdarah merupakan penyakit infeksi yang disebabkan oleh virus dengue dan ditularkan melalui gigitan Nyamuk *Aedes aegepty* dan *Aedes albopictus*. Penyakit ini merupakan salah satu jenis gangguan kesehatan yang mengganggu produktivitas setiap orang dan merupakan salah satu penyakit menular yang sering menimbulkan wabah dan menyebabkan kematian. Oleh karena itu penyakit ini sering menimbulkan kepanikan masyarakat. Seorang yang menderita penyakit demam berdarah pada awalnya akan menderita demam tinggi. Dalam keadaan demam ini tubuh banyak kekurangan cairan karena terjadinya penguapan yang lebih banyak dari pada biasanya. Gejala penyakit demam berdarah selama ini hanya didiagnosa masyarakat awam berdasarkan ciri-ciri yang diketahui tanpa oleh fakta dan pertimbangan medis lainnya. Sehingga masyarakat atau penderita sulit membedakan penyakit demam berdarah dengan penyakit-penyakit demam biasa pada umumnya. Akibatnya penyakit tersebut ditangani dengan cara yang salah. Oleh karena itu agar tidak ada kesalahan diagnosa dan untuk

mempermudah masyarakat atau penderita mengetahui sejak dini penyakit yang diderita dan agar tidak terlambat mendapatkan pengobatan dikarenakan seorang dokter atau pakar memiliki keterbatasan waktu. Maka dibangun suatu sistem yang dapat membantu menyelesaikan masalah tersebut berupa sistem pakar dengan menggunakan metode *Dempster-Shafer*. (Nur Anjas Sari; 2013).

II.4. *Dempster-Shafer*

Ada berbagai macam penalaran dengan model yang lengkap dan sangat konsisten, tetapi pada kenyataannya banyak permasalahan yang tidak dapat terselesaikan secara lengkap dan konsisten. Ketidak konsistenan yang tersebut adalah akibat adanya penambahan fakta baru. Penalaran yang seperti itu disebut dengan penalaran *non monotonis*. Untuk mengatasi ketidak konsistenan tersebut maka dapat menggunakan penalaran dengan teori *Dempster-Shafer*. Secara umum teori *Dempster-Shafer* ditulis dalam suatu interval.

Penulisan umum :

[*belief, plausibility*]

1. *Belief* (Bel).

Belief adalah ukuran kekuatan *evidence* dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada *evidence*, dan jika bernilai 1 menunjukkan adanya kepastian.

2. *Plausibility* (P1) dinotasikan sebagai :

$$Pl(s) = 1 - Bel(\neg s)$$

Plausibility juga bernilai 0 sampai 1. Jika yakin akan ¬s, maka dapat dikatakan bahwa Bel(¬s) = 1, dan PI(¬ s) = 0.

Pada teori *Dempster-Shafer* dikenal adanya *frame of discernment* yang dinotasikan dengan θ. Frame ini merupakan semesta pembicaraan dari sekumpulan hipotesis. Tujuannya adalah mengkaitkan ukuran kepercayaan elemen-elemen θ. Tidak semua *evidence* secara langsung mendukung tiap-tiap elemen. Untuk itu perlu adanya probabilitas fungsi densitas (m). Nilai m tidak hanya mendefinisikan elemen-elemen θ saja, namun juga semua subsetnya. Sehingga jika θ berisi n elemen, maka subset θ adalah 2n. Jumlah semua m dalam subset θ sama dengan 1. Apabila tidak ada informasi apapun untuk memilih hipotesis, maka nilai: m{θ} = 1,0.

Apabila diketahui X adalah subset dari θ, dengan m1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari θ dengan m2 sebagai fungsi densitasnya, maka dapat dibentuk fungsi kombinasi m1 dan m2 sebagai m3, yaitu : (Muhammad Dahria :2013:4).

$$m_3(Z) = \frac{\sum_{X \cap Y = Z} m_1(X).m_2(Y)}{1 - \sum_{X \cap Y = \phi} m_1(X).m_2(Y)}$$

Keterangan :

m = Nilai densitas (kepercayaan)

XYZ = Himpunan evidence

$\sum_{X \cap Y = Z} m_1(X).m_2(Y)$ = merupakan nilai kekuatan dari *evidence* Z yang diperoleh nilai keyakinan sekumpulan *evidence*.

Contoh Kasus :

Untuk menganalisis gejala-gejala yang diberikan oleh pasien untuk mendapatkan kemungkinan nama penyakitnya, dilakukan dengan menghitung nilai densitas dari gejala dengan menghitung nilai kepercayaan menggunakan rumus Demster-shafer. Seorang pasien mengalami gejala penyakit pada kelapa sawit dengan diagnosa dokter penyakit yang mungkin dideritanya adalah Garis Kuning, Busuk Kuncup atau Tanjuk.

Gejala-1 : Daun tampak mengering dan gugur

Apabila diketahui nilai kepercayaan setelah dilakukan observasi nyeri ulu hati sebagai gejala dari penyakit Garis Kuning (GK) dan Busuk Kuncup (BK) adalah:

m1{GK, BK,DBM} = 0,8

m1{θ} = 1 – 0,8 = 0,2

Gejala-2 : Tampak Bercak-bercak lonjong berwarna kuning dan ditengahnya berwarna coklat Kemudian jika diketahui nilai kepercayaan setelah dilakukan observasi terhadap rasa terbakar pada tenggorokan sebagai gejala dari penyakit Tanjuk(PT) adalah :

m2{GK, BK, PT} = 0,9

m2{θ} = 1 – 0,9 = 0,1

Munculnya gejala baru mengharuskan kita untuk menghitung densitas baru untuk beberapa kombinasi (m3) dengan aturan seperti pada Tabel 1.

Tabel II.1. Aturan Kombinasi

	{GB,BK,PT} (0,9) θ (0,1)
{BK,PT,DBM}(0,8)	{BK,PT} (0,72) {M, G}(0,08)
Θ(0,2)	{GB,BK,PT} (0,18) θ (0,02)

(Sumber : Maruli Tua Nahampun; 2014: 57)

{BK,PT} di peroleh dari irisan antara {GK,BK,PT} dan {BK,PT,DBM}. Nilai 0,72 di peroleh dari hasil perkalian 0,9x0,8. Demikian pula {BK,PT,DBM} pada baris Kedua Kolom merupakan irisan dari Θdan {D,M,G} pada baris kedua kolom pertama. Hasil 0,08 merupakan perkalian dari 0,1x0,8. Sehingga diperoleh m3 :

$$m3\{BK,PT\} = \frac{0,72}{1-(0)} = 0,72$$

$$m3\{GB,BK,PT\} = \frac{0,18}{1-(0)} = 0,18$$

$$m3\{BK,PT,DBM\} = \frac{0,08}{1-(0)} = 0,08$$

$$m3\{0\} = \frac{0,02}{1-(0)} = 0,02$$

Gejala paling kuat: {BK,PT}= penyakit Maag dan Dispepsia = 0,72

Gejala-3 : Mual

Kemudian jika diketahui nilai kepercayaan setelah dilakukan observasi terhadap rasa terbakar pada tenggorokan sebagai gejala dari penyakit *Gerd* adalah:

m4{GK} = 0,6

m4{Θ} = 1-0,6=0,4

maka kita harus menghitung kembali nilai densitas baru untuk setiap himpunan bagian dengan fungsi densitas m5. Seperti pada tabel 3.4 hasil kombinasi dari

gejala-1 & gejala-2 dengan fungsi densitas m3. Sedangkan baris pertama berisi himpunan bagianhimpunan bagian gejala-3 dengan fungsi densitas m4.

Tabel II.2.Aturan kombinasi untuk m5

	{GB} (0,6) θ(0,4)
{BK,PT}(0,72)	Θ(0,432) {BK,PT}(0,288)
{GB,BK,PT}(0,18) {GB}(0,108){GB,BK,PT}(0,072)	{GB}(0,108){GB,BK,PT} (0,072)
{BK,PT,DBM}(0,08)	Θ(0,048) {BK,PT,DBM}(0,032)
Θ(0,02)	{GB} (0,012) Θ (0,008)

(Sumber : Maruli Tua Nahampun; 2014: 58)

Sehingga dapat dihitung :

$$m_5\{GB\} = \frac{0,288}{1-(0,432+0,048)} = 0,554$$

$$m_5\{BK,PT\} = \frac{0,72}{1-(0,432+0,048)} = 0,138$$

$$m_5\{GB,BK,PT\} = \frac{0,108+0,012}{1-(0,432+0,048)} = 0,231$$

$$m_5\{BK,PT,DBM\} = \frac{0,032}{1-(0,432+0,048)} = 0,062$$

$$m_5\{\Theta\} = \frac{0,008}{1-(0,432+0,048)} = 0,015$$

II.5. *Entity Relationship Diagram* (ERD)

Entity Relationship Diagram adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antar entitas. Proses memungkinkan analis menghasilkan struktur basis data yang baik sehingga data dapat disimpan dan diambil secara efisien, untuk menggambarkannya digunakan beberapa notasi dan simbol. Pada dasarnya ada tiga simbol yang digunakan, yaitu (Janner Simarmata; 2010: 67).

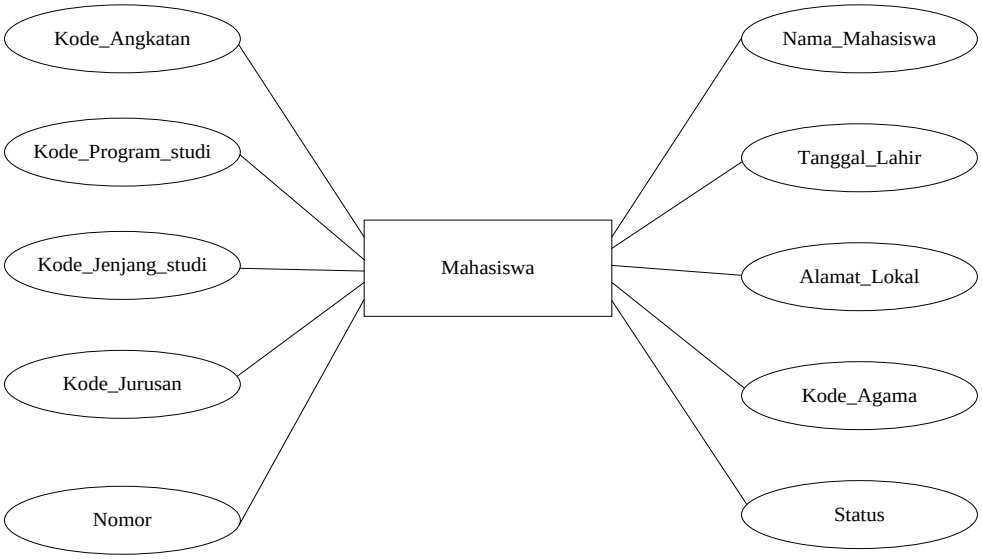
1. Entitas, adalah sesuatu yang nyata atau abstrak di mana kita akan menyimpan data.

Tabel II.3. Contoh Entitas Berupa Orang

Objek Dasar	Simbol Entitas
Mahasiswa	Mahasiswa
Dosen	Dosen

(Sumber : Janner Simarmata; 2010: 67)

2. Atribut, adalah ciri umum semua atau sebagian besar instansi pada entitas tertentu. Sebutan lain atribute adalah properti, elemen data, dan field.



Gambar II.2. Contoh Atribut pada Entitas Mahasiswa
(Sumber : Janner Simarmata; 2010: 67)

3. Relasi, adalah hubungan alamiah yang terjadi antara satu atau lebih entitas, misalnya proses pembayaran pegawai. Kardinalitas menentukan kejadian suatu entitas untuk satu kejadian pada entitas yang berhubungan. Misalnya

Gambar II.3. Contoh Kerelasian antara entitas Mahasiswa dan Mata_kuliah
(Sumber : Janner Simarmata; 2010: 67)

Relasi yang terjadi diantara dua himpunan entitas (misalnya A dan B) dalam satu basis data yaitu:

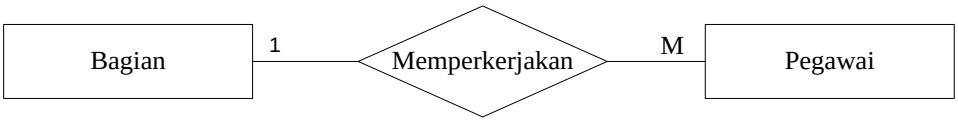
a. ***Satu ke satu (One to one),*** sebuah entitas pada A berhubungan paling banyak satu entitas pada B dan sebuah entitas pada B berhubungan paling banyak satu entitas pada A.

Gambar II.4. Hubungan *One to one*
(Sumber : Janner Simarmata; 2010: 67)

Keterangan :

1. Pada pengajaran privat, satu guru satu siswa.
2. Seorang guru mengajar seorang siswa, seorang siswa diajar oleh seorang guru.

b. ***Satu ke banyak (One to Many),*** sebuah entitas pada A dapat berhubungan dengan banyak entitas pada B. Tetapi setiap entitas pada B dapat dihubungkan dengan satu entitas pada A.



Gambar II.5. Hubungan *One to Many*
(Sumber : Janner Simarmata; 2010: 67)

Keterangan :

1. Dalam suatu perusahaan, satu bagian memperkerjakan banyak pegawai.
2. Satu bagian memperkerjakan banyak pegawai, satu pegawai kerja dalam satu bagian.

c. ***Banyak ke banyak (Many to many),*** sebuah entitas pada A dapat berhubungan dengan banyak entitas pada B.

Gambar II.6. Hubungan *One to Many*
(Sumber : Janner Simarmata; 2010: 67)

Keterangan :

1. Dalam universitas, seorang mahasiswa dapat mengambil banyak mata kuliah.
2. Satu mahasiswa mengambil banyak mata kuliah dan satu mata kuliah diambil banyak mahasiswa.

II.5.1. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan

menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relasional.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang telah ditemukan.

1. Bentuk Nomal Pertama (1NF)

   Contoh yang kita gunakan di sini adalah sebuah perusahaan yang mendapatkan barang dari sejumlah pemasok, masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri.

2. Bentuk Normal Kedua (2NF)

   Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada INF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada kunci utama, ini berarti bahwa setiap kolom bukan kunci harus tergantung pada seluruh kolom yang membentuk kunci utama.

3. Bentuk Normal Ketiga (3NF)

   Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada pada 2NF dan setiap utamanya. Dengan kata lain, semua atribut bukan kunci tergantung secara fungsional hanya pada kunci utama.

4. Bentuk Normal Keempat (4NF)

   Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD). Sebuah ketergantungan multivalue terjadi ketika dalam sebuah tabel relasional yang mengandung setidaknya tiga kolom, satu kolom mempunyai banyak baris bernilai sama.

5. Bentuk Normal Kelima (5NF)

   Sebuah tabel berada pada bentuk normal kelima (5NF) jika dia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan. (Janner Simarmata; 2010: 77).

II.6. Kamus Data

Kamus data (data dictionary) dipergunakan untuk memperjelas aliran data yang digambarkan pada DFD. Kamus data adalah kumpulan daftar elemen data yang mengalir pada sistem perangkat lunak sehingga masukan (input) dan keluaran (output) dapat dipahami secara umum (memiliki standar cara penulisan). Kamus data biasanya berisi:

1. Nama - nama dari data
2. Digunakan pada – merupakan proses-proses yang terkait data
3. Deskripsi – merupakan deksripsi data
4. Informasi tambahan – seperti tipe data, nilai data, batas nilai data, dan komponen yang membentuk data. (Rosa A.S; 2011: 67).

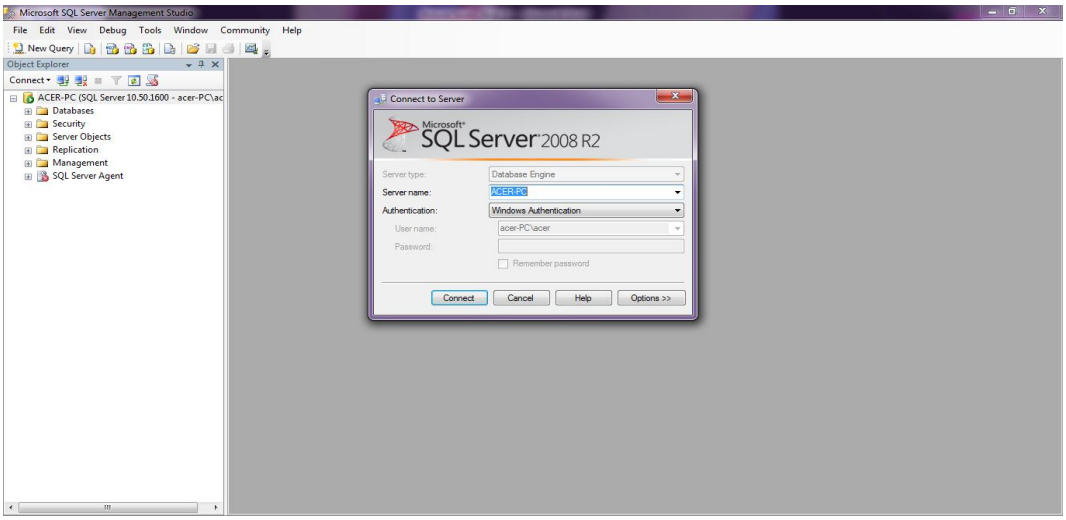
Tabel II.4. Contoh Kamus Data

No	Nama Field	Tipe Data	Panjang
1	Id_pelanggan	CHAR	5
2	Nama	VARCHAR	30
3	Alamat	VARCHAR	60
4	Telp	VARCHAR	15

(Sumber : Rosa A.S; 2011: 67)

II.7. Pengertian SQL Server 2008

SQL Server 2008 adalah sebuah terobosan baru dari Microsoft dalam bidang database. SQL Server adalah sebuah *DBMS (Database Management System)* Yang dibuat oleh *Microsof*t untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan *Oracle. Sql server 2008* dibuat pada saat kemajuan dalam bidang hardware sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server 2008* membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data. (Wenny Widya; 2010 : 3).

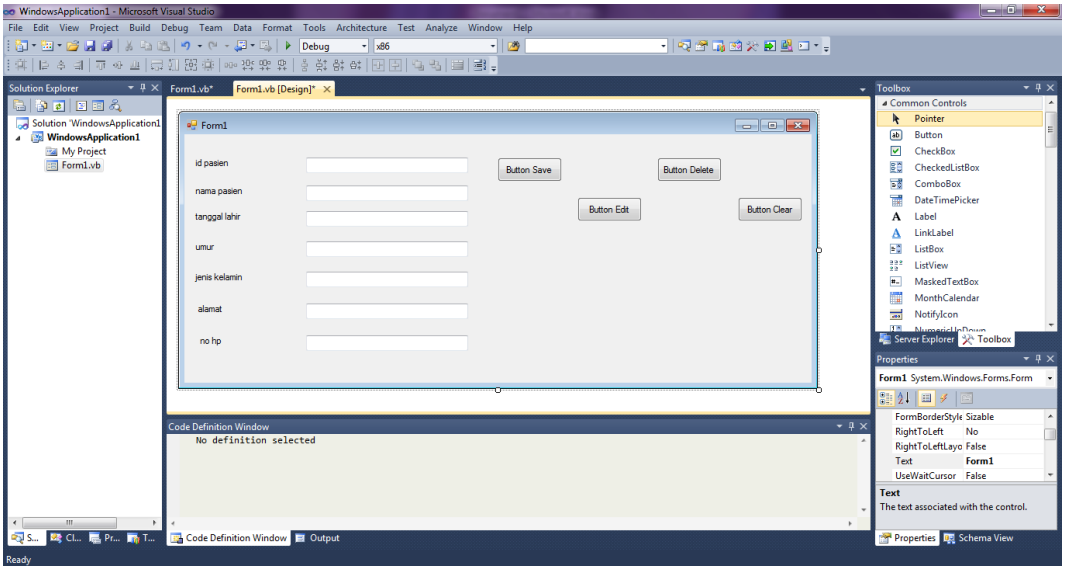


Gambar II.7. Tampilan Microsof SQL Server 2008 R2
(Sumber : Wenny Widya; 2010)

II.8. Pemrograman Visual Basic 2010

Visual Basic 2010 merupakan suatu paket teknologi bahasa pemrograman yang dikeluarkan oleh microsoft. Bahasa pemrograman Visual Basic digunakan untuk membuat aplikasi windows yang berbasis *Graphical User Interface* (GUI). Microsoft Visual Basic 2010 sebagai produk IDE (*Integrated Development Environments)* andalan yang dikeluarkan Microsoft. Microsoft Visual Studio 2010 telah menambahkan berbagai pembaruan dan perbaikan fitur-fitur untuk melengkapi fitur versi sebelumnya.

Framework terbaru, yaitu .Net Framework 3.5 yang merupakan pengembangan sebelumnya dari Net Framework 4.0. Database standart yang disertai dalam Visual Studio 2010 adalah Microsoft SQL server 2010 Express. (Wahana Komputer; 2013: 2).



Gambar II.8. Tampilan Microsoft Visual Studio 2010
(Sumber : Wahana Komputer; 2013)

II.9. Perancangan UML

UML sesuai dengan kata terakhir dari kepanjangannya, UML itu adalah salah satu bentuk *language* atau bahasa. Menurut pencetusnya, UML didefenisikan sebagai bahasa visual untuk menjelaskan, memberikan spesifikasi, membuat model, dan mendokumentasikan aspek-aspek dari sebuah sistem. Karena tergolong bahasa visual, UML lebih mengedepankan penggunaan diagram untuk menggambarkan aspek dari sistem yang sedang dimodelkan. (Yuni Sugiarti; 2013: 36).

Unified Modelling Language (UML) biasanya digunakan untuk:

1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum.
2. Menggambarkan kegiatan atau proses bisnis yang dilaksakan secara umum.
3. Menggambarkan representasi stuktur statik sebuah sistemdalam bentuk *class diagram*.
4. Membuat model *behavior* "yang menggambarkan sifat atau sebuah sistem.
5. Menyatakan arsitektur implementasi fisik menggunakan *component* dan *development diagram.*
6. Menyampaikan atau memperluas *fungsionalty* dengan *stereotypes.* (Yuni Sugiarti; 2013: 36).

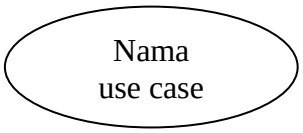
II.9.1 Jenis-Jenis Diagram UML

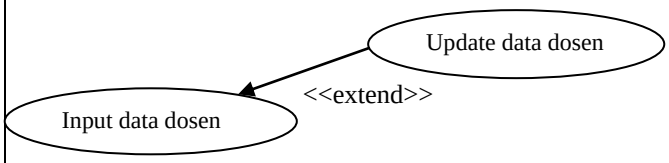
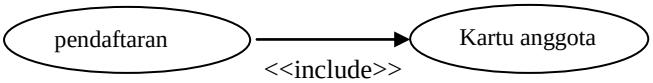
Berikut ini adalah mengenai berbagai diagram UML serta tujuan dan simbol-simbolnya, yaitu :

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

Use case diagram secara garis besar menggambarkan interaksi antara sistem, sistem ekstrenal, dan pengguna. Dengan kata lain secara garis besar *use case diagram* secara grafis menggambarkan siapa yang akan menggunakan sistem dan dengan cara apa pengguna (user) mengharapkan interaksi dengan sistem itu. Terdapat beberapa symbol dalam menggambarkan diagram use case, yaitu use case, actor, relasi. (Yuni Sugiarti; 2013: 41).

Tabel II.5. Simbol *Use Case*

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dengan menggunakan kata kerja di awal frase nama *use case*.
Aktor  Nama aktor	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, tapi aktor belum tentu adalah orang, biaanya dinyatakan menggunakan kata benda di awal fase nama aktor.
Asosiasi / association _____	Komunikasi antara aktor dan use case yang berpartisipasi pada use case atau use case miliki interaksi dengan aktor
Extend  <<extend>>	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 cccase yang di bambahkan, arah panah menunjukkan pada use case yang dituju.contoh: 
Include <<include>>	Relasi use case tambahan sebuah use case di mana 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 use case, include berarti use case yang ditambahkan akan selalu di panggil saat use case tambahan dijalankan. Contoh: 

(Sumber: Yuni Sugiarti; 2013: 42)

2. *Class Diagram* (Diagram Kelas)

Diagram kelas atau c*lass diagram* menggambarkan struktur sistem dari pendefenisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang dimaksud dengan atribut dan metode atau operasi.

Diagram kelas menggambarkan struktur dan deskripsi *class, package* dan objek beserta hubungan satu sama lain seperti *containment*, pewaris, asosiasi, dan lain-lain.

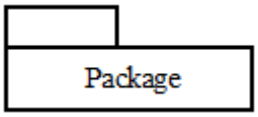
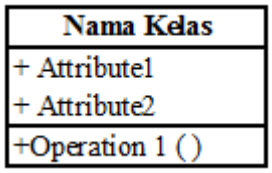
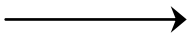
Kelas memiliki tiga area pokok:

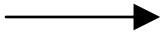
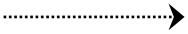
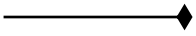
1. Nama
2. Atribut
3. Operasi

Contoh kelas “Manusia” :

1. Atribut: nama, manusia, tanggal lahir
2. Method / operasi: berjalan, makan, minum.

Tabel II.6. *Multiplicity Class Diagram*

Simbol	Deskripsi
Package 	Package merupakan bungkusan dari satu atau lebih kelas
Operasi 	Kelas pada struktur system
Antarmuka/interface  interface	Sama dengan konsep *interface* dalam pemograman berorientasi objek
Asosiasi 1 1..*	Relasi antar kelas dengan maknaumum, asosiasi biasanya juga disertai dengan *multiplicty*
Asosiasi berarah / directed asosiasi 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi	Relasi antar kelas dengan makna *generalisasi*-

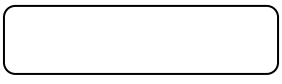
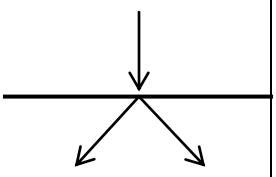
	*spesialisasi* (umum-khusus)
Kebergantungan (*defedency)* 	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi 	Relasi antar kelas dengan makna semua-bagian (*whole-part*)

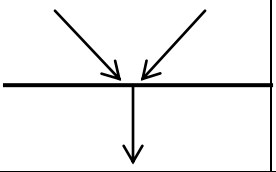
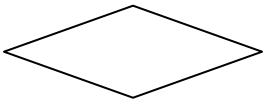
(Sumber: Yuni Sugiarti; 2013: 59)

3. *Actifity Diagram* (Diagram Aktivitas)

Actifity diagram atau diagram aktivitas menggambarkan rangkaian aliran aktivitas baik proses bisnis maupun use case. Actifity diagr*am* dapat juga digunakan untuk memodelkan action yang akan dilakukan saat sebuah operasi dieksekusi, dan memodelkan hasil dari action tersebut.

Tabel II.7. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.

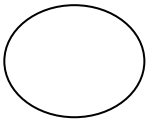
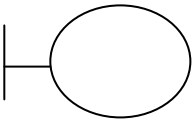
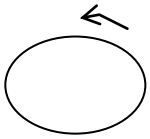
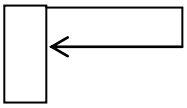
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
New Swimline	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber: Yuni Sugiarti; 2013: 59)

4. *Sequence Diagram* (Diagram Rangkaian)

Secara garis besar menggambarkan bagaimana objek berinteraksi dengan satu sama lain melalui pesan dan pada sekuensi sebuah use case atau operasi. Diagram ini mengilustrasikan bagaimana pesan terkirim dan diterima antara objek dan dalam sekuensi atau timing apa.

Tabel II.8. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class,* berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak.
	*Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive*, menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber: Yuni Sugiarti; 2013: 38)