BAB II

LANDASAN TEORI

BAB II

LANDASAN TEORI

II.1. Penelitian Terdahulu

Berdasarkan penelitian yang dilakukan oleh Stephanie Halim dan Seng Hansun (2015) mengenai Penerapan Metode *Certainty Factor* dalam Sistem Pakar Pendeteksi Resiko Osteoporosis dan Ostearthritis, Stephanie Halim dan Seng Hansun menyimpulkan bahwa implementasi metode *certainty factor* untuk aplikasi sistem pakar mendeteksi resiko penyakit *osteoporosis* dan *osteoarthritis* berhasil diimplementasikan dengan persentase keakuratan 80%.

Berdasarkan penelitian yang dilakukan oleh Sri Yastita, dkk. (2012) mengenai Sistem Pakar Penyakit Kulit Pada Manusia Menggunakan Metode *Certainty Factor* Berbasis *Web*, Sri Yastita, dkk. menyimpulkan bahwa dari inputan 30 pengguna, dokter telah menyesuaikan kecocokan hasil keluaran sistem dengan pengetahuannya, setelah melakukan pencocokan hasil keluaran sistem maka dokter menyimpulkan bahwa 73,15% gejala yang diinputkan dengan hasil keluaran jenis penyakit pengguna sudah sesuai.

Berdasarkan penelitian yang dilakukan oleh Ahyar Supani, dkk. (2014) mengenai Sistem Pakar Diagnosa Gangguan Rahim Dengan Metode *Certainty Factor* Berbasis Web, Ahyar Supani, dkk. menyimpulkan bahwa menurut pemeriksaan dokter bahwa data Ny. B mengalami penyakit rahim endometriosis, kemudian data pemeriksaan gejala yang dialami Ny. B dimasukkan ke sistem

pakar dengan hasil persentase faktor keyakinan menderita 92%, dan menjadi persentase paling tinggi.

Kesimpulan dari perbandingan beberapa jurnal mengenai metode terkait penelitian skripsi ini adalah bahwa beberapa jurnal diatas cukup berbeda dengan penelitian skripsi ini, meski menggunakan metode yang sama yaitu metode *certainty factor* dalam mengukur nilai kepastian namun penelitian ini berfokus pada identifikasi kerusakan treadmill.

II.2. Landasan Teori

II.2.1. Sistem Pakar

Sistem pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yang dipikirkan oleh pakar. Pakar yang dimaksud disini adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam. Sebagai contoh, dokter adalah seorang pakar yang mampu mendiagnosis penyakit yang diderita pasien serta dapat memberikan penatalaksanaan terhadap penyakit tersebut.

Sistem pakar adalah program *artificial intellegence* yang menggabungkan pangkalan pengetahuan *(Knowledge Base)* dengan sistem inferensi. Ini merupakan bagian *software* spesialisasi tingkat tinggi yang berusaha menduplikasi fungsi seorang pakar dalam satu bidang keahlian. Program ini bertindak sebagai seorang konsultan yang cerdas atau penasihat dalam suatu lingkungan keahlian tertentu, sebagai hasil himpunan pengetahuan yang telah dikumpulkan dari beberapa orang

pakar. Dengan demikian seorang awam sekalipun bisa menyadap sistem pakar itu untuk memecahkan berbagai persoalan yang ia hadapi. Sistem pakar sungguh merupakan sesuatu yang baru dan masih segar. Ia sangat inovatif dalam menghimpun dan mengemas pengetahuan. Keampuhannya yang paling utama terletak pada kemampuan dan penggunaan praktisnya bila di satu tempat tidak ada seorang pakar dalam suatu bidang ilmu (Swono Sibagariang : 2015 : 35).

II.2.1.1. Konsep Dasar Sistem Pakar

Dalam konsep dasar sistem pakar ada tiga orang yang terlibat di dalamnya:

1. Pakar adalah orang yang memiliki pengetahuan, khusus, pendapat pengalaman dan metode, serta kemampuan untuk mengaplikasikan keahliannya tersebut guna menyelesaikan masalah.
2. *Knowledge engineer* (Perekayasa Sistem) adalah orang yang membantu pakar dalam menyususun area permasalahan dengan menginterprestasikan dan mengintegrasikan jawaban-jawaban pakar atas pertanyaan yang diajukan, menggambarkan analogi, mengajukan *counter example* dan menerangkan kesulitan-kesulitan konspetual.
3. Pemakai, sistem pakar memiliki beberapa pemakai, yaitu : pemakai bukan pakar, pelajar, pembangun sistem pakar yang ingin meningkatkan dan menambahkan basis pengetahuan dan pakar (Laila Septiana : 2016 : 2).

II.2.1.2. Aktivitas Sistem Pakar

Pengalihan keahlian dari para ahli ke komputer untuk kemudian dialihkan lagi ke orang lain yang bukan ahli, merupakan tujuan utama dari sistem pakar. Proses ini membutuhkan 4 aktivitas yaitu :

1. Tambahan pengetahuan (dari para ahli atau sumber – sumber lainnya).
2. Representasi pengetahuan (ke komputer).
3. Inferensi pengetahuan ke user.
4. Pengalihan pengetahuan ke user. (Ahyar Supani, dkk. : 2014 : 2).

II.2.1.3. Ciri-ciri Sistem Pakar

Ciri-ciri dari sistem pakar adalah sebagai berikut:

1. Terbatas pada *domain* tertentu
2. Dapat memberikan penalaran pada data-data yang bersifat tidak pasti
3. Dapat mengemukakan alasan-alasan yang diberikan dengan cara yang bisa dipahami
4. Dibuat berdasarkan aturan tertentu.
5. Pengembangannya secara bertahap
6. *Output* bersifat saran atau anjuran

(Stephanie Halim, Seng Hansun; 2015: 4)

II.2.1.4. Struktur Sistem Pakar

1. *Knowledge Base Utama*

   Knowledge base adalah representasi pengetahauan dari seorang atau beberapa pakar yang diperlukan untuk memahami, memformulasikan dan memecahkan masalah. Dalam hal ini digunakan untuk memecahkan masalah–masalah yang terjadi pada komputer. *Knowledge base* ini terdiri dari dua eleman dasar, yaitu fakta dan rules.

2. *Interface Engine*

   Inference engine merupakan otak dari sistem pakar yang mengandung mekanisme fungsi berpikir dan pola–pola penalaran sistem yang digunakan oleh seorang pakar. Mekanisme ini menganalisa suatu masalah tertentu dan kemudian mencari solusi atau kesimpulan yang terbaik.

3. *Explanation Subsystem*

   Explanation subsystem memberikan penjelasan saat mana user mengetahui apakah alasan yang diberikan sebuah solusi. Bagian ini yang secara konkrit membedakan sistem pakar dengan sistem aplikasi biasa. Pada bagian ini terdapat informasi tambahan mengapa dan darimana sebuah solusi diperoleh.

4. *User Interface*

   User interface merupakan pengendali *input output* dengan *user* atau sebagai sarana komunikasi antara user dengan sistem pakar tersebut.

5. *Knowledge Base Editor*

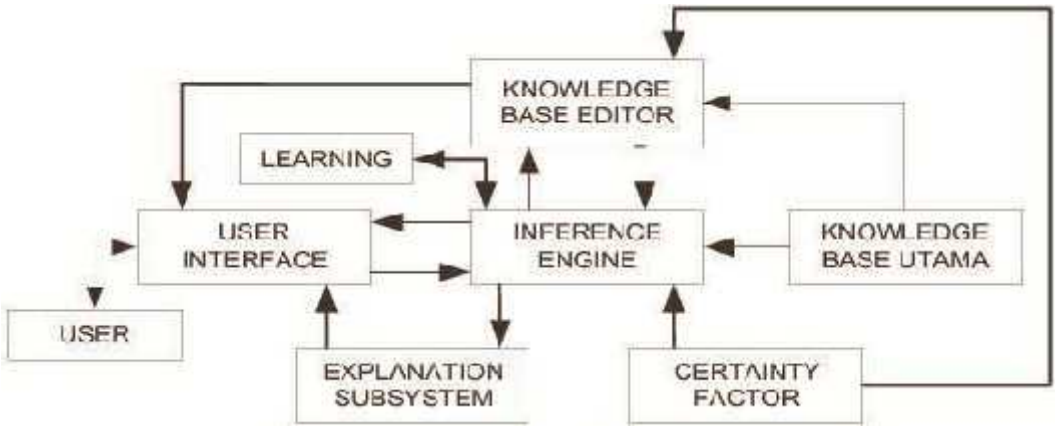
   Knowledge base editor merupakan bagian yang digunakan untuk menambah, menghapus atau memperbaiki basis pengetahuan.

6. *Learning*

   Learning adalah suatu proses belajar dari suatu sistem pakar bila sistem pakar tersebut tidak menemukan solusi.

7. *Certainty Factor*

   Certainty factor merupakan faktor keyakinan dari jawaban user. (Ahyar Supani, dkk. : 2014 : 2)



Gambar II.1. Struktur Sistem Pakar
(Sumber : Ahyar Supani, dkk. : 2014 : 2)

II.2.2. *Certainty Factor*

Faktor kepastian (*Certainty Factor*) diperkenalkan oleh Shortlife Buchanan dalam pembuatan MYCIN. Certainty Factor (CF) merupakan nilai parameter klinis yang diberikan MYCIN untuk menunjukkan besarnya kepercayaan. Certainty Factor menurut Giarrantano dan Riley [2] didefinisikan sebagai berikut :

$$CF(H, E) = MB(H, E) – MD(H, E)$$

Dimana :

CF (H, E) : Certainty Factor dari hipotesis H yang dipengaruhi oleh gejala(evidence) E. Besarnya CF berkisar antara -1 sampai dengan 1. Nilai -1 menunjukkan ketidakpercayaan mutlak, sedangkan 1 menunjukkan kepercayaan mutlak.

MB (H, E) : Ukuran kenaikan kepercayaan (measure of increased belief) terhadap hipotesis H yang dipengaruhi oleh gejala E.

MD (H, E) : Ukuran kenaikan ketidakpercayaan (measure of increased disbelief) terhadap hipotesis H yang dipengaruhi oleh gejala E.

Tabel II.1. Interpretasi Nilai CF

Certainty Term	MD/MB
Tidak Tahu/ Tidak ada	0.2
Mungkin	0.4
Kemungkinan Besar	0.6
Hampir Pasti	0.8
Pasti	1.0

(Sumber : Turban dan Frenzel, 1992)

Berikut contoh perhitungan *certainty factor* pada kasus penyakit *Brucellosis* :

Tabel II.2. Keputusan Gejala Penyakit Brucellosis

Nama Penyakit	Nilai CF Penyakit	Nama Gejala	Nilai CF Gejala
Brucellosis	0.04	Demam Tinggi	0.5
		Badan Lemah	0.2
		Turun berat Badan	0.3
		Mengalami Aborsi	0.6

Nilai Kepastian Pada Gejala Demam Tinggi :

MB(h,E1) = (CFgejala-CFpenyakit)/(1-CFpenyakit)

= (0.5-0.04)/(1-0.04)

= 0.46/0.96

= 0.479

MD(h,E1) = (0.04-0.04)/(0-0.04)

= 0

CF(h,E1) = MB(h,E1)-MD(h,E1)

= 0.479 – 0

= 0.479

Nilai Kepastian Pada Gejala Badan Lemah :

MB(h,E2) = (CFgejala-CFpenyakit)/(1-CFpenyakit)

= (0.2-0.04)/(1-0.04)

= 0.16/0.96

= 0.166

MD(h,E2) = (0.04-0.04)/(0-0.04)

= 0

CF(h,E2) = MB(h,E2)-MD(h,E2)

= 0.479 – 0

= 0.166

Nilai Kepastian Pada Gejala Turun Berat Badan

MB(h,E3) = (CFgejala-CFpenyakit)/(1-CFpenyakit)

= (0.3-0.04)/(1-0.04)

= 0.26/0.96

= 0.270

MD(h,E3) = (0.04-0.04)/(0-0.04)

= 0

CF(h,E3) = MB(h,E3)-MD(h,E3)

= 0.270 – 0

= 0.270

Nilai Kepastian Pada Gejala Aborsi

MB(h,E4) = (CFgejala-CFpenyakit)/(1-CFpenyakit)

= (0.6-0.04)/(1-0.04)

= 0.56/0.96

= 0.583

MD(h,E4) = (0.04-0.04)/(0-0.04)

= 0

CF(h,E4) = MB(h,E4)-MD(h,E4)

= 0.583 – 0

= 0.583

CFkombinasi (CF1, CF2, CF3, CF4) = CF(h,E1)

+ CF(h,E2) + CF(h,E3) + (1-CF(h,E1))

Cfkombinasi = 0.479 + 0.166+0.270+0.583+(1-0.479)

= 0.521 * 1.498

= 0.710

Maka nilai perhitungan rumus 4 diatas menunjukkan bahwa nilai kepastian penyakit sapi Brucellosis dengan tingkat kepastian 0.710. Setelah diperoses akan muncul nilai CF, yang berdasarkan perhitungan nilai MB dan MD dari gejala yang dipilih. Dan hasil proses diagnosis adalah CFpenyakit adalah 0.710 dan kondisi derajat CF adalah Hampir Pasti (Swono Sibagariang : 2015 : 38).

II.2.3 Treadmill

Uji latih jantung dengan menggunakan *treadmill* yang dikenal sebagai uji *treadmill* banyak digunakan sebagai uji penapisan CAD stabil. Uji latih ini bertujuan untuk mengetahui keberadaan abnormalitas pada pembuluh darah koroner dan menjadi referensi untuk pengambilan keputusan perlu tidaknya dilakukan pemeriksaan corangiografi. Uji *treadmill* ini dipertimbangkan positif apabila pada pasien didapatkan angina tipikal terkait aktivitas atau ditemukan abnormalitas EKG yang konsisten dengan gambaran iskemik (depresi segmen ST dengan gambaran horizontal atau *downsloping* >1 mm). Namun, peran uji *treadmill* sebagai uji penapisan CAD stabil memiliki keterbatasan, yaitu terkait dengan ketepatan diagnostiknya. Sementara itu, *Duke Treadmill Score* (DTS) yang diketahui sebagai sistem penilaian prognostik, dapat menghasilkan stratifikasi kelompok risiko berdasarkan prognosis dan prediksi moralitas pada CAD. Metode ini telah diuji untuk penilaian diagnostik dari CAD dan dilaporkan dapat memprediksi CAD secara lebih baik. Oleh karena itu, studi ini bertujuan mengetahui peran *Duke Treadmill Score* (DTS sebagai prediktor penyakit jantung

koroner pada pasien dengan uji *Treadmill* postitif. (Muhammad Ikhsan, Sally Aman Nasution, Ika Prasetya Wijaya, Cleopas Martin Rumende : 2016 : 2)

II.2.4. Visual Studio 2010 (Visual Basic Express 2010)

Visual Basic Express 2010 adalah pengembangan dari Visual Basic 2008 dan Visual Basic 6. Merupakan bahasa pemrograman yang cukup populer dan mudah untuk dipelajari. Microsoft Visual Basic Express menyediakan fasilitas yang memungkinkan kita untuk menyusun sebuah program yang memasang objek-objek grafis dalam sebuah form.

Visual Basic Express 2010 berawal dari bahasa pemrograman BASIC *(Beginners All-Purpose Symbolic Instruction Code)*. Bahasa Basic pada dasarnya adalah bahasa yang mudah dimengerti sehingga pemrograman di dalam bahasa Basic dapat dengan mudah dilakukan meskipun oleh orang yang baru belajar membuat program. Hal ini lebih mudah lagi dengan hadirnya Microsoft Visual Basic, yang dibangun dari ide untuk membuat bahasa yang sederhana dan mudah dalam pembuatan script-nya *(simple scripting language)* untuk *graphic user interface* yang dikembangkan dalam sistem operasi Microsoft Windows. (Nadiatus Sobrina : 2013 : 17-18)

II.2.5. Microsoft SQL Server 2008 R2

Microsoft SQL Server 2008 R2 adalah data yang paling canggih, terpercaya, terukur, dan merupakan platform dirilis sampai saat ini. Membangun kesuksesan SQL Server 2008 yang asli dirilis, SQL Server 2008 R2 telah

membuat dampak pada organisasi di seluruh dunia dengan kemampuan groundbreaking, memberdayakan pengguna akhir melalui intelijen bisnis swalayan (BI), meningkatkan efisiensi dan kolaborasi antara administrator database (DBA) dan aplikasi pengembang, dan penskalaan untuk mengakomodasi beban kerja data yang paling menuntut. (Ross Mistry, Stacia Misner)

II.2.6. *Unified Modelling Language* (UML)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use case* Diagram

   Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-

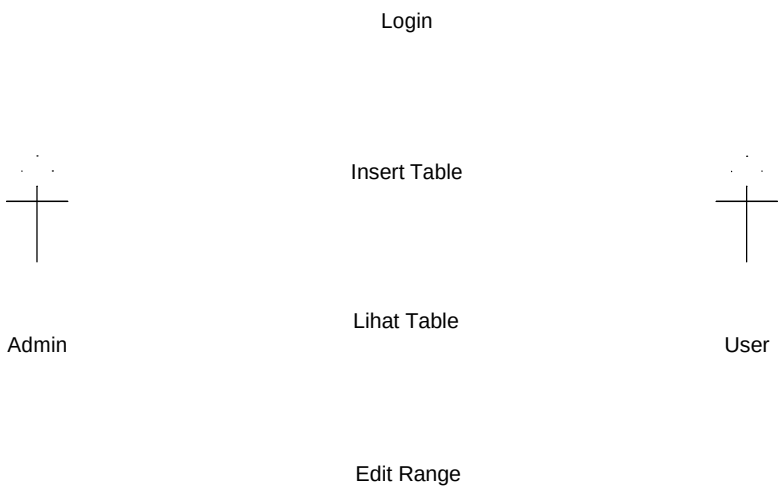
fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel II.3. Simbol *Use Case*

Gambar	Keterangan
	Use case menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Windu Gata, 2013 : 4)

Contoh *use case* diagram :

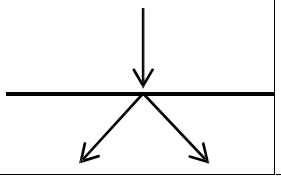


Gambar II.2. Contoh *use case* diagram

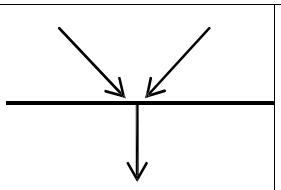
2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu :

Tabel II.4. Simbol *Activity Diagram*

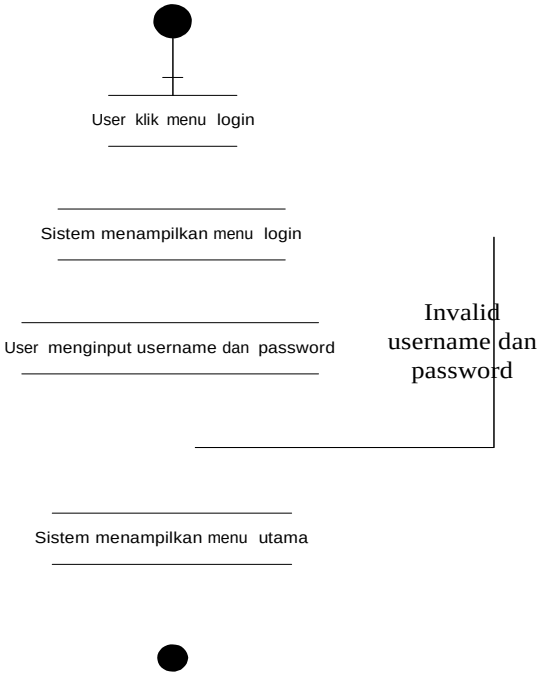
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan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

Tabel II.4. Simbol *Activity Diagram* (Lanjutan)

	*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 : Windu Gata, 2013 : 6)

Contoh *activity diagram* :

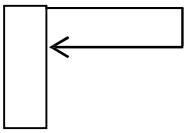


Gambar II.3. Contoh *activity* diagram

3. Diagram Urutan (*Sequence Diagram*)

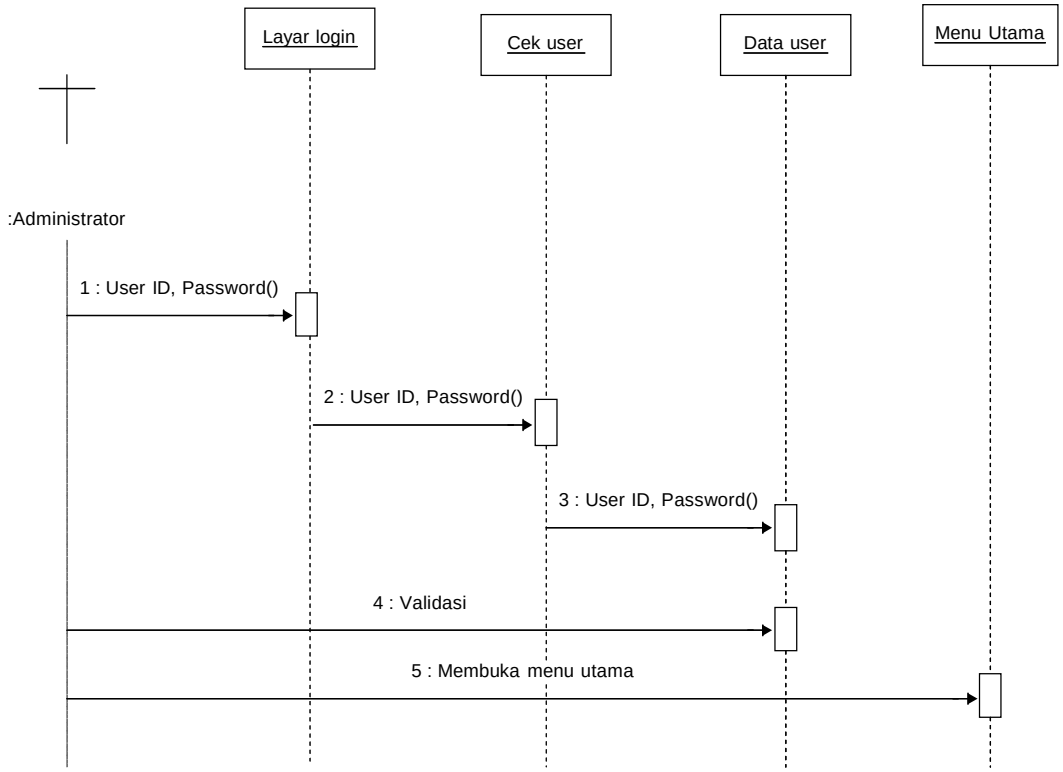
Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.5. 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 : Windu Gata, 2013 : 7)

Berikut contoh diagram *squence* :



Gambar II.4. Contoh diagram *squence*

4. *Class Diagram* (Diagram Kelas)

   Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/ Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan

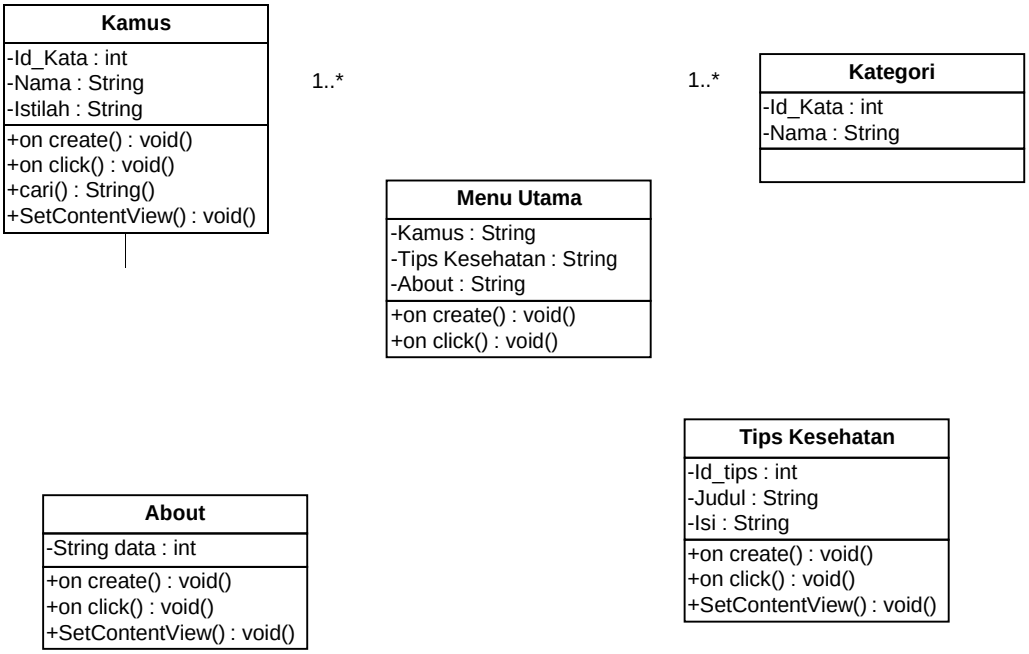
antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.6. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Windu Gata, 2013 : 9)

Berikut contoh *class* diagram :



Gambar II.5. Contoh *class* diagram

II.2.7. *Entity Relationship Diagram* (ERD)

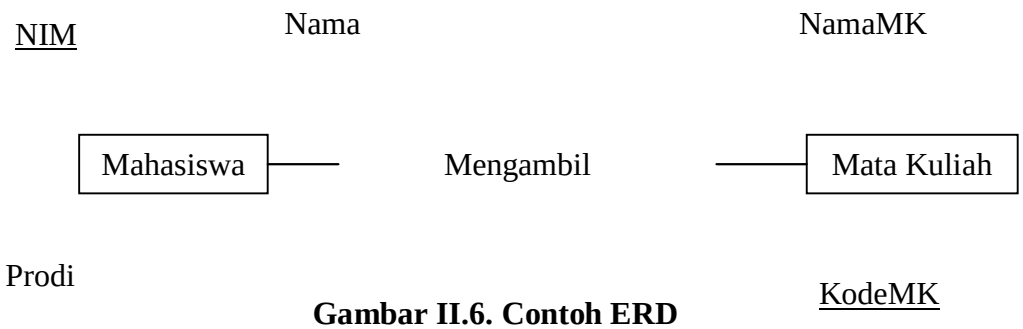
Entity Relationship Diagram atau ERD adalah alat pemodelan data utama dan akan membantu mengorganisasi data dalam suatu proyek ke dalam entitas-entitas dan menentukan hubungan antarentitas. Proses memungkinkan analis menghasilkan struktur basisdata yang baik sehingga data dapat disimpan dan diambil secara efisien (Janner Simarmata, 2010 : 67).

Tabel II.7. Simbol ERD

Notasi	Keterangan
	**Entitas**, adalah suatu objek yang dapat diidentifikasi dalam lingkungan pemakai.
	**Relasi**, menunjukkan adanya hubungan di antara sejumlah entitas yang berbeda.
	**Atribut**, berfungsi mendeskripsikan karakter entitas (atribut yg berfungsi sebagai key diberi garis bawah)
	Garis, sebagai penghubung antara relasi dengan entitas, relasi dan entitas dengan atribut.

(Sumber : Janner Simarmata, 2010 : 67)

Berikut contoh ERD :



Gambar II.6. Contoh ERD

II.2.8. Teknik 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 rasional.

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 normal tahap pertama (*First Normal Form*)**

   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.

   Supaya bisa menggabungkan jumlah barang yang di pasok (qty) secara unik dengan barang (b#) dan pemasok (#p), kita menggunakan kunci utama gabungan yang tersusun dari b# dan p#. Sebuah tabel relasional secara definisi selalu berada dalam bentuk normal pertama. Semua nilai pada kolom-kolom nya adalah atomik. Ini berarti kolom-kolom tidak mempunyai nilai berulang . Tabel II.8. menunjukkan tabel pemasok dalam 1NF.(Janner Simarmata; 2010 : 79).

Tabel II.8. Tabel bentuk normal pertama (1NF)

P#	Status	Kota	B#	Qty
P1	20	Yogyakarta	B1	300
P1	20	Yogyakarta	B2	200
P1	20	Yogyakarta	B3	400
P1	20	Yogyakarta	B4	200
P1	20	Yogyakarta	B5	100
P1	20	Yogyakarta	B6	100
P2	10	Medan	B1	300
P2	10	Medan	B2	400
P3	10	Medan	B2	200
P4	20	Yogyakarta	B2	200
P4	20	Yogyakarta	B2	300
P4	20	Yogyakarta	B5	400

(Sumber : Janner Simarmata, 2010 : 80)

2. Bentuk normal tahap kedua (*Second normal form*)

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

Tabel pemasok berada pada INF, tetapi tidak pada 2NF karena status dan kota tergantung secara fungsional hanya pada kolom p# dari kunci gabungan (p#, b#). Ini dapat digambarkan dengan membuat daftar ketergantungan fungsional;

p# : kota, status

kota : status

(p#, b#) : qty

Pada contoh, kita memindahkan kolom p#, status,dan kota ke tabel baru yang disebut PEMASOK2. kolom p# menjadi kunci utama tabel ini. Tabel II.9. menunjukkan hasilnya.(Janner Simarmata; 2010 ; 81-82).

Tabel II.9. Tabel bentuk normal kedua (2NF)

Pemasok2

P#	Status	Kota
P1	20	Yogyakarta
P2	10	Medan
P3	10	Medan
P4	20	Yogyakarta
P5	30	Bandung

Barang

P#	B#	Qty
P1	B1	300
P1	B2	200
P1	B3	400
P1	B4	200
P1	B5	100
P1	B6	100
P2	B1	300
P2	B2	400
P3	B2	200
P4	B2	200
P4	B2	300
P4	B5	400

(Sumber : Janner Simarmata, 2010 : 82)

3. Bentuk normal tahap ketiga (*Third normal form*)

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 kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya. Untuk mengubah PEMASOK2 menjadi 3NF, kita membuat tabel baru yang disebut KOTA_STATUS dan memindahkan kolom kota dan status ke tabel baru. Status dihapus dari tabel asal, kota tetap dibiarkan karena akan berfungsi sebagai kunci asing bagi KOTA_STATUS, dan tabel asal diberi

nama baru PEMASOK_KOTA. Tabel II.10. menunjukkan hasilnya. (Andi; 2010 ; 82-83).

Tabel II.10. Tabel bentuk normal ketiga (3NF)

PEMASOK_KOTA KOTA_STATUS

P#	Kota
P1	Yogyakarta
P2	Medan
P3	Medan
P4	Yogyakarta
P5	Bandung

Status	Kota
20	Yogyakarta
10	Medan
10	Medan
20	Yogyakarta
30	Bandung

(Sumber : Janner Simarmata, 2010 : 82)

4. *Boyce Code Normal Form* (BCNF)

Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3NF sudah cukup karena sangat jarang entitas yang berada pada 3NF bukan merupakan 4NF dan 5NF.(Janner Simarmata; 2010 ; 84-85).

5. Bentuk Normal Tahap Keempat dan Kelima

Sebuah tabel relasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD).

Misalnya, ada sebuah tabel relasional R dengan kolam A, B dan C, maka R.A R.B (kolom A menentukan kolom B). Adalah benar jika dan hanya jika

himpunan nilai B yang cocok dengan pasangan nilai A dan nilai C pada R hanya tergantung pada nilai A dan tidak tergantung pada nilai C.

Misalnya, pegawai ditugaskan kebanyak proyek dan ia mempunyai banyak keahlian. Jika kita mencatat informasi ini pada satu tabel, ketiga atribut harus digunakan sebagai kunci karena tidak ada satu atribut pun yang dapat secara unik mengidentifikasi sebauh *record*. untuk mengubah sebuah tabel dengan ketergantungan *multivalue* ke dalam 4NF, pindahkan masing-masing pasangan MVD ke tabel baru. Tabel 2.11. menunjukkan hasilnya.

Gambar II.11. Tabel bentuk normal keempat (4NF)

PEGAWAI_PROYEK

Peg#	#pry
1211	P1
1211	P3

PEGAWAI_AHLI

Peg#	Ahli
1211	Analisis
1211	Perancangan
1211	Pemrograman

(Sumber : Janner Simarmata, 2010 : 86)

Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia 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 (*join dependence*).

Sebagai contoh, misalkan kita mempunyai pegawai yang menggunakan keahlian perancangan pada suatu proyek lainnya. (Janner Simarmata, 2010 : 85-86).