BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Berdasarkan penelitian yang dilakukan oleh Jayawardanu dan Hansun (2015) mengenai penelitian yang berjudul Rancang Bangun Sistem Pakar Untuk Deteksi Katarak Dini Menggunakan Algoritma C45, Jayawardanu dan Hansun menyimpulkan Sistem pakar untuk mendeteksi adanya penyakit mata katarak secara dini, sudah berhasil dibangun. Sistem ini berbasis *website,* dengan mengimplementasikan algoritma C4.5 dari metode *learning decision tree.*

Berdasarkan penelitian yang dilakukan oleh Himawan, dkk (2015) mengenai Metode *DecisionTree* untukmendapatkan data yang akurat.pada tahap analisis melihat kebutuhan system yang ada mulai dari sisi kebutuhan system yang mendukung penggunaan aplikasi, sehingga kebutuhan user terhadap aplikasi dari analisa tersebut akan dikembangkan sebuah rancangan yang sesuia ddegan kebutuhan pengguna.

Algoritma C4.5 merupakan salah satu algoritma yang digunakan dalam penyelesaian masalah pada pohon keputusan (decision tree) yang merupakan teknik klasifikasi pada data mining. Secara umum teknik klasifikasi dalam data mining di kelompokkan menjadi beberapa kelompok data (Rachmawati Soewono, Rachmat Gernowo,dkk: 2014)

II.2. Uraian Teoritis

II.2.1. Sistem Pakar

Sistem pakar (*Expert System*) merupakan solusi AI bagi masalah pemrograman pintar (*Intelligent*).Profesor Edward Feigenbaum dari *Stanford University* yang merupakan *pionir* dalam teknologi sistem pakar mendefenisikan sistem pakar sebagai sebuah program komputer pintar (*intelligentcomputer program*) yang memanfaatkan pengetahuan (*knowledge*) dan prosedur inferensi (*inference procedure*) untuk memecahkan masalah yang cukup sulit hingga membutuhkan keahlian khusus dari manusia.

Dengan kata lain, sistem pakar adalah sistem komputer yang ditujukan untuk meeniru semua aspek (*emulates*) kemampuan pengambilan keputusan (*decision making*) seorang pakar. Sistem pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah.

Pakar atau ahli (*expert*) didefenisikan sebagai seseorang yang memiliki pengetahuan atau keahlian khusus yang tidak dimiliki oleh kebanyakan orang.Seorang pakar dapat memecahkan masalah yang tidak mampu dipecahkan kebanyakan orang. Dengan kata lain, dapat memecahkan suatu masalah dengan lebih efisien namun bukan berarti lebih murah. Pengetahuan yang dimuat ke dalam sistem pakar dapat berasal dari seorang pakar atau pun pengetahuan yang berasal dari buku, jurnal, majalah, dan dokumentasi yang dipublikasikan lainnya, serta orang yang memiliki pengetahuan meskipun bukan ahli. Istilah sistem pakar (*expert system*). Sering disinonimkan dengan sistem berbasis pengetahuan (*knowledge-based system*) atau sistem pakar berbasis pengetahuan

(*knowledge based expert system*). (Rika Rosnelly, 2012).

II.2.2. Kelebihan Sistem Pakar

Sistem pakar memiliki beberapa fitur menarik yang merupakan kelebihannya, seperti :

1. Meningkatnya ketersediaan (*increased availability*). Kepakaran atau keahlian menjadi tersedia dalam sistem komputer. Dapat dikatakan bahwa sistem pakar merupakan produksi kepakaran secara masal (*massproduction*).
2. Mengurangi biaya (*reduced cost*). Biaya yang diperlukan untuk menyediakan keahlian per satu orang *user* menjadi berkurang.
3. Mengurangi bahaya (*reduced danger*). Sistem pakar dapat digunakan di lingkungan yang mungkin berbahaya bagi manusia.
4. Permanen (*permanence*). Sistem pakar dan pengetahuan yang terdapat di dalamnya bersifat lebih permanen dibandingkan manusia yang dapat merasa lelah, bosan, dan pengetahuannya hilang saat sang pakar meninggal dunia.
5. Keahlian *multiple* (*multiple expertise*). Pengetahuan dari beberapa pakar dapat dimuat ke dalam sistem dan bekerja secara simultan dan kontinyu menyelesaikan suatu masalah setiap saat. Tingkat keahlian atau pengetahuan yang digabungkan dari beberapa pakar dapat melebihi pengetahuan yang digabungkan dari beberapa pakar dapat melebihi pengetahuan satu orang pakar.
6. Meningkatkan kehandalan (*increased reliability*). Sistem pakar meningkatkan kepercayaan dengan memberikan hasil yang benar sebagai alternatif pendapat dari seorang pakar atau sebagai penengah jika terjadi konflik antara beberapa

pakar. Namun hal tersebut tidak berlaku jika sistem dibuat oleh salah seorang pakar, sehingga akan selalu sama dengan pendapat pakar tersebut kecuali jika sang pakar melakukan kesalahan yang mungkin terjadi pada saat tertekan atau stres.

7. Penjelasan (*explanation*). Sistem pakar dapat menjelaskan detail proses penalaran (*reasoning*) yang dilakukan hingga mencapai suatu kesimpulan. Seorang pakar mungkin saja terlalu lelah, tidak bersedia atau tidak mampu melakukannya setiap waktu. Hal ini akan meningkatkan tingkat kepercayaan bahwa kesimpulan yang dihasilkan adalah benar.
8. Respon yang cepat (*fast response*). Respon yang cepat atau *real time* diperlukan pada beberapa aplikasi. Meskipun bergantung pada *hardware* dan *software* yang digunakan, namun sistem pakar relatif memberikan respon yang lebih cepat dibandingkan seorang pakar.
9. Stabil, tidak emosional, dan memberikan respon yang lengkap setiap saat (*steady, unemotional, and complete response at all times*). Karakteristik ini diperlukan pada situasi *real-time* dan keadaan darurat (*emergency*) ketika seorang pakar mungkin tidak berada pada kondisi puncak disebabkan oleh stress atau kelelahan.
10. Pembimbing pintar (*intelligent tutor*). Sistem pakar dapat berperan sebagai *intelligent tutor* dengan memberikan kesempatan pada *user* untuk menjalankan contoh program dan menjelaskan proses *reasoning* yang dilakukan.

Basis data cerdas (*intelligent database*).Sistem pakar dapat digunakan untuk mengakses basis data secara cerdas.(Rika Rosnelly, 2012).

II.2.3. Konsep Umum Sistem Pakar

Pengetahuan yang dimiliki sistem pakar direpresentasikan dalam beberapa cara. Salah satu metode yang paling umum digunakan adalah tipe *rule* bahkan, pendekatan berbasis pengetahuan (*knowledgebased approach*) untuk membangun sistem pakar telah mematahkan pendekatan awal yang digunakan pada sekitar tahun 1950-an dan 1960-an yang menggunakan teknik penalaran (*reasoning*) yang tidak mengandalkan pengetahuan.(Rika Rosnelly, 2012).

II.2.4. Elemen Manusia Pada Sistem Pakar

Sistem pakar tidak lepas dari elemen manusia yang terkait di dalamnya. Personil yang terkait dengan sistem pakar ada 4 yaitu :

1. Pakar (*expert*)
2. Pembangun pengetahuan (*knowledge engineer*)
3. Pembangunan sistem (*system engineer*)
4. Pemakai (*user*)

Paling tidak terdapat dua komponen orang atau lebih yang berpartisipasi dalam pembangunan dan penggunaan sistem pakar, yakni sedikitnya seorang pembangun pengetahuan dan seorang pakar.(Rika Rosnelly, 2012).

II.2.5. Pakar

Pakar adalah seorang individu yang memiliki pengetahuan khusus, pemahaman, pengalaman, dan metode-metode yang digunakan untuk memecahkan persoalan dalam bidang tertentu. (Rika Rosnelly, 2012)

II.2.5.1. Pembangun/ Pembuat Pengetahuan

Pembuat pengetahuan memiliki tugas utama menerjemahkan dan merepresentasikan pengetahuan yang diperoleh dari pakar, baik berupa pengalaman pakar dalam menyelesaikan masalah maupun sumber terdokumentasi lainnya ke dalam bentuk yang bisa diterima oleh sistem pakar. Dalam hal ini pembangunan pengetahuan (*knowledge engineer*) menginterpretasikan dan merepresentasikan pengetahuan yang diperoleh dalam bentuk jawaban-jawaban atas pertanyaan-pertanyaan yang diajukan pada pakar atau pemahaman, penggambaran analogis, sistematis, konseptual yang diperoleh dari membaca beberapa dokumen cetak seperti *text book,* jurnal, makalah, dan sebagainya. Kurangnya pengalaman *knowledge engineer* merupakan kesulitan utama dalam mengkonstruksi sistem pakar. Untuk mengatasi hal tersebut, perancang sistem pakar menggunakan *tools* komersial. (Seperti pada editor-editor khusus maupun *logic debuggers*) dan usahanya akan dipusatkan pada pembangunan mesin inferensi. (Rika Rosnelly, 2012)

II.2.5.2. Pembangun/ Pembuat Sistem

Pembangunan sistem adalah orang yang bertugas untuk merancang antarmuka pemakai sistem pakar, merancang pengetahuan yang sudah diterjemahkan oleh pembangun pengetahuan ke dalam bentuk yang sesuai dan dapat diterima oleh sistem pakar dan mengimplementasikannya ke dalam mesin inferensi. Selain hal tersebut, pembangun sistem juga bertanggung jawab apabila sistem pakar akan diintegrasikan dengan sistem komputerisasi lain. Alat pembangun (*tool builder*) dapat dipakai untuk menyajikan atau membangun *tool*

yang spesifik. Penjual (*vendor*) dapat memberikan *tool* dan saran, staf pendukung dapat memberikan saran dan bantuan secara teknis dalam proses pembangunan sistem pakar. (Rika Rosnelly, 2012)

II.2.5.3. Pengembangan Sistem Pakar

Pada pengembangan sistem pakar diperlukan beberapa tahapan agar dapatmenghasilkan suatun sistem yang berhubungan dengan tahapan hingga suatu sistemterwujud. Tahapan-tahapan tersebut dapat dilihat pada gambar berikut ini. Secara garis besar pengembangan sistem pakar pada gambar dibawah ini adalah sebagai berikut :

1. Mengidentifikasi masalah dan kebutuhan. Mengkaji situasi dan memutuskan dengan pasti tentang masalah yang akan dikomputerisasi dan apakah dengan sistem pakar bisa lebih membantu atau tidak.
2. Menentukan masalah yang sesuai.
3. Mempertimbangkan alternatif, misalkan menggunakan sistem pakar atau komputer tradisional. (Rika Rosnelly, 2012)

II.2.5.4. Pengguna

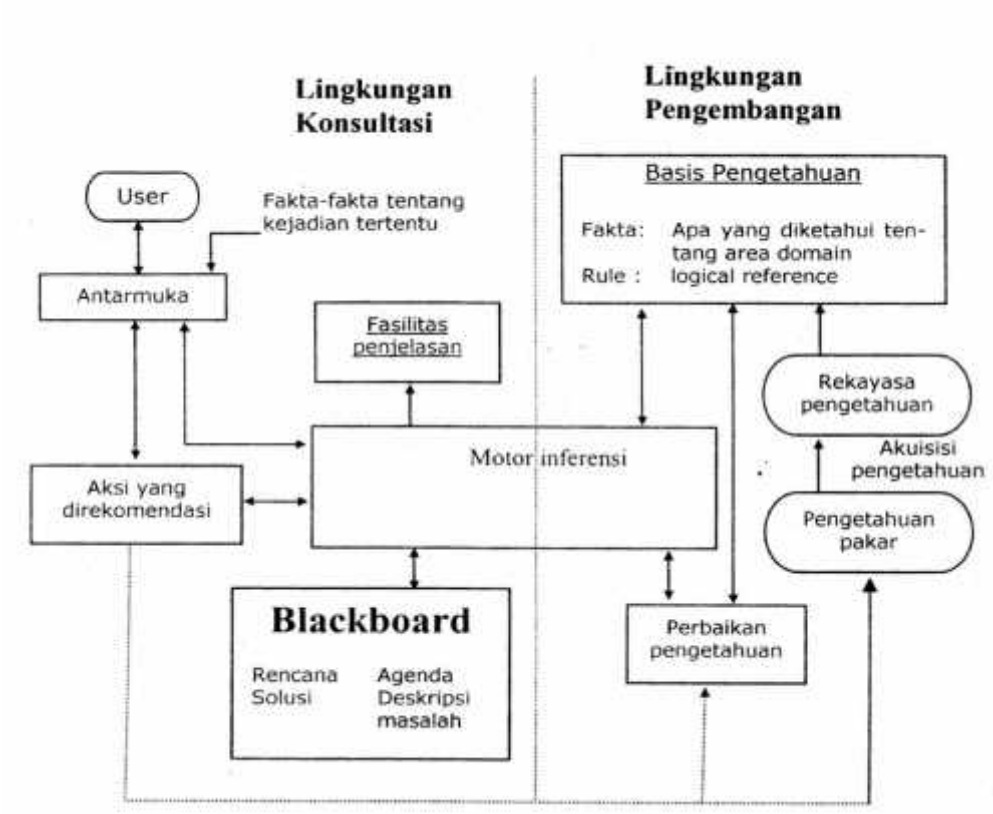
Banyak sistem berbasis komputer mempunyai susunan pengguna tunggal. Hal ini berbedaa jauh dengan sistem pakar yang memungkinkan mempunyai beberapa kelas pengguna. Pengguna mungkin tidak terbiasa dengan komputer dan mungkin pada domain masalah. Bagaimanapun juga, banyak solusi permasalahan menjadi lebih baik dan kemungkinan lebih murah dan keputusan yang cepat bila menggunakan sistem pakar. Pakar dan pembangun sistem harus mengantisipasi

kebutuhan-kebutuhan pengguna dan membuat batasan-batasan ketika mendesain sistem pakar. (Rika Rosnelly, 2012)

II.2.5.5. Struktur Sistem Pakar

Ada dua bagian utama sistem pakar:

1. Lingkungan pengembangan (development environment): digunakan oleh pembuat sistem pakar untuk membagun komponen-komponennya dan memperkenalkan pengetahuan kedalam *knowledge base* (basis pengetahuan)
2. Lingkungan konsultasi (consultation environment): digunakan oleh pengguna berkonsultasi sehingga pengguna mendapatkan pengetahuandan nasihat sistem pakar yang layaknya berkonsultasi dengan seorang pakar.



Struktur Sistem Pakar

Sumber: (T. Sutojo; 2011)

Keterangan:

1. Akuisisi Pengetahuan

   Subsistem ini digunakan untuk memasukkan pengetahuan dari seorang pakar dengan cara merekayasa pengetahuan agar bisa diproses oleh komputer dan manaruhnya ke dalam basis pengetahuan dengan format tertentu (dalam bentuk representasi pengetahuan). Sumber-sumber pengetahuan bisa diperoleh dari pakar, buku, dokumen, multimedia, basis data, laporan riset khusus, dan informasi yang terdapat di web.

2. Basis Pengetahuan (*Knowledge Base*)

   Basis pengetahuan mengandung pengetahuan yang diperlukan untuk memahami, memformulasikan, dan menyelesaikan masalah. Basis pengetahuan terdiri dari dua elemen dasar, yaitu:

   a. Fakta, misalnya situasi, kondisi, atau permasalahan yang ada.
   b. *Rule* (aturan), untuk mengarahkan penggunaan pengetahuan dalam memecahkan masalah.

3. Mesin Inferensi (*Inference Engine*)

   Mesin inferensi adalah sebuah program yang berfungsi untuk memandu proses penalaran terhadap suatu kondisi berdasarkan pada basis pengetahuan yang ada, memanipulasi dan mengarahkan kaidah, model, dan fakta yang disimpan dalam basis pengetahuan untuk mencapai solusi atau kesimpulan. Dalam prosesnya, mesin inferensi menggunakan strategi pengendalian, yaitu strategi yang berfungsi sebagai panduan arah dalam melakukan proses penalaran. Ada tiga teknik pengendalian yang digunakan, yaitu *forward chaining, backward chaining,* dan gabungan dari kedua teknik tersebut.

4. Daerah Kerja (*Blackboard*)

   Untuk merekam hasil sementara yang akan dijadikan sebagai keputusan dan untuk menjelaskan sebuah masalah yang sedang terjadi, sistem pakar membutuhkan *Blackboard* yaitu area pada memori yang berfungsi sebagai basis data. Tiga tipe keputusan yang dapat direkam pada *Blackboard,* yaitu:

a. Rencana : bagaimana menghadapi masalah.

b. Agenda : aksi-aksi potensial yang sedang menunggu untuk dieksekusi.

c. Solusi : calon aksi yang akan dibangkitkan.

5. Antarmuka Pengguna (*User Interface*)

Digunakan sebagai media komunikasi antara pengguna dan sistem pakar, komunikasi ini paling bagus bila disajikan dalam bahasa alami (*natural language*) dan dilengkapi dengan grafik, menu, dan formulir elektronik. Pada bagian ini akan terjadi dialog antara sistem pakar dan pengguna.

6. Subsistem Penjelasan (*Explanation Subsystem/Justifire*)

Berfungsi memberi penjelasan kepada pengguna, bagaimana suatu kesimpulan dapat diambil. Kemampuan seperti ini sangat penting bagi pengguna untuk mengetahui proses pemindahan keahlian pakar maupun dalam pemecahan masalah.

7. Sistem Perbaikan Pengetahuan (*Knowledge Refining System*)

Kemampuan memperbaiki pengetahuan (*knowledge refining system*) dari seorang pakar diperlukan untuk menganalisis pengetahuan, belajar dari kesalahan masa lalu, kemudian memperbaiki pengetahuannya sehingga dapat dipakai pada masa mendatang. Kemampuan evaluasi diri seperti itu diperlukan oleh program agar dapat menganalisa alasan-alasan kesuksesan dan kegagalannya dalam mengambil kesimpulan. Dengan cara ini basis

pengetahuan yang lebih baik dan penalaran yang lebih efektif akan dihasilkan.

8. Pengguna (*User)*

Pada umumnya pengguna sistem pakar bukanlah seorang pakar (*non-expert*) yang membutuhkan solusi, saran, atau pelatihan (*training*) daru berbagai permasalahan yang ada. (T. Sutojo; 2011: 169).

II.3. Metode Algoritma C4.5

Algoritma C4.5 merupakan pengembangan dari algoritma ID3 yang dilakukan [Quinlan].Algoritma ini dapat menyelesaikan masalah secara sistematis dengan membentuk suatu *decision tree* (Ivana Herliana W. Jayawardanu, Seng Hansun ,2015) .

decision tree merupakan metode yang merubah fakta yang sangat besar menjadi sebuah pohon keputusan yang mereprentasikan aturan aturan(Quinlan). Data dalam pohon keputusan biasanya dinyatakan dalam bentuk tabel dengan atribut dan *record*. (Rachmawati Soewono, Rachmat Gernowo,dkk: 2014).

Atribut menyatakan suatu parameter yang disebut sebagai kriteria dalam pembentukan pohon. Salah satu atribut merupakan atribut yang menyatakan data solusi per-item data yang disebut dengan target atribut. Setiap atribut memiliki nilai yang dinamakan dengan *instance*. (Rachmawati Soewono, Rachmat Gernowo,dkk: 2014)

Secara umum algoritma C4.5 untuk membangun langkah-langkah sebagai berikut :

a. Pilih Atribut sebagai akar.
b. Baut cabang untuk masing-masing *record* dari atribut.
c. Membagi kasus dalam cabang.
d. Ulangi proses untuk masing-masing cabang sampai semua kasus pada cabang menghasilkan suatu keputusan yang sesuai.

Dalam memilih sebuah atribut menjadi akar, dilakukan perhitungan nilai dari atribut yang ada. Nilai *gain* yang paling tinggi dijadikan *root* pada pohon keputusan. Untuk menghitung nilai *gain* digunakan rumus:

$$Gain\ (S,A) = Entrophy\ (S) \sum_{i=1}^{n} - pi * \log_2 pi \quad \text{.....(1)}$$

Keterangan :

S : Himpunan kasus

A : Atribut

n : Jumlah Partisi Atribut A

|Si| : Jumlah kasus pada partisi ke-i

Sementara itu, penghitungan nilai *entrophy* dapat dilihat pada persamaan berikut :

$$Entrophy\ (S) = \sum_{i=1}^{n} - pi * \log_2 pi \quad \text{.....(2)}$$

Keterangan :

S : Himpunan kasus

n : Jumlah partisi S

pi : Proporsi dari Si terhadap S. (Ivana Herliana W.Jayawardanu, Seng Hansun ,2015).

II.4. Penyakit Katarak

Katarak adalah Bahasa Yunani, *Katarrhakies* yang berarti air terjun.Katarak merupakan suatu kelainan pada mata dimana lensa mata terlihat keruh.Kekeruhan lensa dapat terjadi pada saat perkembangan serat lensa masih berlangsung atau sesudah serat lensa berhenti perkembangannya. Faktoryang dapat menyebabkan seseorang mengalami katarak adalah : Usia, Cedera Mata, Penyakit Sstemtik, Radiasi Sinar Ultraviolet, Campak (Infeksi Virus Rubella), Riwayat Keturunan, Riwayat Obat Streroid. (Ivana Herliana W. Jayawardanu, Seng Hansun ,2015) .

II.5. *Visual Basic 2012*

Visual studio 2012.net merupakan versi dari visual basic yang diluncurkan oleh Microsoft pada tahun 2012. Microsoft visual studio merupakan sebuah perangkat lunak lengkap (suite) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik ituaplikasi bisnis, aplikasi personal, ataupunkomponen aplikasinya, dalam bentuk aplikasiconsole,aplikasi windows, ataupun aplikasiweb.(Noviardi; 2016)

Visual basic merupakan bahasa pemograman yang berbasis prosedur. Sampai sekarang visual basic telah mengeluarkan versi-versi lain yang lebih tinggi, (Noviardi; 2016)

II.6. *SQL Server 2008*

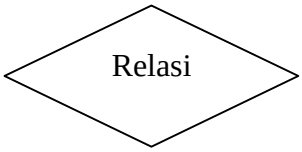
SQL Server 2008 adalah sebuah terobosan baru dari Microsoft dalam bidang *database*. SQL Server adalah DBMS (Database Management *System*) yang dibuat oleh Microsoft untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan Oracle. SQl Server 2008 dibuat pada 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 Prayogi, dkk, 2015).

II.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).

Tabel II.5. Simbol Dari Komponen –Komonen ERD

Entitas	Persegi anjang mewakili kumpulan entitas
Atribut	Elips mewakili atribut

	Belah ketupat mewakii relasi
	Garis menghubugkan atribut degan kumpulan entitas dengan relasi

(Sumber : Janner Simarmata, 2010)

II.8. UML *(Unified Modelling Language)*

Menurut Windu Gata (2013) 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.

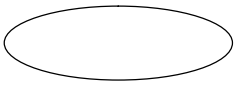
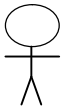
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

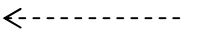
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

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.1. 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.

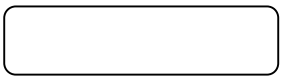
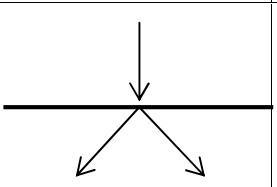
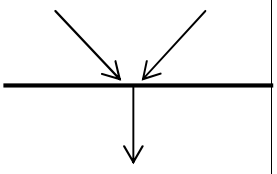
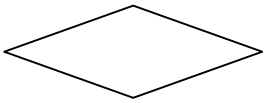
	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Windu Gata, 2013)

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.2. Simbol *Activity Diagram*

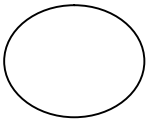
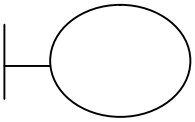
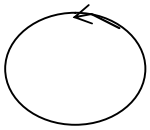
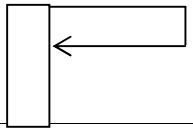
Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

(Sumber : Windu Gata, 2013)

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.3. Simbol *Sequence Diagram*

Gambar	Keterangan
	*EntityClass,* 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)

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.4. *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)

II.9. Teknik Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basisdata 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 tekah ditemukan.

II.9.1.Bentuk-bentuk Normalisasi

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 . Gambar 2.3 menunjukkan tabel pemasok dalam 1NF.(Janner Simarmata; 2010).

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

Gambar II.1. Tabel Bentuk Normal Pertama (1NF)

(Sumber :Janner Simarmata, 2010)

2. Bentuk normal tahap kedua (*Secondnormal 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. Gambar 2.4 menunjukkan hasilnya.(Janner Simarmata; 2010)

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

Gambar II.2. Tabel bentuk normal kedua (2NF)
(Sumber :Janner Simarmata, 2010 : 82)

3. Bentuk normal tahap ketiga (*Thirdnormal 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. Gambar 2.5 menunjukkan hasilnya.(Andi; 2010).

PEMASOK_KOTA

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

KOTA_STATUS

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

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

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, makaR.A R.B (kolom A menentukkan 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. Gambar 2.6 menunjukkan hasilnya.

PEGAWAI_PROYEK

Peg#	#pry
1211	P1
1211	P3

PEGAWAI_AHLI

Peg#	Ahli
1211	Analisis
1211	Perancangan
1211	Pemrograman

Gambar II.4. Tabel bentuk normal keempat (4NF)
(Sumber :Janner Simarmata, 2010)

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*).

Contoh, misalkan kita mempunyai pegawai yang menggunakan keahlian perancangan pada suatu proyek lainnya.(Janner Simarmata, 2010).