

BAB II

TINJAUAN PUSTAKA

II.1. Kecerdasan Buatan

II.1.1. Pengertian Kecerdasan Buatan

Kecerdasan buatan berasal dari bahasa Inggris “*Artificial Intelligence*” atau disingkat AI, yaitu *intelligence* adalah kata sifat yang berarti cerdas, sedangkan *artificial* artinya buatan. Kecerdasan buatan yang dimaksud di sini menunjuk pada mesin yang mampu berfikir, menimbang tindakan yang akan diambil, dan mampu mengambil keputusan seperti yang dilakukan oleh manusia (T.Sutojo, dkk ; 2011 :1).

Berikut ini adalah beberapa definisi kecerdasan buatan yang telah didefinisikan oleh beberapa ahli :

1. Hebert Alexander Simon (June 15, 1916 February 9, 2001) :

“Kecerdasan buatan (*Artificial Intelligence*) merupakan kawasan penelitian, aplikasi, dan instruksi yang terkait dengan pemograman komputer untuk melakukan sesuatu hal yang dalam pandangan adalah cerdas”.

2. Rich and Knight (1991) :

“Kecerdasaan buatan (AI) merupakan sebuah studi tentang bagaimana membuat komputer melakukan hal-hal yang pada saat ini dapat dilakukan lebih baik oleh manusia”.

3. Encyclopedia Britannica

“Kecerdasan buatan (AI) merupakan cabang dari ilmu komputer yang dalam merepresentasi pengetahuan lebih banyak menggunakan simbol-simbol daripada bilangan dan memproses informasi berdasarkan metode heuristic atau dengan berdasarkan sejumlah aturan”.

4. Menurut Winston dan Prendergast (1984), tujuan dari kecerdasan buatan adalah :
 - a. Membuat mesin menjadi lebih pintar (tujuan utama).
 - b. Memahami apa itu kecerdasan (tujuan ilmiah).
 - c. Membuat mesin lebih bermanfaat (tujuan *entrepreneurial*).

II.2. Sistem

II.2.1. Pengertian Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan/berinteraksi dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu (Asbon Hendra;2012:157).

Berikut adalah beberapa pengertian sistem yang telah didefinisikan oleh beberapa ahli :

1. Menurut Jerry FithGerald, sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu.
2. Menurut L. Ackof, sistem adalah setiap kesatuan secara konseptual atau fisik yang terdiri dari bagian-bagian dalam keadaan saling tergantung satu sama lainnya.

II.3. Pakar

II.3.1. Pengertian Pakar

Pakar adalah seorang individu yang memiliki pengetahuan khusus, pemahaman, pengalaman, dan metode-metode yang digunakan untuk memecahkan persoalan dalam bidang tertentu (Rika Rosnelly; 2012:10).

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.

II.4. Sistem Pakar (*Expert System*)

II.4.1. Pengertian Sistem Pakar

Sistem Pakar (*Expert System*) adalah sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pernyataan dan memecahkan suatu masalah (T.Sutojo;2010:13).

Sistem pakar adalah sebuah sistem yang menggunakan pengetahuan manusia dimana pengetahuan tersebut dimasukkan ke dalam sebuah komputer dan kemudian digunakan untuk menyelesaikan masalah-masalah yang biasanya membutuhkan kepakaran atas keahlian manusia (T.Sutojo;2010:161).

Dengan kata lain, sistem pakar adalah sistem komputer yang ditunjukkan untuk meniru semua aspek (*emulates*) kemampuan pengambilan keputusan

(*decision making*) seorang pakar. Sistem pakar memanfaatkan secara maksimal pengetahuan khusus selayaknya seorang pakar untuk memecahkan masalah (Rika Rosnelly;2012:2).

II.4.2. Manfaat Sistem Pakar

Menurut (T.sutojo;2011:160) Sistem pakar menjadi sangat populer karena sangat banyak kemampuan dan manfaat yang diberikan, diantaranya:

1. Meningkatkan produktivitas, karena Sistem Pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberikan 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. Integritas Sistem Pakar dengan sistem komputer lain membuat sistem lebih efektif dan mencakup lebih banyak sistem.
9. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti. Berbeda dengan sistem komputer konvensional, Sistem Pakar dapat bekerja dengan informasi yang tidak lengkap. Pengguna dapat merespons dengan: “tidak tahu” atau “tidak yakin” pada satu atau lebih pertanyaan selama konsultasi dan sistem pakar tetap akan memberikan jawabannya.

10. Bisa digunakan sebagai media pelengkap dalam pelatihan. Pengguna pemula bekerja dengan Sistem Pakar akan menjadi lebih berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.
11. Meningkatkan kemampuan untuk menyelesaikan masalah karena Sistem Pakar mengambil sumber pengetahuan dari banyak pakar.

II.4.3. Kekurangan Sistem Pakar

Selain manfaat, sistem pakar juga memiliki beberapa kelemahan, diantaranya:

1. Memerlukan biaya yang sangat mahal untuk membuat dan memelihatanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
3. Sistem Pakar tidak selamanya 100% bernilai benar.

II.4.4. Ciri-ciri Sistem Pakar

Sistem Pakar memiliki beberapa ciri-ciri, diantaranya sebagai berikut:

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran untuk data-data yang tidak lengkap atau tidak pasti.
3. Dapat menjelaskan alasan-alasan dengan cara yang dapat dipahami.
4. Bekerja berdasarkan kaidah/*rule* tertentu.
5. Mudah dimodifikasi
6. Basis pengetahuan dan mekanisme inferensi terpisah.
7. Keluarannya bersifat anjuran.
8. Sistem dapat mengaktifkan kaidah secara searah yang sesuai, dituntun oleh dialog dengan pengguna.

II.4.5. Konsep Dasar Sistem Pakar

Konsep dasar sistem pakar meliputi enam hal berikut ini:

II.4.5.1. Kepakaran (*Expertise*)

Kepakaran merupakan suatu pengetahuan yang diperoleh dari pelatihan, membaca dan pengalaman. Kepakaran inilah yang memungkinkan para ahli dapat mengambil keputusan lebih cepat dan lebih baik daripada seseorang yang bukan pakar. Kepakaran itu sendiri meliputi pengetahuan tentang.

1. Fakta-fakta tentang bidang permasalahan tertentu.
2. Teori-teori tentang bidang permasalahan tertentu.
3. Aturan-aturan dan prosedur-prosedur menurut bidang permasalahan umumnya.
4. Atuan *heuristic* yang harus dikerjakan dalam suatu situasi tertentu.
5. Strategi global untuk memecahkan permasalahan
6. Pengetahuan tentang pengetahuan (*meta knowledge*).

II.4.5.2. Pakar (*Expert*)

Pakar adalah seseorang yang mempunyai pengetahuan, pengalaman dan metode khusus serta mampu menerapkannya untuk memecahkan masalah atau memberi nasehat. Seorang pakar harus mampu menjelaskan dan mempelajari hal-hal baru yang berkaitan dengan topik permasalahan, jika perlu harus mampu menyusun kembali pengetahuan-pengetahuan yang didapatkan dan dapat memecahkan aturan-aturan serta menentukan relevansi kepakarannya. Jadi seorang pakar harus mampu melaksanakan kegiatan-kegiatan berikut ini:

1. Mengenali dan memformulasikan permasalahan.
2. Memecahkan permasalahan secara cepat dan tepat.
3. Menerangkan pemecahannya.

4. Belajar dari pengalaman.
5. Merestrukturasikan pengetahuan.
6. Memecahkan aturan-aturan.
7. Menentukan relevansi.

II.4.5.3. Pemindahan Kepakaran (*Transferring Expertisi*)

Tujuan dari Sistem Pakar adalah memindahkan kepakaran dari seorang pakar ke dalam komputer, kemudian kepada orang lain yang bukan pakar. Proses ini melibatkan empat kegiatan, yaitu:

1. Akuisisi pengetahuan (dari pakar atau sumber lain).
2. Representase pengetahuan (pada komputer).
3. Inferensi pengetahuan.
4. Pemindahan pengetahuan ke pengguna.

II.4.5.4. Inferensi (*Inferencing*)

Inferensi adalah sebuah prosedur (program) yang mempunyai kemampuan dalam melakukan penalaran. Inferensi ditampilkan pada suatu komponen yang disebut mesin inferensi yang mencakup prosedur-prosedur mengenai pemecahan masalah. Semua pengetahuan yang dimiliki oleh seorang pakar disimpan pada basis pengetahuan oleh sistem pakar. Tugas mesin adalah mengambil kesimpulan berdasarkan basis pengetahuan.

II.4.5.5. Aturan-aturan (*Rules*)

Kebanyakan software pakar komersil adalah sistem yang berbasis *rule* (*rule-based system*), yaitu pengetahuan disimpan terutama dalam bentuk *rule*, sebagai prosedur pemecahan masalah.

II.4.5.6. Kemampuan Menjelaskan (*Explanation Capability*)

Fasilitas lain dari sistem pakar adalah kemampuannya untuk menjelaskan saran atau rekomendasi yang diberikannya. Penjelasan dilakukan dalam subsistem yang disebut subsistem penjelasan (*explanation*). Bagian dari sistem ini memungkinkan sistem untuk memeriksa penalaran yang dibuatnya sendiri dan menjelaskan operasi-operasinya.

Karakteristik dan kemampuan yang dimiliki oleh sistem pakar berbeda dengan sistem konvensional. Perbedaan ini ditunjukkan pada Tabel II.1

Tabel II.1 Perbandingan antara Sistem Konvensional dengan Sistem Pakar

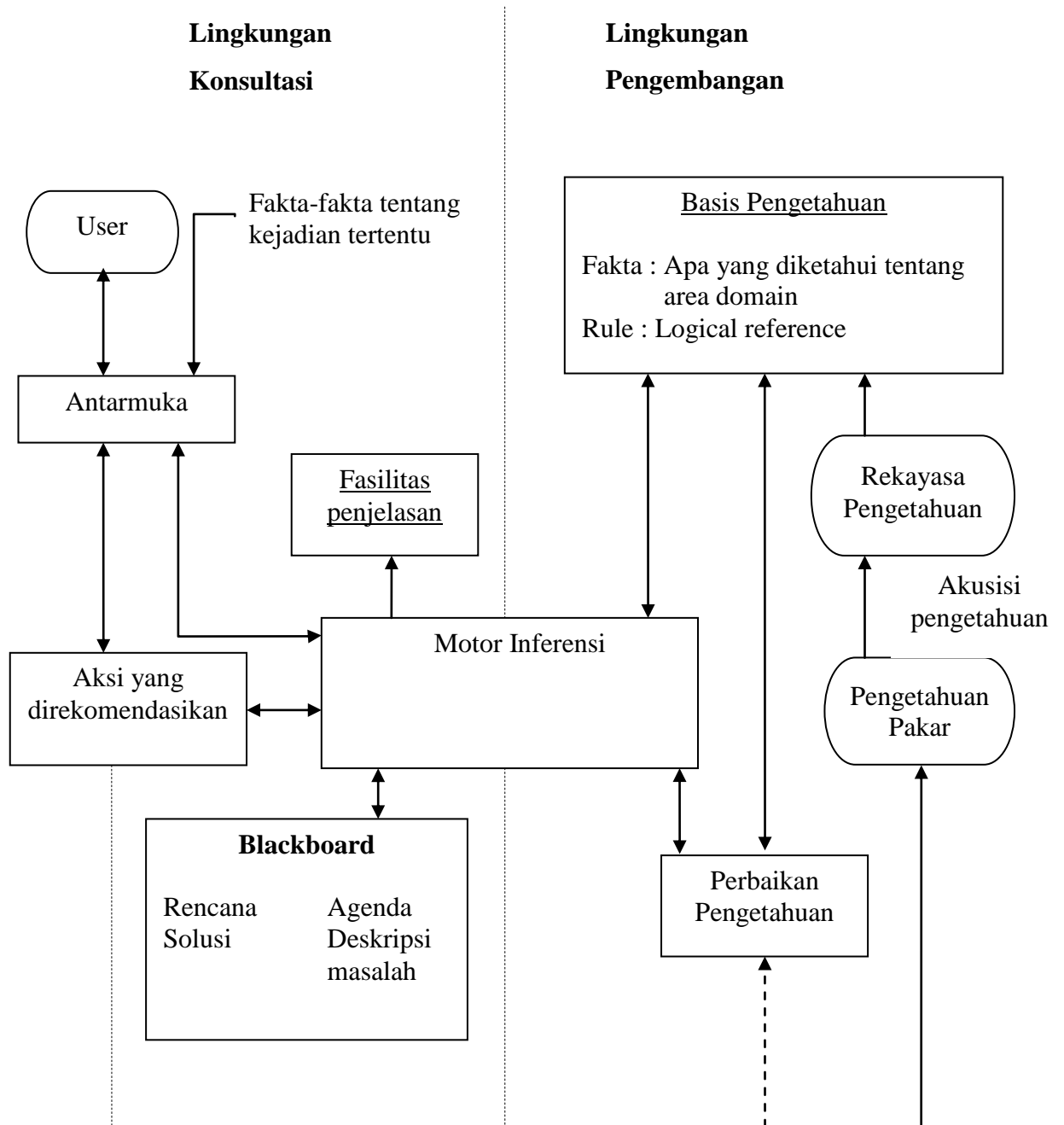
| Sistem Konvensional | Sistem Pakar |
|--|---|
| Informasi dan pemrosesannya biasanya digabungkan dalam satu program. | Basis pengetahuan dipisahkan secara jelas dengan mekanisme inferensi. |
| Program tidak membuat kesalahan (yang membuat kesalahan: Pemrogram atau Pengguna). | Program dapat berbuat kesalahan. |
| Biasanya tidak menjelaskan mengapa data masukan diperlakukan atau bagaimana output dihasilkan. | Penjelasan merupakan bagian terpenting dari semua sistem pakar. |
| Perubahan program sangat menyulitkan. | Perubahan dalam aturan-aturan mudah untuk dilakukan. |
| Sistem hanya bisa beroperasi setelah lengkap atau selesai. | Sistem dapat beroperasi hanya dengan aturan-aturan yang sedikit (sebagai prototipe awal). |
| Eksekusi dilakukan berdasarkan langkah demi langkah (<i>algorithmic</i>). | Eksekusi dilakukan dengan menggunakan <i>heuristic</i> dan logika pada seluruh basis |

| | |
|--|--|
| | pengetahuan. |
| Perlu informasi lengkap agar bisa beroperasi. | Dapat beroperasi dengan informasi yang tidak lengkap atau mengandung ketidakpastian. |
| Menaipulasi efektif dari basis data yang besar. | Manipulasi efektif dari basis pengetahuan yang besar. |
| Menggunkan data. | Menggunakan pengetahuan. |
| Tujuan utama efisiensi. | Tujuan utama efektivitas. |
| Mudah berurusan dengan data kuantitatif. | Mudah berurusan dengan data kualitatif. |
| Menangkap, menambah dan mendistribusikan akses ke data numerik atau informasi. | Menangkap, menambah dan mendistribusikan akses ke pertimbangan dan pengetahuan. |

Sumber: (T. Sutojo;2011:165)

II.4.5.7. Struktur Sistem Pakar

Ada dua bagian penting dari sistem pakar, yaitu lingkungan pengembangan (development environment) dan lingkungan konsultasi (consultation environment). Lingkungan pengembangan digunakan oleh pembuat sistem pakar untuk membangun komponen – komponennya dan memperkenalkan pengetahuan kedalam knowledge base (basis pengetahuan). Lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan nasihat dari sistem pakar layaknya berkonsultasi dengan seorang pakar. Gambar II.1 menunjukkan komponen- komponen yang penting dalam sebuah sistem pakar.



Gambar II.1. Komponen-komponen yang penting dalam sebuah sistem pakar

Sumber : (T.Sutojo,dkk ;2011:167)

Keterangan:

1. Akuisi Pengetahuan

Subsistem ini digunakan untuk memasukkan pengetahuan dari seorang pakar dengan cara merekayasa pengetahuan agar bisa diproses oleh komputer dan menaruhnya kedalam 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 diweb.

2. Basis pengetahuan (*Knowledge Base*)

Basis pengetahuan mengandung pengetahuan yang diperlukan untuk memahami merformulasikan, 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.

- *Forward Chaining*

Forward chaining adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan bagian IF dari *rules* IF-THEN. Bila ada fakta yang cocok dengan bagian IF, maka rule tersebut dieksekusi. bila sebuah rule dieksekusi, maka sebuah fakta baru (bagian THEN) ditambahkan ke dalam database. setiap kali pencocokan, di mulai dari rule teratas. Setiap rule hanya boleh dieksekusi sekali saja. proses pencocokan berhenti bila tidak ada lagi rule yang bisa dieksekusi.

- *Backward Chaining*

Backward chaining adalah metode inferensi yang bekerja mundur ke arah kondisi awal. proses diawali dari goal (yang berada dibagian THEN dari rule IF-THEN), kemudian pencarian mulai dijalankan untuk mencocokkan apakah fakta-fakta yang ada cocok dengan premis-premis di bagian IF. Jika cocok, rule dieksekusi, kemudian hipotesis di bagian THEN ditempatkan di basis data sebagai fakta baru.

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 formulis elektronik. Pada bagian ini akan terjadi dialog antara sistem pakar dan pengguna.

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

Berfungsi memeberi 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 Perbaiki Pengetahuan (*Knowledge Refining System*)

Kemampuan memperbaiki pengetahuan (*Knowledge Refining System*) dari seorang pakar diperlukan untuk menganalisa 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*) dari berbagai permasalahan yang ada.

II.4.5.8. Karakteristik Sistem Pakar

Sistem Pakar pada umumnya dirancang untuk memenuhi beberapa karakteristik umum berikut ini :

1. Kinerja sangat baik (*high performance*). Sistem harus mampu memberikan respon berupa saran (*advice*) dengan tingkat kualitas yang sama dengan seorang pakar atau melebihinya.
2. Waktu respon yang baik (*adequate respon time*). Sistem juga harus mampu bekerja dalam waktu yang sama baiknya (*reasonable*) atau lebih cepat dibandingkan dengan seorang pakar dalam menghasilkan keputusan.
3. Dapat diandalkan (*good reliability*). Sistem harus dapat diandalkan dan tidak mudah rusak/ crash.
4. Dapat dipahami (*understandable*). Sistem harus mampu menjelaskan langkah-langkah penalaran yang dilakukannya seperti seorang pakar.
5. Fleksibel (*fleksibility*). Sistem harus menyediakan mekanisme untuk menambah, mengubah, dan menghapus pengetahuan (Rika Rosnelly; 2012 :20)

II.5. Penyakit Kista

II.5.1. Pengertian Penyakit Kista

Kista adalah suatu organ yang membesar dan didalamnya berisi cairan, seperti sebuah balon yang berisi air. Pada wanita, organ yang paling sering menjadi kista adalah indung telur. Tidak ada ketentuan apakah indung telur kiri

atau kanan yang sering menjadi kista. Pada kebanyakan kasus kista justru tidak memerlukan operasi. (Andhita Caya Anolis:2011;113)

Kista ovarium terbentuk oleh bermacam sebab. Penyebab inilah yang nantinya akan menentukan tipe sari kista. Diantaranya beberapa tipe kista ovarium, tipe folikuler merupakan tipe kista yang paling banyak ditemukan. Kista jenis ini terbentuk oleh karena pertumbuhan folikel ovarium yang tidak terkontrol. Bisa juga sejak kecil atau lahir sudah berbakat ke arah penyakit atau yang lain, kista ini disebut kista dermoid dan sudah dibawa sejak dalam kandungan ibunya. (Andhita Caya Anolis:2011;114)

Ada yang kemudian tumbuh belakangan seperti kista endometriosis yang merupakan gangguan hormonal dan gangguan kekebalan tubuh. Ada juga yang berisi nanah bisa juga karena perubahan sel tubuh, isinya seperti cairan bening disebut kista mesinosum atau serosum. Jadi penyebabnya banyak sekali.

Kista ovarium ada yang bersifat neoplastik (bisa jinak atau ganas), dan nonneoplastik (kista fisiologis, kista akibat peradangan). Walaupun umumnya tidak ganas, kadang kista ini mengganggu siklus menstruasi dan menimbulkan rasa nyeri perut dibagian bawah.

Kista yang bersifat fisiologis sering terjadi dan ini masih dalam batas normal. Sesuai siklus menstruasi. Di ovarium timbul folikel dan folikelnya berkembang membentuk kista ukuran kurang dari 5 cm dan dalam waktu 2-3 bulan akan hilang. (Andhita Caya Anolis:2011;114)

II.5.2. Jenis – Jenis Penyakit Kista

Ada 4 macam kista indung telur. Kista fungsional, dermoid, cokelat (endometriosis) dan kista kelenjar (cystadenoma). Sampai saat ini masih belum diketahui bagaimana terjadinya kista. Biasanya tumbuh sangat pelan dan sering terjadi keganasan pada umur 45 tahun. (Andhita Caya Anolis:2011;115)

Sebagian besar kista tanpa gejala dan diketahui secara kebetulan pada waktu periksa dokter. Menurut pengalaman, diketahuinya menderita kista indung telur biasanya sewaktu periksa check up atau sewaktu periksa karena sebab lain.

Selain itu juga dapat timbul gejala yang khas untuk kista indung telur dan sangat terkait dengan jenis kista indung telur. dan sangat terkait dengan jenis kista indung telur.

1. Kista fungsional. Sering tanpa gejala. Timbul gejala rasa sakit apabila disertai komplikasi seperti terpuntir atau pecah, tetapi komplikasi ini sangat jarang. Kista fungsional ini paling sering terjadi dan sangat jarang pada dua indung telur. Ia bisa mengecil sendiri 1-3 bulan.
2. Kista dermoid. Kista ini terjadi karena jaringan dalam telur yang tidak dibuahi. Kemudian tumbuh menjadi beberapa jaringan seperti rambut, tulang dan lemak. Kista ini dapat terjadi pada dua indung telur dan biasanya tanpa gejala. Timbul gejala rasa sakit apabila kista terpuntir atau pecah.
3. Kista cokelat (endometrioma). Terjadi karena lapisan di dalam rahim (yang biasanya terkelupas sewaktu haid dan terlihat keluar dari kemaluan seperti darah), tidak terletak dalam rahim tetapi melekat pada dinding luar indung telur. Akibat peristiwa ini setiap kali haid, lapisan tersebut menghasilkan darah haid, yang akan terus menerus tertimbun dan menjadi kista. Kista ini

bisa pada satu atau dua indung telur. Timbul gejala utama yaitu rasa sakit terutama sewaktu haid atau sexual intercourse.

4. Kista denoma. Berasal dari pembungkus indung telur yang tumbuh menjadi kista. Kista jenis ini juga dapat menyerang indung telur kanan dan kiri. Gejala yang timbul biasanya akibat penekanan pada sekitar kandung kencing sehingga dapat menyebabkan semacam “beser”.

II.6. Metode *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* (Rismawati;2013:72-75).

II.6.1. Pengertian Metode *Dempster Shafer*

Metode *Dempster-Shafer* pertama kali diperkenalkan oleh *Dempster*, yang melakukan percobaan model ketidakpastian dengan *range probabilities* dari pada sebagai probabilitas.tunggal. Kemudian pada tahun 1976 *Shafer* mempublikasikan teori *Dempster* itu pada sebuah buku yang berjudul *Mathematical Theory Of Evident* [8]. *Dempster-Shafer Theory Of Evidence*, menunjukkan suatu cara untuk memberikan bobot keyakinan sesuai fakta yang

dikumpulkan. Pada teori ini dapat membedakan ketidakpastian dan ketidaktahuan. Teori *Dempster-Shafer* adalah representasi, kombinasi dan propogasi ketidakpastian, dimana teori ini memiliki beberapa karakteristik yang secara institutif sesuai dengan cara berfikir seorang pakar, namun dasar matematika yang kuat (Elyza Gustru Wahyuni,Widodo Prijodiprojo;2013:133-144).

Dempster-Shafer adalah suatu teori matematika untuk pembuktian berdasarkan *belief functions and plausible reasoning*(fungsi kepercayaan dan pemikiran yang masuk akal), yang digunakan untuk mengkombinasikan potongan informasi yang terpisah (bukti) untuk mengkalkulasi kemungkinan dari suatu peristiwa. Teori ini dikembangkan oleh Arthur P. Dempster dan Glenn Shafer. Secara umum teori *Dempster-Shafer* ditulis dalam suatu interval. (Rismawati;2013:72-75).

Belief (Bel) 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. Dimana nilai bel yaitu (0-0.9) (Rismawati;2013:72-75).

Plausibility (Pl) dinotasikan sebagai :

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

Plausibility juga bernilai 0 sampai 1. Jika yakin akan -s, maka dapat dikatakan bahwa $Bel(-s)=1$, dan $Pl(-s)=0$. Pada teori *Dempster-Shafer* dikenal adanya *frame of discrement* yang dinotasikan dengan θ . Frame ini merupakan semesta pembicaraan dari sekumpulan hipotesis. Tujuannya adalah mengaitkan 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 2^n . Jumlah semua m dalam subset θ sama dengan 1. Apabila tidak ada informasi apapun untuk memilih hipotesis, maka nilai : $m\{\theta\} = 1,0$.

Apabila diketahui X adalah subset dari θ , dengan m_1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari θ dengan m_2 sebagai fungsi densitasnya, maka dapat dibentuk fungsi kombinasi m_1 dan m_2 sebagai m_3 , yaitu:

$$m_3(Z) = \frac{\sum_{x \cap y = z} m_1(X).m_2(Y)}{1 - \sum_{x \cap y = \emptyset} m_1(X).m_2(Y)}$$

Dimana :

$m_3(Z) =$ *mass function* dari *evidence* (Z)

$m_1(X) =$ *mass function* dari *evidence* (X), yang diperoleh dari nilai keyakinan suatu *evidence* dikalikan dengan nilai *disbelief* dari *evidence* tersebut.

$m_2(Y) =$ *mass function* dari *evidence* (Y), yang diperoleh dari nilai keyakinan suatu *evidence* dikalikan dengan nilai *disbelief* dari *evidence* tersebut.

$\sum m_1(X).m_2(Y) =$ merupakan nilai kekuatan dari *evidence* Z yang diperoleh dari kombinasi nilai keyakinan sekumpulan *evidence*).

II.7. Visual Basic 2010

II.7.1. Pengertian Visual Basic 2010

Visual Basic merupakan salah satu bahasa pemrograman yang handal dalam lingkungan Windows. Visual Basic telah merajai pasar pembuatan *software* / perangkat lunak sampai beberapa dekade tanpa ada yang menyaingi. Visual Basic 2010 merupakan teknologi terbaru yang masuk ke dalam Visual Studio bersama dengan C#, C++, dan yang lainnya (Wahana Komputer ;2010).

Visual Basic adalah salah satu development tools untuk membangun aplikasi dalam lingkungan Windows. Visual Basic menggunakan pendekatan visual untuk merancang user interface dalam bentuk form, sedangkan untuk codingnya menggunakan dialek bahasa BASIC yang cenderung mudah dipelajari. Pada pemrograman visual, pengembangan aplikasi dimulai dengan pembentukan user interface, kemudian mengatur properties dari objek-objek yang digunakan dalam user interface, dan baru dilakukan penulisan kode program untuk menangani ke jadian-kejadian (event) (Wahana Komputer ; 2010).

Visual Basic 2010 merupakan salah satu bagian dari produk pemograman terbaru yang dikeluarkan oleh Microsoft, yaitu Microsoft Visual Studio 2010. Sebagai produk lingkungan pengembangan terintegrasi atau IDE andalan yang dikeluarkan oleh Microsoft (Wahana Komputer ;2010:2).

II.8. SQL Server 2008

II.8.1 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 Microsoft untuk ikut berkecimpung dalam persaingan dunia. (Wahana Komputer ;2010; 2)

SQL Server 2008 memiliki suatu GUI (*Graphic User Interface*) yang dapat digunakan untuk melakukan aktivitas sehari-hari berkaitan dengan database, seperti menulis T-SQL, melakukan backup dan restore database, melakukan security database terhadap aplikasi dan sebagainya. Pada GUI tersebut terdapat setingan terhadap SQL Server untuk bekerja lebih optimal. Settingannya juga bisa dilakukan dengan menggunakan script untuk memudahkan developer mengubah setting pada SQL Server 2008.

II.9. *Unified Modeling Language* (UML)

II.9.1 Pengertian *Unified Modeling Language* (UML)

Unified Modeling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual (Braun, *et. al.*2011). Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek (Whitten, *et. al.* 2004)

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam berkomunikasi beberapa aspek dalam sistem melalui sejumlah elemen grafis yang bisa dikombinasikan menjadi diagram. UML mempunyai banyak diagram

yang dapat mengakomodasi berbagai sudut pandang dari suatu perangkat lunak yang akan dibangun.(Yuni Sugiarti; 2013:36-37)

Dengan menggunakan UML, kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam bahasa pemrograman apapun. Tetapi karena UML juga menggunakan dan operation dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mengidentifikasi notasi dan syntax/semantik. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML syntax mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (*Object-Oriented Software Design*), Jim Rumbaugh OMT (*Object Modeling Technique*), dan Ivar Jacobson OOSE (*Object-Oriented Software Engineering*) (Yuni Sugiarti;2013:34).

Blok pembangunan utama UML adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainnya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorientasi objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para anggota team untuk bekerja sama dengan bahasa model yang sama dengan mengaplikasikan beragam sistem.

Intinya UML merupakan alat komunikasi yang konsisten dalam mendukung para pengembang sistem saat ini.

II.9.2. Use Case Diagram

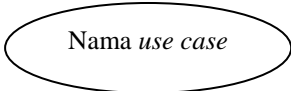
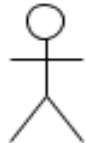

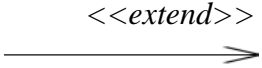
Dalam membuat sebuah sistem, langkah awal yang perlu dilakukan adalah menentukan kebutuhan. Terdapat dua jenis kebutuhan, yaitu kebutuhan fungsional dan kebutuhan nonfungsional. Kebutuhan fungsional adalah kebutuhan pengguna dan stakeholder sehari-hari yang akan dimiliki oleh sistem, dimana kebutuhan ini akan digunakan oleh pengguna *stakeholder*. Sedangkan kebutuhan nonfungsional adalah kebutuhan yang memperhatikan hal-hal berikut yaitu performansi, kemudahan dalam menggunakan sistem, kehandalan sistem, keamanan sistem, keuangan, legalitas, dan operasional.

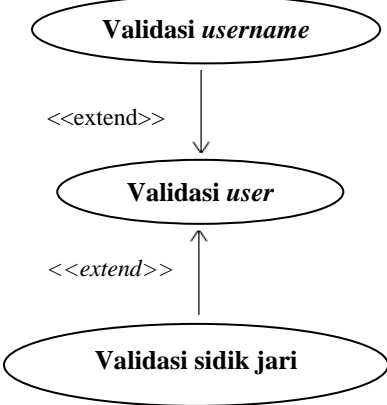

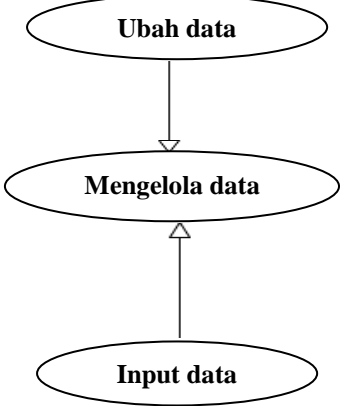
Kebutuhan fungsi akan digambarkan melalui sebuah diagram yang dinamakan diagram use case. *Use case* diagram atau diagram use case merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. Diagram *use case* mendeskripsikan sebuah interaksi antar satu atau lebih actor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, diagram use case digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan diagram use case, yaitu use case, actor dan relasi. Hal perlu diingat mengenai diagram use case adalah diagram use case bukan menggambarkan tampilan antarmuka (*user interface*), arsitektur dari sistem, kebutuhan nonfungsional, dan tujuan performansi. Sedangkan untuk penamaan

use case adalah nama didefinisikan sesimpel mungkin, dapat dipahami dan menggunakan kata kerja (Yuni Sugiarti; 2013:41)

Berikut ini pada tabel II.2. adalah simbol-simbol yang ada pada diagram *use case*:

Tabel II.2. Simbol – Simbol *Use Case* Diagram

| Simbol | Deskripsi |
|--|---|
| <p><i>Use case</i></p>  | <p>Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja diawal diawal frase nama <i>use case</i></p> |
| <p>Aktor / <i>actor</i></p>  <p>nama aktor</p> | <p>orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari <i>actor</i> adalah gambar orang, tapi <i>aktor</i> belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal nama <i>aktor</i>.</p> |
| <p>Asosiasi/ <i>association</i></p>  | <p>Komunikasi antara <i>actor</i> dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan aktor</p> |
| <p>Ekstensi / <i>extend</i></p>  | <p>Relasi <i>use case</i> tambahan sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu: mirip dengan prinsip <i>inheritance</i> pada pemrograman berorientasi objek; biasanya <i>use</i></p> |

| | |
|---|--|
| | <p><i>case</i> tambahan memiliki nama depan yang sama dengan <i>use case</i> yang ditambahkan, misal</p>  <pre> graph BT A([Validasi sidik jari]) -- "<<extend>>" --> B([Validasi user]) B -- "<<extend>>" --> C([Validasi username]) </pre> <p>Arah panah mengarah pada <i>use case</i> yang ditambahkan; biasanya <i>use case</i> yang menjadi <i>extend</i>-nya merupakan jenis yang sama dengan <i>use case</i> yang menjadi induknya.</p> |
| <p>Generalisasi/<i>generalization</i></p>  | <p>Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah <i>use case</i> dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:</p>  <pre> graph BT A([Input data]) --> B([Mengelola data]) B --> C([Ubah data]) </pre> <p>Arah panah mengarah pada <i>use case</i> yang menjadi generalisasinya (umum).</p> |

| | |
|---|---|
| <p>Menggunakan / <i>include</i> / <i>uses</i></p> <p>—————></p> <p><<<i>uses</i>>></p> <p>—————▷</p> | <p>Relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat <i>use case</i> ini.</p> <p>Ada dua sudut pandang yang cukup besar mengenai <i>include</i> di <i>use case</i> :</p> <p><i>include</i> berarti <i>use case</i> yang ditambahkan akan selalu dipanggil saat <i>use case</i> tambahan dijalankan.</p> <p><i>include</i> berarti <i>use case</i> yang tambahan akan selalu melakukan pengecekan apakah <i>use case</i> yang ditambahkan telah dijalankan sebelum <i>use case</i> tambahan dijalankan.</p> |
|---|---|

Sumber : (Rosa A.S, M.Shalahuddin; 2013; 155-158).

II.9.3. *Class Diagram* (**Diagram Kelas**)

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

1. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas.
2. Atribut mendeskripsikan property dengan sebaris teks didalam kotak kelas tersebut.
3. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas.

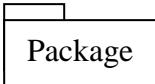
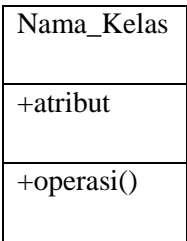

Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat diantara mereka. Diagram kelas juga menunjukkan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek. Diagram kelas menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, perwarisan, asosiasi, dan lain-lain (Yuni Sugiarti;2013:57).

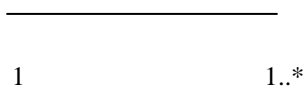

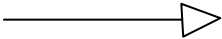
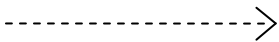
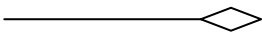
Kelas memiliki tiga area pokok :

1. Nama
2. Atribut
3. Operasi

Berikut adalah simbol-simbol yang ada pada diagram kelas :

Tabel II.3. Simbol-Simbol Diagram Kelas

| Simbol | Deskripsi |
|--|--|
| Package  | Package merupakan bungkusan dari satu kelas atau lebih |
| Operasi  | Kelas pada struktur system |
| Antarmuka / <i>interface</i>  Nama <i>interface</i> | Sama dengan konsep <i>interface</i> dalam pemrograman berorientasi objek |

| | |
|---|---|
| <p>Asosiasi / <i>association</i></p>  | <p>Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i></p> |
| <p>Asosiasi berarah / <i>Directed association</i></p>  | <p>Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i></p> |
| <p>Generalisasi</p>  | <p>Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)</p> |
| <p>Kebergantungan</p>  | <p>Relasi antar kelas dengan makna Kebergantungan antar kelas</p> |
| <p>Agregasi / <i>aggregation</i></p>  | <p>Relasi antar kelas dengan makna Semua bagian (<i>whole part</i>)</p> |

Sumber : (Yuni Sugiarti;2013:59)



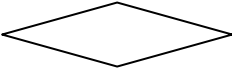


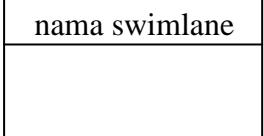
II.9.4. Activity Diagram (Aktivitas)

Diagram aktivitas atau *activity diagram* menggambarkan *workflow*(aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. *Activity diagram* menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir.

Activity Diagram merupakan state diagram khusus, dimana sebagian besar state adalah action dan sebagian besar transisi di-*trigger* oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses- proses dan jalur-jalur aktivitas dari level atas secara umum (Yuni Sugiarti;2013:75).

Berikut adalah tabel II.4. yang menggambarkan simbol-simbol yang digunakan pada diagram aktivitas :

Tabel II.4. Simbol *Activity Diagram*

| Simbol | Deskripsi |
|--|--|
| Status awal  | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal. |
| Aktivitas  | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja. |
| Percabangan / <i>decision</i>  | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu. |
| Penggabungan / <i>join</i>  | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu. |
| Status akhir  | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir. |
| Swimlane  | Memisahkan organisasi bisnis yang bertanggungjawab terhadap aktivitas yang terjadi. |

Sumber : (Rosa A.S, M.Shalahuddin; 2013; 161-163).


II.6.5. Sequence Diagram


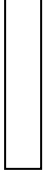




Diagram sekuen menggambarkan kelakuan/perilaku objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek.

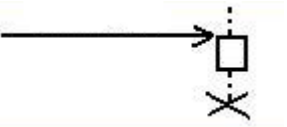
Banyaknya diagram Sekuence yang digambarkan adalah sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefenisikan interaksi jalanya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak use case didefenisikan maka diagram sekuen yang harus dibuat juga semakin banyak (Yuni Sugiarti;2013:69).

Berikut adalah tabel II.5 yang menerangkan simbol-simbol yang ada pada diagram sekuen:

Tabel II.5. Diagram Sequence

| Simbol | Deskripsi |
|---|--|
| <p data-bbox="325 1397 405 1424">Aktor</p>  <p data-bbox="427 1704 587 1731">nama aktor</p> | <p data-bbox="715 1397 1342 1704">orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari <i>actor</i> adalah gambar orang, tapi <i>actor</i> belum tentu merupakan orang;</p> <p data-bbox="715 1756 1342 1839">biasanya dinyatakan menggunakan kata benda di awal nama <i>actor</i>.</p> |

| | |
|--|--|
| <p>Garis hidup / <i>lifeline</i></p>  | <p>Menyatakan kehidupan suatu objek.</p> |
| <p>Objek</p> <div style="border: 1px solid black; padding: 5px; width: fit-content; margin: 10px auto;"> <p>nama objek :nama kelas</p> </div> | <p>Menyatakan objek yang berinteraksi pesan.</p> |
| <p>Waktu aktif</p>  | <p>Menyatakan objek dalam keadaan aktif dan berinteraksi, semua yang terhubung dengan waktu aktif ini adalah tahapan yang dilakukan didalamnya.</p> |
| <p>Pesan tipe create</p> <p style="text-align: center;"><<create>></p>  | <p>Menyatakan suatu objek membuat objek yang lain, arah panah mengarah pada objek yang dibuat</p> |
| <p>Pesan tipe call</p> <p style="text-align: center;">1 : nama_metode{ }</p>  | <p>Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri.</p> |
| <p>Pesan tipe send</p> <p style="text-align: center;">1 : masukan</p>  | <p>Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainnya, arah panah mengarah pada objek yang dikirim.</p> |
| <p>Pesan tipe return</p> <p style="text-align: center;">1 : keluaran</p>  | <p>Menyatakan bahwa suatu objek yang telah menjalankan suatu operasi atau metode menghasilkan suatu kembalian ke objek tertentu, arah panah mengarah pada objek yang menerima kembalian.</p> |
| <p>Pesan tipe destroy</p> | <p>Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada</p> |

| | |
|--|--|
| <p style="text-align: center;"><<destroy>></p>  | <p>objek yang diakhiri, sebaiknya jika ada create maka ada <i>destroy</i>.</p> |
|--|--|

Sumber : (Rosa A.S, M.Shalahuddin; 2013; 165-167).

II.10. Basis Data

Basis data adalah sekumpulan fakta berupa representasi tabel yang saling berhubungan dan disimpan dalam media penyimpanan secara digital. Suatu basis data terdiri dari sekumpulan tabel yang saling berelasi ataupun tidak berelasi. semua tabel tersebut merupakan representasi tempat untuk penyimpanan data yang mendukung fungsi dari basis data tersebut pada suatu sistem (Yudi Priadi ; 2014:1-2).

II.10.1. Basis Data Relational

Basis data *relational* diciptakan oleh seorang peneliti dari IBM yang bernama Dr. E. F. Codd, kemudian dikembangkan terus oleh peneliti lainnya hingga saat ini. Prinsipnya, model basis data *relational* digunakan sebagai suatu cara untuk mengelompokkan data dari sebuah kumpulan data yang besar. hal ini dapat dilakukan dengan cara menghapus duplikasi dari suatu data melalui proses yang disebut normalisasi. Proses ini terdiri dari beberapa langkah yang akan menjadi suatu bentuk normal. Hasilnya berupa bahasa umum untuk melakukan akses terhadap suatu basis data, yang disebut dengan *Structured Query Language* (SQL), sehingga dimungkinkan untuk melakukan *query* terhadap

organisasi struktur datanya. SQL sebagai bahasa pada basis data telah menjadi standar untuk semua perusahaan pengembang basis data hingga saat ini (Yudi Priadi ; 2014: 14).

II.11. Normalisasi Data

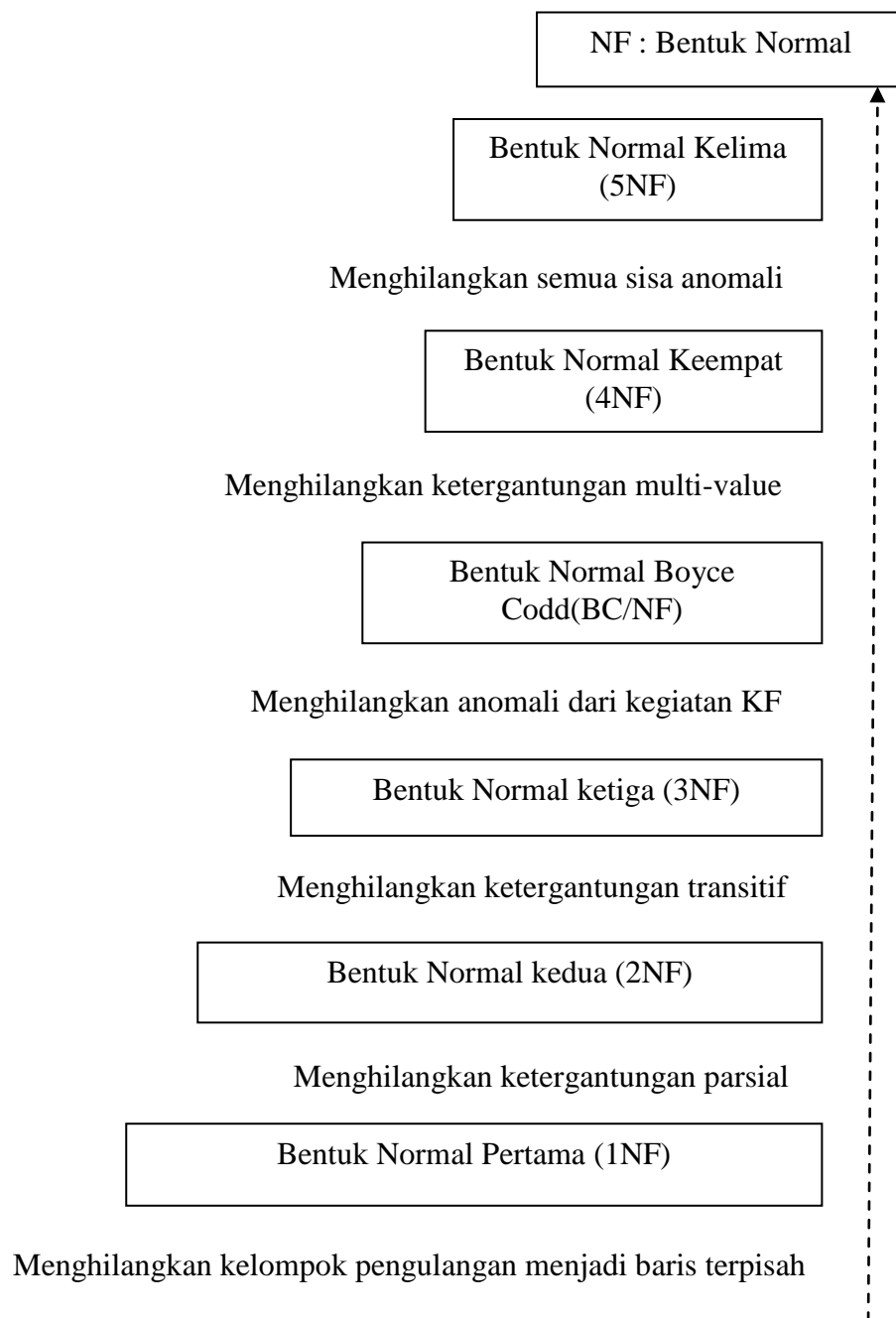
Normalisasi merupakan tahapan proses sistematis yang dilakukan pada struktur tabel basis data menjadi struktur tabel yang memiliki integritas data, sehingga tidak memiliki data anomali pada saat melakukan insert, delete, dan *update*. Kegiatan normalisasi adalah melakukan dekomposisi atau penguraian tabel beserta datanya, menjadi tabel yang normal menurut konsep RDBMS. Dekomposisi diawali dengan melakukan analisis pada suatu tabel atau beberapa contoh formulir yang sudah memiliki data lengkap untuk suatu basis data, tetapi masih dalam bentuk yang tidak normal (UNF). Oleh karena itu, agar dapat memenuhi syarat bentuk normal pertama (1NF), pada setiap barisnya diisikan suatu *value* dengan kelompok data yang sama, berdasarkan suatu atribut key. Dengan demikian, kelompok pengulangan dalam suatu baris dapat dihilangkan, karena sudah tidak terdapat *value* yang kosong untuk setiap *field* dan *record*-nya (Yudi Priadi ; 2014: 67- 68).

Setelah memenuhi syarat bentuk normal pertama (1NF), proses berikutnya adalah menghilangkan ketergantungan secara parsial, yaitu dengan cara melakukan dekomposisi tabel menjadi beberapa kelompok berdasarkan *field*

yang memiliki status sebagai *key*. Hal ini dapat dilakukan oleh salah satu *field* saja, dengan tetap tidak mengubah arti relasi dan ketergantungannya. Oleh sebab itu, disebut ketergantungan fungsional sebagian (*partially functional*), sehingga syarat bentuk normal kedua (2NF) sudah tercapai (Yudi Priadi ; 2014:68).

Bentuk normal kedua (2NF) merupakan syarat yang harus dimiliki untuk menuju bentuk normal ketiga (3NF). Pada proses ini, dilakukan dengan menghilangkan ketergantungan secara transitif, yaitu konsep untuk tabel dari hasil relasi yang didalamnya terdapat ketergantungan secara tidak langsung pada beberapa atributnya. Pada umumnya, proses normalisasi sudah dapat tercapai pada bentuk normal ketiga (3NF), yaitu dengan menghasilkan tabel yang tidak mengalami anomali basis data pada saat proses *insert*, *delete*, dan *update* (Yudi Priadi ; 2014:68).

Berikut ini pada gambar II.2. adalah tahapan aturan proses normalisasi :



| |
|---------------------------|
| UNF : Bentuk Tidak Normal |
|---------------------------|

Gambar II.2. Tahapan Aturan Proses Normalisasi

Sumber (Yudi Priyadi;2014:72)

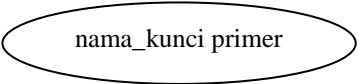
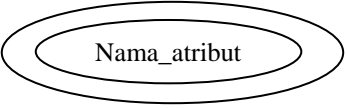

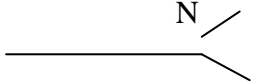
II.12. ERD (*Entity Relationship Diagram*)

Pemodelan awal basis data yang paling banyak digunakan adalah menggunakan ERD (*Entity Relationship Diagram*). ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relasional. Sehingga jika penyimpanan basis data menggunakan OODBMS maka perancangan basis data perlu menggunakan ERD. ERD memiliki beberapa aliran notasi seperti notasi Chen (dikembangkan oleh Peter Chen), Barker (dikembangkan oleh Richard Barker, Ian Palmer, Harry Ellis), notasi Crow's Foot, dan beberapa notasi lain. Namun yang banyak digunakan adalah notasi Chen. Berikut pada Tabel II.6. adalah simbol – simbol yang digunakan pada ERD dengan notasi Chen : (Rosa A.S, M.Shalahuddin; 2013; 50-51).

Tabel II.6. Simbol-Simbol ERD

| Simbol | Deskripsi |
|---|---|
| Entitas / <i>entity</i> <div style="border: 1px solid black; width: 100px; height: 20px; margin: 0 auto; text-align: center;">nama_entitas</div> | Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data; benda yang memiliki data dan harus disimpan datanya agar dapat diakses oleh aplikasi komputer; penamaan entitas biasanya lebih ke kata benda dan belum merupakan nama tabel. |
| Atribut | <i>Field</i> atau kolom data yang butuh disimpan |

nama_atribut

| | |
|--|--|
| | dalam suatu entitas. |
| <p>Atribut kunci primer</p>  | <p><i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses <i>record</i> yang diinginkan; biasanya berupa id; kunci primer dapat lebih dari satu kolom, asalkan kombinasi dari beberapa kolom tersebut dapat bersifat unik (berbeda tanpa ada yang sama).</p> |
| <p>Atribut multivalai / <i>multivalue</i></p>  | <p><i>Field</i> atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu.</p> |
| <p>Relasi</p>  | <p>Relasi yang menghubungkan antar entitas; biasanya diawali dengan kata kerja.</p> |
| <p>Asosiasi / <i>association</i></p>  | <p>Penghubung antara relasi dan entitas dimana dikedua ujungnya memiliki <i>multiplicity</i> kemungkinan jumlah pemakaian</p> <p>kemungkinan jumlah maksimum keterhubungan antara entitas satu dengan yang lain disebut kardinalitas. Misalkan ada kardinalitas 1 ke N atau sering disebut dengan <i>one to many</i> menghubungkan entitas A dan B maka.</p> |

Sumber : (Rosa A.S, M.Shalahuddin; 2013; 50-51).

Ada beberapa relasi yang dapat terjadi, yaitu :

- Satu ke satu (1:1), mempunyai arti bahwa setiap satu entitas pada himpunan entitas A hanya dapat berelasi dengan satu saja pada himpunan entitas B.
- Satu ke banyak (1:N), mempunyai arti bahwa setiap satu entitas pada himpunan entitas A dapat berelasi dengan banyak entitas pada himpunan entitas B.
- Banyak ke satu (N:1), mempunyai arti bahwa beberapa entitas pada himpunan entitas A hanya dapat berelasi dengan satu entitas saja pada himpunan entitas B.
- Banyak ke banyak (N:N), mempunyai arti bahwa beberapa entitas pada himpunan entitas A dapat berelasi dengan banyak entitas pada himpunan entitas B.

