BAB II

TINJAUAN PUSTAKA

II.1. Konsep Dasar Sistem

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

II.1.1. Syarat - Syarat Sistem

Adapun beberapa syarat - syarat sistem yaitu sebagai berikut :

1. Sistem harus dibentuk untuk menyelesaikan tujuan.
2. Elemen sistem harus mempunyai rencana yang ditetapkan.
3. Adanya hubungan di antara elemen sistem.
4. Unsur dasar dari proses (arus informasi, energi, dan material) lebih penting dari pada elemen sistem.
5. Tujuan organisasi lebih penting dari pada tujuan elemen (Asbon Hendra;2012:158)

II.1.2. Karakteristik Sistem

Ada beberapa karakteristik dalam sistem diantaranya adalah :

1. Komponen

Suatu sistem terjadi dari sejumlah komponen yang saling berinteraksi, bekerja sama membentuk satu kesatuan.

2. Batas Sistem

Batas sistem merupakan daerah yang membatasai antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya.

3. Lingkungan Luar Sistem

Lingkungan luar sistem merupakan segala sesuatu di luar batas sistem yang mempengaruhi operasi dari suatu sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan atau merugikan.

4. Penghubung Sistem

Penghubung sistem merupakan media penghubung antara satu subsistem dengan susbsitem yang lainnya untuk membentuk satu kesatuan sehingga sumber - sumber daya mengalir dari subsistem yang satu ke subsistem yang lainnya. Dengan kata lain, *output* dari suatu subsistem akan menjadi *input* dari subsistem yang lainnya.

5. Masukan Sistem *(Input)*

Input merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa Masukan Perawatan adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi.

6. Keluaran Sistem *(Output)*

Output merupakan hasil dari energi yang diolah oleh sistem, meliputi *output* yang berguna, Contohnya informasi yang dikeluarkan oleh komputer. Dan *output* yang tidak berguna dikenal sebagai sisa pembuangan, contohnya panas yang dikeluarkan oleh komputer.

7. Pengolah Sistem

Pengolah sistem merupakan bagian yang memproses masukan untuk menjadi keluaran yang diinginkan. Contoh CPU pada komputer, bagian produksi yang mengubah bahan baku menjadi barang jadi, serta bagian akuntansi yang mengolah data transaksi menjadi laporan keuangan.

8. Tujuan Sistem

Setiap sistem pasti merupakan tujuan ataupun sasaran yang memengarungi *input* yang dibutuhkan dan *output* yang dihasilkan. Dengan kata lain, suatu sistem akan dikatakan berhasil kalau pengoperasian sistem itu mengenai sasaran atau tujuannya (Asbon Hendra; 2012: 158).

II.1.3. Klasifikasi Sistem

Dalam sistem informasi ada beberapa klasifikasi sistem diantaranya adalah :

1. Sistem Abstrak

Sistem yang berupa pemikiran atau ide - ide yang tidak tampak secara fisik. Sebagai contoh, sistem Teologia merupakan suatu sistem yang menggambarkan hubungan Tuhan dengan manusia.

2. Sistem Fisik

Merupakan sistem yang ada secara fisik sehingga setiap makhluk dapat melihatnya. Contohnya, sistem komputer, sistem akuntansi, sistem produksi, dan lain - lain.

3. Sistem Alamiah

Sistem yang terjadi melauli proses alam, dalam artian tidak dibuat oleh manusia, seperti sistem tata surya, sistem galaksi, sistem reproduksi, dan lain - lain.

II.2. Konsep Dasar Informasi

Informasi merupakan data yang telah diproses menjadi bentuk yang memiliki arti bagi penerima dan dapat berupa fakta, suatu nilai yang bermanfaat. Jadi, ada suatu proses transformasi data menjadi suatu

informasi = *input* – proses – *output.*

Data merupakan *raw material* untuk suatu informasi. Perbedaan informasi dan data sangat relatif, tergantung pada nilai gunanya bagi manajemen yang memerlukan. Suatu informasi bagi level manajemen tertentu bisa menjdi data bagi manajemen level di atasnya, atau sebaliknya (Asbon Hendra; 2012: 167).

II.2.1. Kualitas Informasi

Kualitas informasi merupakan satuan ukuran informasi, tergantung representasi. Untuk representasi biner, satuannya bit, byte, word, dan lain - lain.

Kualitas informasi tergantung dari tiga hal, yaitu informasi harus :

1. Akurat, berarti informasi harus bebas dari kesalahan - kesalahan dan tidak biasa atau menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya.

2. Tepat pada waktunya, berarti informasi yang datang pada penerima tidak boleh terlambat.
3. Relevan, berarti informasi tersebut mempunyai manfaat untuk pemakainnya. Relevansi informasi untuk tiap - tiap orang satu dengan yang lainnya berbeda.

II.3. Sistem Pakar

Sistem pakar adalah sistem komputer yang ditujukan 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).

Sistem pakar merupakan salah satu aplikasi pertama yang muncul dari riset awal dalam bidang kecerdasan buatan. Penjelasan dari penalaran sistem pakar merupakan salah satu aplikasi pertama dari generasi bahasa alami. Hal ini disebabkan oleh kebutuhan untuk penjelasan adalah nyata, generasi dari aplikasi basis pengetahuan seperti penalaran harus secara relatif dan langsung (Rika Rosnelly; 2012: 105).

II.3.1. Kelebihan Sistem Pakar

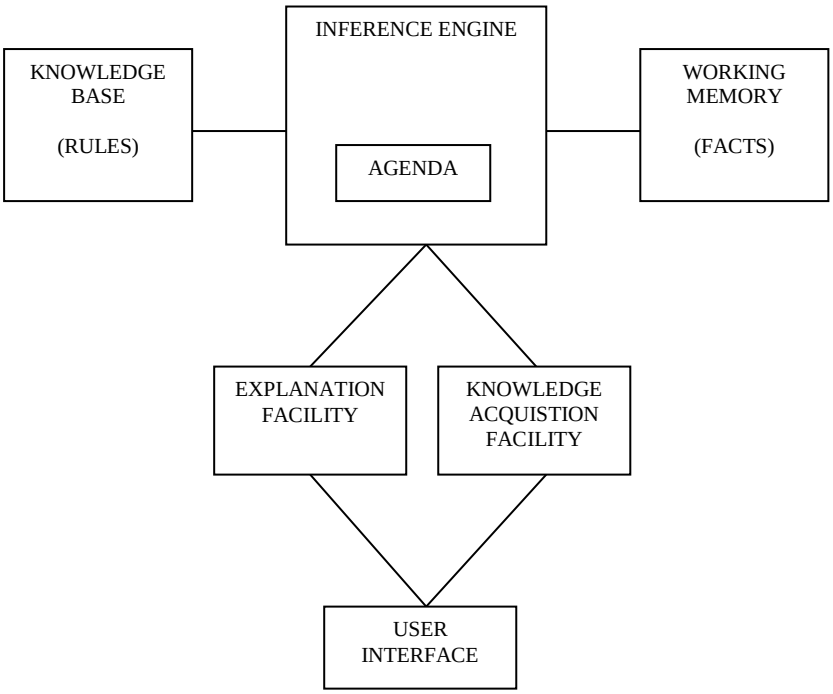
Sistem pakar memiliki beberapa fitur menarik yang merupakan kelebihannya, seperti :

1. **Meningkatkan ketersediaan.** Kepakaran atau keahlian menjadi tersedia dalam sistem komputer. Dapat dikatakan bahwa sistem pakar merupakan produksi kepakaran secara masal.

2. **Mengurangi biaya.** Biaya yang diperlukan untuk menyediakan keahlian satu per satu orang *user* menjadi berkurang.
3. **Mengurangi bahaya.** Sistem pakar dapat digunakan di lingkungan yang mungkin berbahaya bagi manusia.
4. **Permanen.** Sistem pakar dan pengetahuan yang terdapat didalamnya bersifat lebih permanen dibandingkan manusia yang dapat merasa lelah, bosan, dan pengetahuannya hilang saat sang pakar meninggal dunia.
5. **Keahlian multipel.** 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 bebarapa pakar dapat melebihi pengetahuan satu orang pakar.
6. **Meningkatkan kehandalan.** Sistem pakar meningkatkan kepercayaan dengan memberikan hasil yang benar sebagai alternatif pendapat dari seseorang pakar atau sebagai penengah jika terjadi konflik antara beberapa pakar. Namun hal tersebut 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.** Sistem pakar dapat menjelaskan detail proses penalaran yang dilakukan hingga mencapai suatu kesimpulan. Seorang mungkin saja terlalu lelah, tidak bersedia atau tidak mampu melakukannya setiap waktu. Hal ini akan meningkatkan tingkat kepercayaan bahwa kesimpulan yang dihasilkan adalah benar (Rika Rosnelly; 2012: 5).

II.3.2. Struktur Sistem Pakar

Adapun struktur sistem pakar dapat dilihat pada gambar II.1.



Gambar II.1. ***Struktur Sistem Pakar***
(Sumber : Rika Rosnelly; 2012: 13)

Komponen yang terdapat dalam struktur sistem pakar ini adalah :

1. *Knowledge Base* (Basis Pengetahuan)

Basis pengetahuan mengandung pengetahuan untuk pemahaman, formulasi, dan penyelesaian masalah. Komponen sistem pakar disusun atas dua elemen dasar, yaitu fakta dan aturan. Fakta merupakan informasi tentang objek dalam area permasalahan tertentu, sedangkan aturan merupakan informasi tentang cara bagaimana memperoleh fakta baru dari fakta yang telah diketahui. Pada struktur sistem pakar diatas, *knowledge base* disini untuk menyimpan pengetahuan dari

pakar berupa rule/aturan (if <kondisi> then <aksi> atau dapat juga disebut conditional-action rules).

2. *Inference Engine* (Mesin Inferensi)

Mesin inferensi merupakan otak dari sebuah sistem pakar dan dikenal juga dengan sebuah struktur control atau dalam sistem pakar berbasis kaidah. Komponen ini mengandung mekanisme pola pikir dan penalaran yang digunakan oleh pakar dalam menyelesaikan suatu masalah. Mesin inferensi disini adalah *processor* pada sistem pakar yang mencocokan bagian kondisi rule yang tersimpan di dalam *knowledge base* dengan fakta yang tersimpan di *working memory*.

3. *Working Memory*

Berguna untuk menyimpan fakta yang dihasilkan oleh *inference engine* dengan penambahan parameter berupa derajat kepercayaan atau juga dikatakan sebagai global database dari fakta yang digunakan oleh rule - rule yang ada.

4. *Explanation Facility*

Menyediakan kebenaran dari solusi yang dihasilkan kepada user *(reasoning chain).*

5. *Knowledge Acquisition Facility*

Meliputi proses pengumpulan, pemindahan dan perubahan dari kemampuan pemecahan masalah seorang pakar atau sumber pengetahuan terdokumentasi ke program komputer, yang bertujuan untuk memperbaiki atau mengembangkan basis pengetahuan.

6. *User Interface*

Mekanisme untuk memberi kesempatan kepada user dan sistem pakar untuk berkomunikasi. Antar muka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu antarmuka menerima informasi dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai.

II.3.3. Karakteristik Sistem Pakar

Sistem pakar pada umunya dirancang untuk memenuhi beberapa karakteristik umum berikut ini :

1. **Kinerja sangat baik.** Sistem harus mampu memberikan respon berupa saran dengan tingkat kualitas yang sama dengan seorang pakar atau melebihinya.
2. **Waktu respon yang baik.** Sistem juga harus harus bekerja dalam waktu yang sama baiknya atau lebih cepat dibandingkan dengan seorang pakar dalam menghasilkan keputusan. Hal ini sangat penting terutama pada sistem waktu nyata.
3. **Dapat diandalkan.** Sistem harus dapat diandalkan dan tidak mudah rusak.
4. **Dapat dipahami.** Sistem harus mampu menjelaskan langkah - langkah penalaran yang dilakukannya seperti seorang pakar. Hal ini penting untuk beberapa alasan, yaitu :
   a. Dimungkinkan bahwa sistem pakar berkaitan dengan nyawa manusia atau properti lainnya harus dapat menjelaskan mengapa dihasilkan suatu kesimpulan tertentu.

b. Untuk mengkonfirmasi bahwa pengetahuan pakar telah disimpulkan dengan benar dan digunakan oleh sistem dengan benar pula. Hal ini penting dalam proses debugging pengetahuan yang mungkin salah karena pengetikan atau pemahaman yang salah dari *knowledge engineer*.

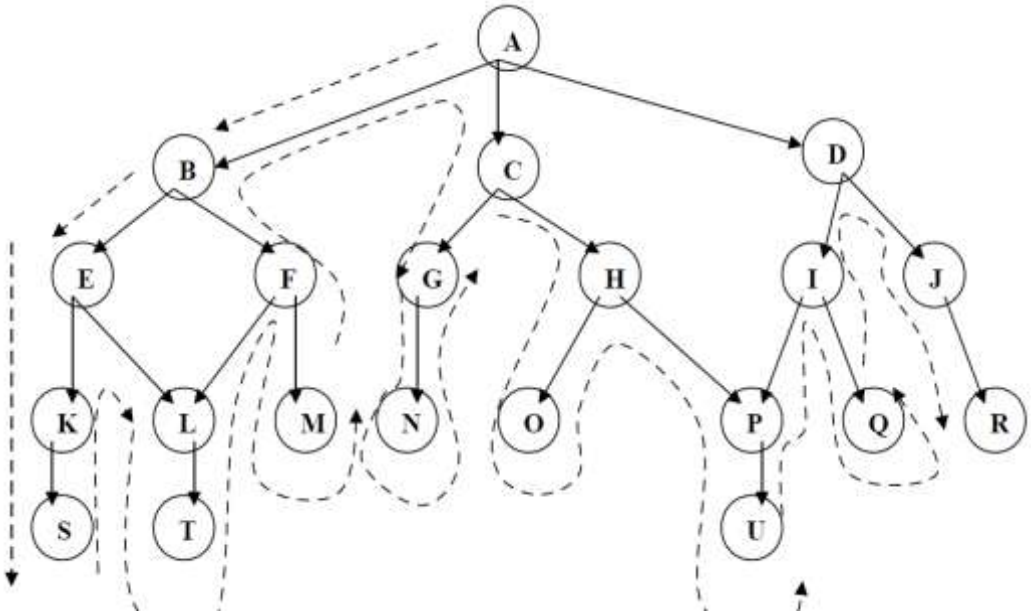
5. **Fleksibel.** Sistem harus menyediakan mekanisme untuk menambah, mengubah, dan menghapus pengetahuan (Rika Rosnelly; 2012: 20).

II.4. Metode *(Forward Chaining)*

Metode *(Forward Chaining)* adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan bagian *IF* dari rule *IF-THEN*. Bila ada fakta yang cocok dengan bagian *IF*, maka rule tersebut dieksekusi. Bila sebuah *rule* dieksekusi, maka sebuah fakta baru (bagian *THEN*) ditambah kedalam data base. Setiap kali pencocokan dimulai dari rule teratas. Setiap *rule* hanya boleh dieksekusi sekali saja. Proses pencocokan berhenti bila tidak ada lagi *rule* yang bisa dieksekusi. Metode pencarian yang digunakan adalah DFS (*Depth-First-Search*), BFS (*Breadth-First Search*), *Best-First-Search*(Sutejo,T,dkk; 2011:171-173).

DFS (*Depth-First-Search*) yaitu algoritma pencarian simpul dalam grafik secara travelsal yang dimulai dari simpul akar dan mengecek simpul anaknya yang pertama, setelah itu algoritma mengecek simpul anak dari simpul anak yang pertama tersebut, hingga mencapai simpul daun atau simpul tujuan, Jika solusi belum ditemukan, algoritma melakukan runut balik (*backtracking*) kesimpul orang tuanya yang paling baru diperiksa dan mengecek simpul anaknya yang

belum diperiksa, denikian seterusnya hingga simpul solusi ditemukan. Depth First Serch jauh lebih efisien untuk ruang pencarian dengan banyak percabangan, karena tidak perlu mengefaluasi semua simpul pada tingkat tertentu pada saat dimulainya pencarian.

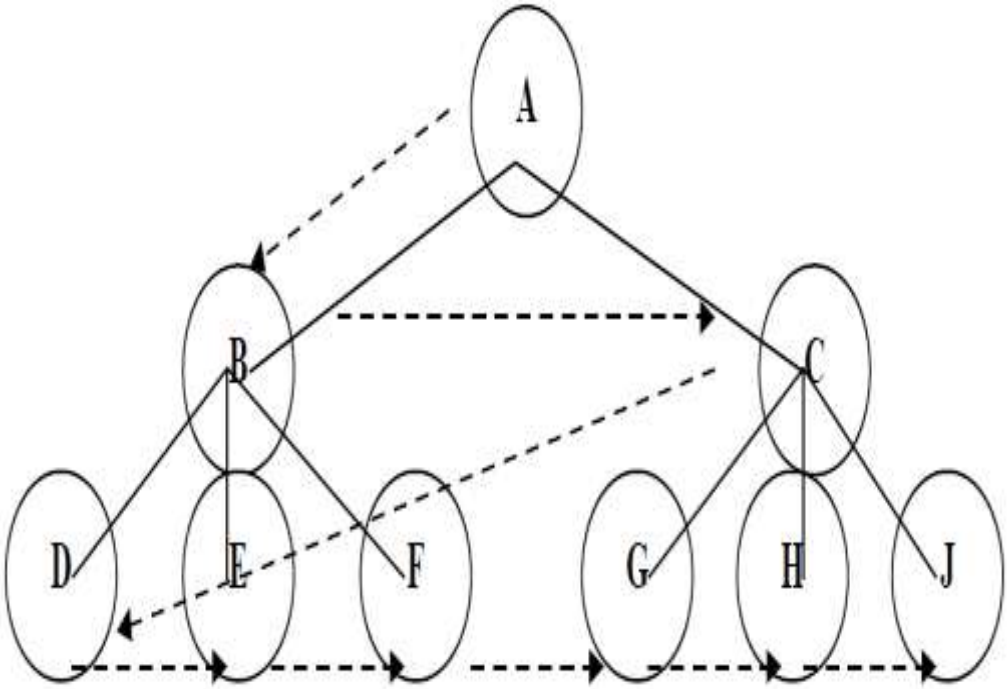


Gambar II.2. Depth First Search

(Sumber : Hendry ; 2011 : 43*)*

BFS (*Breadth First Search*)yaitu algoritma pencarian simpul dalam grafik (pohon) secara travelsal yang dimulai dari simpul akar dan mengecek semua simpul-simpul tetangganya. Setelah itu, dari tiap simpul tetangganya, algoritma akan terus mencek semua simpul tetangganya yang belum dicek, demikian seterusnya hingga menemukan simpul tujuan *Breadt First Search*. Interpreter kaidah mulai dari fakta yang ada yaitu hipotesa kemudian kaidah bagian *THEN* mulai di uji untuk mendukung hipotesa awal. Jika ditemulan maka kaidah *IF* yang cocok digunakan untuk menghasilkan hipotesa yang baru. Kemudian proses

berantai terus diulang, mengumpulkan bukti yang mendukung sehingga hipotesa terbukti kebenarannya.

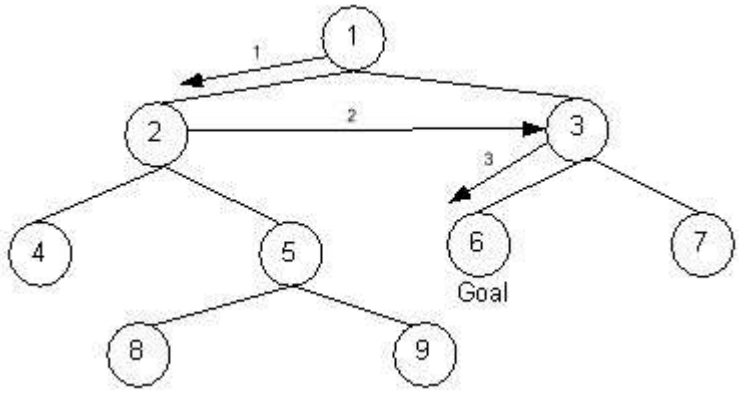


Gambar II.3. Breadth First Search

(Sumber : Hendry ; 2011 : 43*)*

BFS (*Best First Search*) adalah teknik penelusuran yang menggunakan pengetahuan akan suatu masalah untuk melakukan panduan pencarian kearah node tempat dimana solusi berada. Pencarian jenis ini dikenal sebagai *heuristic*. Pendekatan yang dilakukan adalah mencari solusi yang terbaik berdasarkan pengetahuan yang dimiliki sehingga penelusuran dapat ditentukan harus dimulai darimana dan bagaimana menggunakan proses terbaik untuk mencari solusi. Keuntungannya dapat mengurangi beban komputasi karena hanya solusi yang

memberikan harapan saja yang diuji dan berhenti apabila solusi mendekati yang terbaik.



Gambar II.4. Best First Search

(Sumber : Hendry ; 2011 : 43*)*

II.5. Penyakit Pada Tanaman Jagung

Penyakit pada tanaman jagung sering kali membuat para petani kecewa karena hasil yang didapatkan kurang memuaskan, dengan adanya penyakit yang menyerang pada tanamannya. Petani kadang hanya mengandalkan pupuk saja untuk memperoleh hasil panen yang memuaskan, namun bukan itu saja yang perlu di lakukan. Perlu diketahui penyakit yang menyerang pada tanaman dan apa penyebab serta bagaimana cara pengendalianya. Maka dibuat suatu sistem pakar mengenai penyakit tanaman jagung agar semua masalah yang terjadi dapat terselesaikan dengan waktu yang cepat.

II.6. *Microsoft Visual Basic* Dan *Microsoft Sql Server*

II.6.1. *Visual Basic 2008*

Visual Basic 2008 merupakan salah satu bahasa pemrograman yang handal dan banyak digunakan oleh pengembangan untuk membangun berbagai macam aplikasi *windows. Visual basic 2008* merupakan aplikasi pemrograman yang menggunakan teknologi. *NET Framework 3.5.* Teknologi. *NET Framework 3.5* merupakan komponen *windows* yang terintegrasi serta mendukung pembuatan. Penggunaan aplikasi, dan halaman *web*. Teknologi.*Net Framework 3.5* mempunyai 2 komponen utama, yaitu CLR (*Common Language Runtime*) dan *Class Library* adalah kelas pustaka atau perintah yang digunakan untuk membangun aplikasi (Wahana Komputer:2010:2)

II.6.2. *Microsoft SQL Server*

Bahasa *query* merupakan bahasa khusus yang digunakan untuk melakukan manipulasi dan menanyakan pertanyaan (*query*) yang berhubungan dengan bahasa pemrograman, dimana bahasa *query* tidak memiliki kemampuan untuk menyelesaikan banyak masalah seperti bahasa pemrograman pada umumnya. Dalam pemrograman basis data, salah satu bahasa yang harus kita kuasai adalah *SQL*, merupakan bahasa komputer standart yang digunakan untuk berkomunikasi dengan sistem manajemen basis data *relasional* (RDBMS). (Ema Utami dan Anggi Dwi Hartanto : 2012:63).

II.7. UML (*Unified Modeling Languange)*

II.7.1. Pengenalan UML (*Unified Modeling Languange)*

Pemodelan (*modeling*)sesungguhnya untuk peyederhana permasalahan. Adapun tujuan pemodelan (dalam kerangka pengembangan sistem/perangkat lunak aplikasi)adalah sebagai saran visualisasi dan komunikasi antar anggota tim pengembang(saat seorang analis/perancang sistem/perangkat lunak bekerja dalam tim yang beranggotakan beberapa/banyak anggota)serta sebagai sarana dokumentasi(bermanfaat untuk menelaah perilaku sistem secara seksama dan bermanfaat untuk menguji (*testing*) sistem yang telah selesai dikembangkan.

Dalam hal ini kita sebagai perancang sistem/perangkat lunak kita menggambarkan komponen-komponen sistem/perangkat lunak dalam bentuk-bentuk geometri tertentu, misalnya untuk menggambarkan suatu kelas dalam aplikasi kita membentuk empat persegi panjang. Untuk menggambarkan hubungan antar kelas kita menggunakan garis lurus(Adi Nugroho;2009:5).

II.7.2. Tujuan Pemanfaatan UML (*Unified Modeling Languange)*

Berikut tujuan utama dalam desain UML adalah :

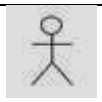
1. Menyediakan bagi pengguna (analisis dan desain sistem) suatu bahasa pemodelan visual yang ekspresif sehingga mereka dapat mengembangkan dan melakukan pertukaran model data yang bermakna.
2. Menyediakan mekanisme yang spesialisasi untuk memperluas konsep inti.
3. Karena merupakan bahasa pemodelan visual dalam proses pembangunannya maka UML bersifat independen terhadap bahasa pemograman tertentu.
4. Memberikan dasar formal untuk pemahaman bahasa pemodelan.

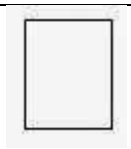
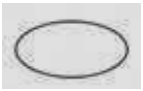
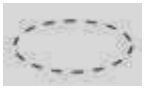
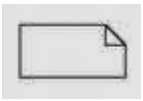
5. Mendorong pertumbuhan pasar terhadap penggunaan alat desain sistem yang berorientasi objek.
6. Mendukung konsep pembangunan tingkat yang lebih tinggi seperti kolaborasi, kerangka, pola dan komponen terhadap suatu sistem.
7. Memiliki integrasi praktik terbaik (Jurnal Informatika Mulawarman; 2011: 2).

II.7.3. *Use Case Diagram*

Menurut Imam Adi Nugaha (2012) Notasi atau simbol pembentuk *use case diagram* dapat dilihat pada tabel II. 1. Berikut ini :

Tabel II.1. Simbol-simbol Use Case Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *use case*.
2		*Dependency*	Hubungan dimana perubahan yang terjadi pada suatu elemen mandiri *(independent)* akan mempengaruhi elemen yang bergantung padanya elemen yang tidak mandiri (*independent*).
3		*Generalization*	Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
4		*Include*	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
5		*Extend*	Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.
6		*Association*	Apa yang menghubungkan antara objek satu dengan objek lainnya.
7		*System*	Menspesifikasikan paket yang menampilkan sistem secara terbatas.

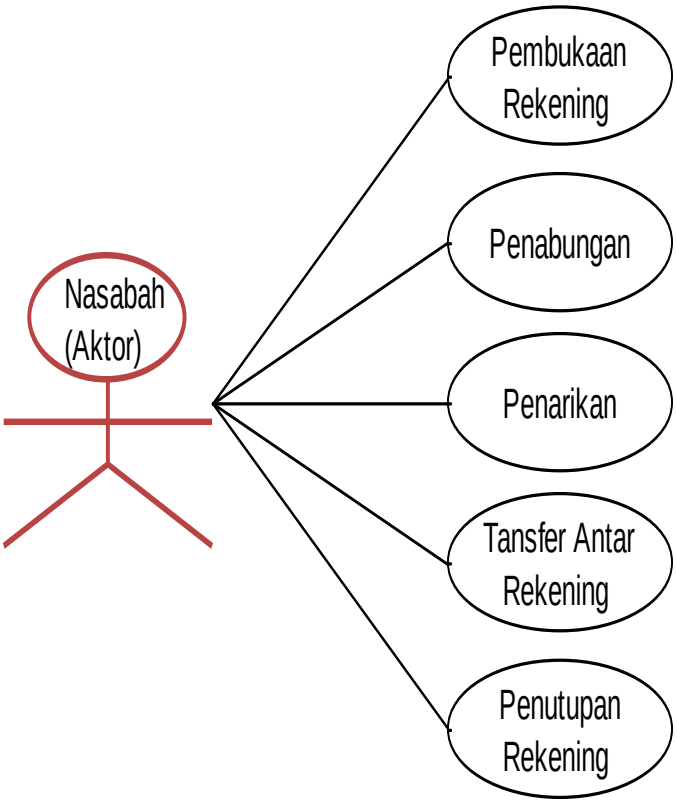
			
8		*Use Case*	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
9		*Collaboration*	Interaksi aturan-aturan dan elemen lain yang bekerja sama untuk menyediakan prilaku yang lebih besar dari jumlah dan elemen-elemennya (sinergi).
10		*Note*	Elemen fisik yang eksis saat aplikasi dijalankan dan mencerminkan suatu sumber daya komputasi

Segala sesuatu yang secara akademis dikembangkan pada umumnya berawal dari suatu konsep. Analisis kebutuhan ini adalah tahap konseptualisasi, yaitu suatu tahap yang mengharuskan analis dan perancang sistem untuk berusaha tahu secara pasti mengenai hal yang menjadi kebutuhan dan harapan pengguna sehingga kelak aplikasi yang dibuat memang akan digunakan oleh pengguna (*user*) serta akan memuaskan kebutuhan dan harapannya.

Selanjutnya, *use case diagram* tidak hanya sangat penting pada saat analisis, tetapi juga sangat penting dalam tahap perancangan (*design*), untuk mencari kelas-kelas yang terlibat alam aplikasi, dan untuk melakukan pengujian (*testing*).

Saat akan mengembangkan *use case diagram*, hal yang pertama kali harus dilakukan adalah mengenali *actor* untuk sistem yang sedang dikembangkan. Dalam hal ini, ada beberapa karakteristik untuk para *actor*, yaitu *actor* yang ada di luar sistem yang sedang dikembangkan dan *actor* yang berinteraksi dengan sistem

yang sedang dikembangkan (Adi Nugroho ; 2009 : 7) dapat dilihat pada gambar II.5 berikut ini :

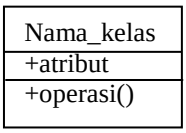
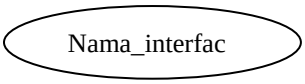
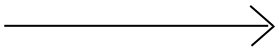
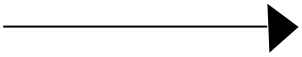
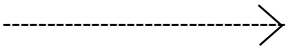


Gambar II.5. Contoh *Use Case Diagram*
(Sumber : Adi Nugroho ; 2009 : 8)

II.7.4. *Class Diagram*

Menurut Imam Adi Nugaha (2012) Notasi atau simbol pembentuk *class diagram* dapat dilihat pada tabel II. 2. Berikut ini :

Tabel II.2. Simbol-simbol Class Diagram

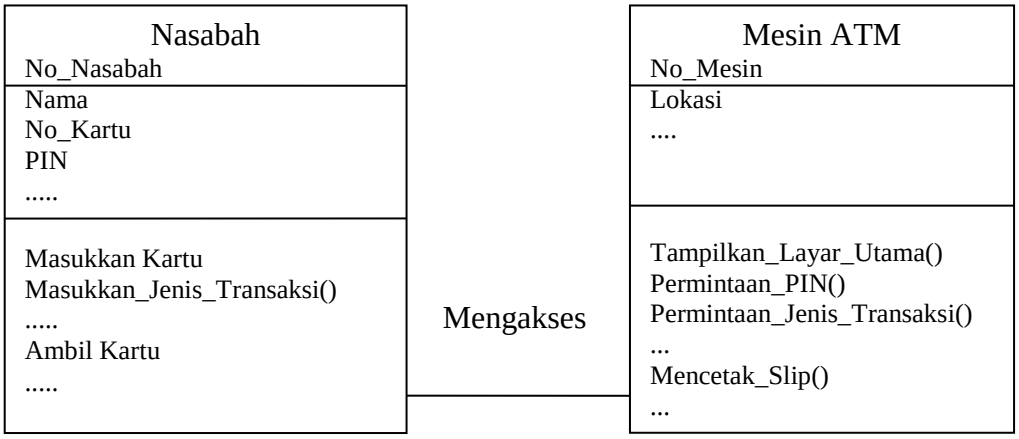
SIMBOL	DESKRIPSI
Kelas 	Kelas pada struktur sistem
Antarmuka / Interface 	Sama dengan konsep interface dalam pemrograman berorientasi objek
asosiasi / association 	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*.
Asosiasi berarah / directed association 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity.*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus).
Kebergantungan / *dependency* 	Relasi antar kelas dengan makna kebergantungan antar kelas.
Agregasi / *aggregation*	Relasi antar kelas dengan makna

Class didefenisikan sebagai kumpulan/himpunan objek yang memiliki kesamaan dalam *atribut/properti*, perilaku (operasi), serta cara berhubungan dengain objek lain (Adi Nugroho ; 2009 : 18).

Selain itu, kita juga mendefenisikan objek sebagai konsep, abstraksi dari sesuatu dengan batas nyata, sehingga kita dapat menggambarkan secara sistematis. Pemahaman objek memiliki dua fungsi, yaitu :

a. Memudahkan untuk mempelajari secara seksama hal-hal yang ada di dunia nyata.

b. Menyediakan suatu dasar yang kuat dalam implementasi ke dalam sistem terkomputerisasi (Adi Nugroho ; 2009 :17) dapat dilihat pada gambar II.6 berikut ini :

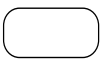


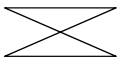
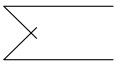
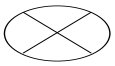
Gambar II.6. Contoh *Class Diagram*
(Sumber : Adi Nugroho ; 2009: 39)

II.7.5. *Activity Diagram*

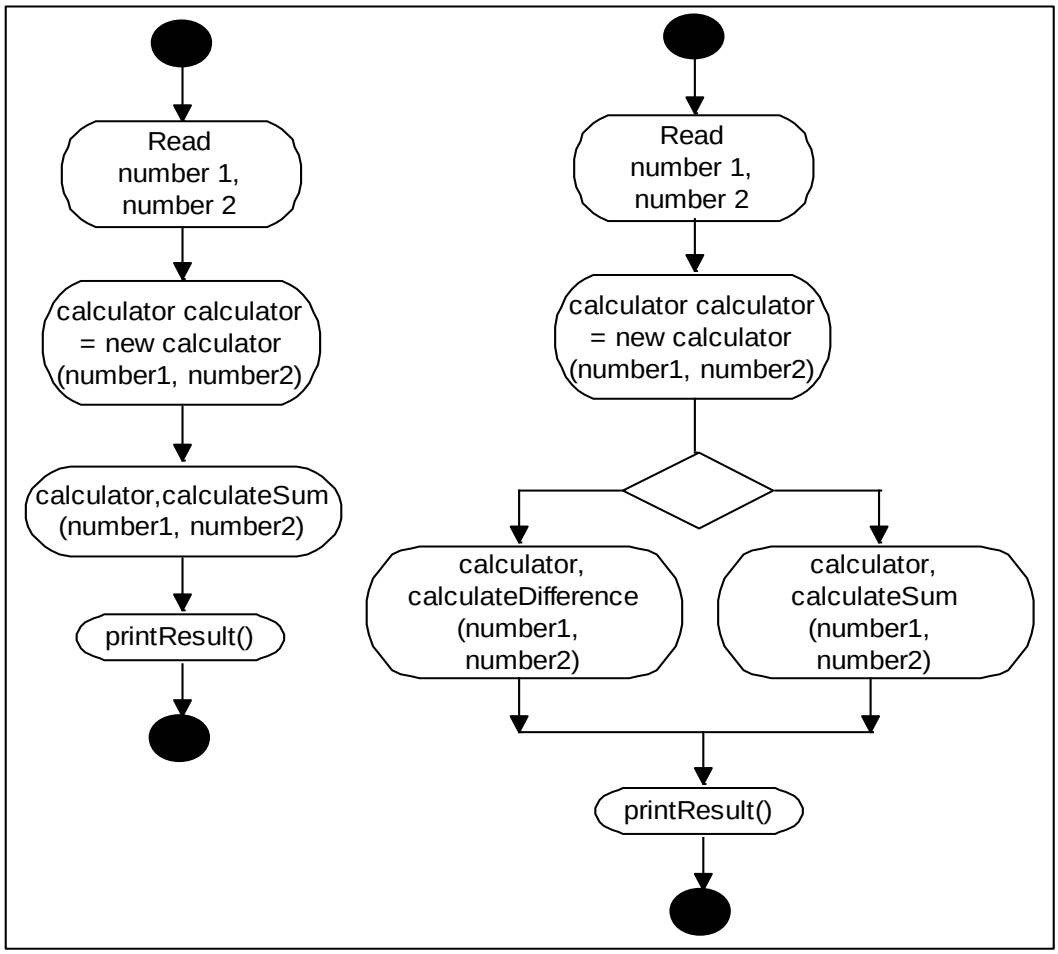
Menurut Imam Adi Nugaha (2012) Notasi atau simbol pembentuk *activity diagram* dapat dilihat pada tabel II.3. Berikut ini :

Tabel II.3. Simbol-simbol Activity Diagram

SIMBOL	KETERANGAN
	Titik awal
	Titik akhir
	Activity
	Pilihan untuk pengambilan keputusan
	Fork; digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.

	Rake; menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (Flow Final)

Apakah langkah yang harus kita lakukan selanjutnya setelah kita membuat *use case diagram* ? *use case diagram* merupakan gambaran menyeluruh dan pada umumnya sangatlah tidak terperinci. Oleh karena itu, kita harus memperinci lagi perilaku sistem untuk masing-masing *use case* yang ada. Apa perkakas *(tool)* yang bisa kita gunakan ? jika kasus kita cukup sederhana, mungkin kita bisa menggunakan skenario seperti yang tercantum berikut, sementara jika kasusnya cukup kompleks, kita mungkin bisa menggunakan *activity diagram* agar bisa mendapatkan gambaran yang lebih menyeluruh (Adi Nugroho ; 2009 : 10) dapat dilihat pada gambar II. 7 berikut ini :



Gambar II.7. Contoh *Activity Diagram*
(Sumber : Adi Nugroho ; 2009 : 74)

II.7.6. *Sequence Diagram*

Menurut Imam Adi Nugaha (2012) Notasi atau simbol pembentuk *activity diagram* dapat dilihat pada tabel II.4. Berikut ini :

Tabel II.4. Simbol-simbol Sequence Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		*Actor*	actor juga dapat berkomunikasi dengan object, maka actor juga dapat diurutkan sebagai kolom. Simbol Actor sama dengan simbol pada Actor Use Case Diagram.
2		*Message*	message digambarkan dengan anak panah horizontal antara

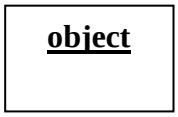
			Activation.Message mengindikasikan komunikasi antara object-object. (*independent*).
3		*Lifeline*	Lifeline mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk Lifeline hádala garis putus-putus vertikal yang ditarik dari sebuah obyek.
4		*object*	instance dari sebuah class dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah class (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma.
5		*activation*	activation dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah lifeline. Activation mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.

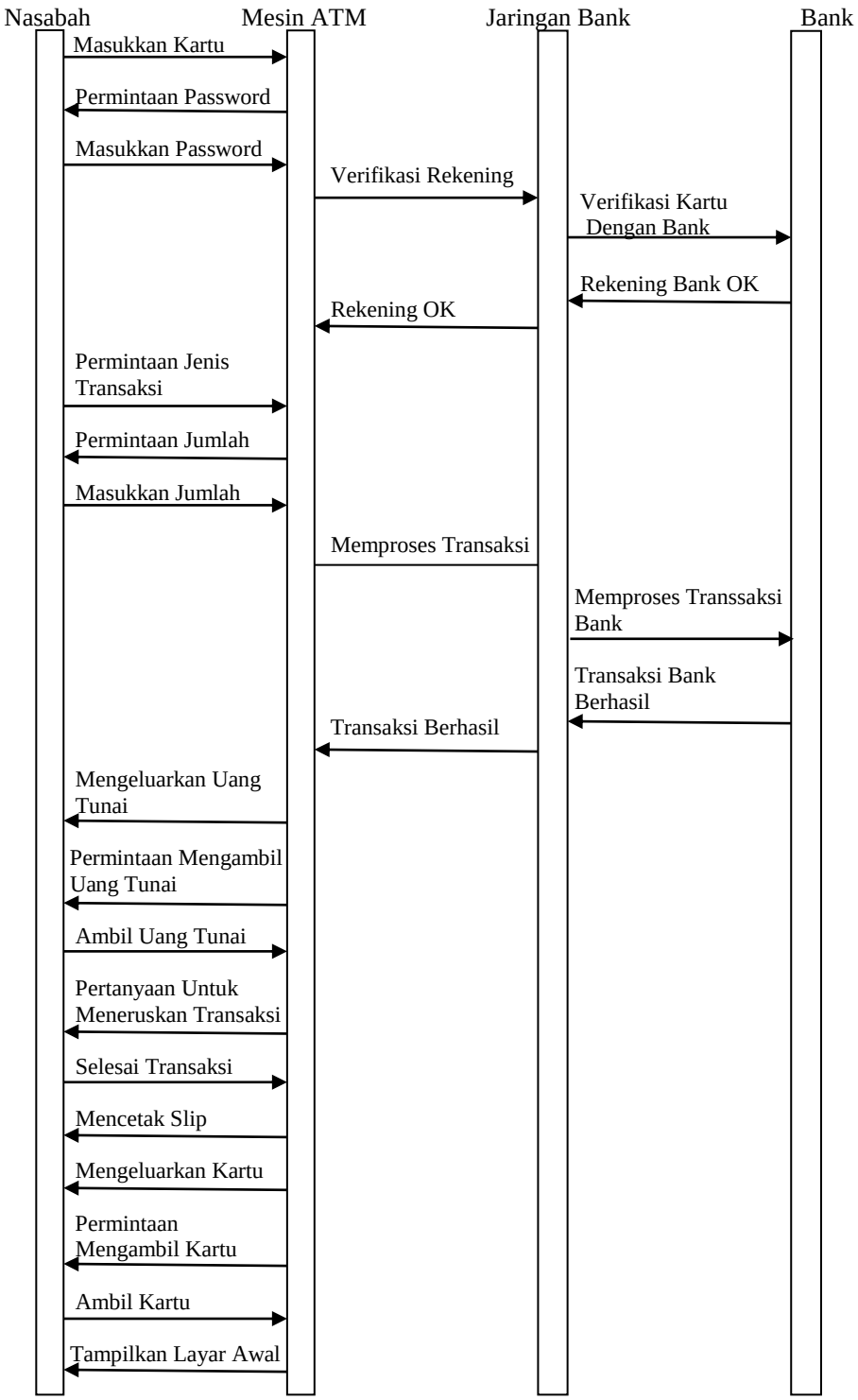
Diagram sekuensial atau *sequence diagram* digunakan untuk menunjukkan aliran fungsionalitas dalam *use case*. Diagram sekuensial adalah diagram yang disusun berdasarkan urutan waktu. Kita membaca diagram sekuensial dari atas ke bawah. Setiap diagram sekuensial mempresentasikan suatu aliran dari beberapa aliran di dalam *use case*.

Jadi dengan kata lain sekuensial diagram menunjukkan aliran fungsionalitas berdasarkan urutan waktu serta kejadian yang nantinya akan menetukan metode/fungsi atribut masing-masing. Dimana fungsi-fungsi tersebut akan diterapkan pada suatu kelas/objek.

Perhatikan gambar II.8. dimana terlihat pengelompokkan *event-event* serta fungsi masing-masing atribut tersebut. Di dalam diagram terlihat jelas bagaimana aliran suatu proses kejadian dimana seorang nasabah yang akan

melakukan transaksi dengan sebuah mesin ATM. Dari diagram tersebut kita mengetahui *event-event* yang terjadi, seperti : Nasabah memasukkan kartu ATM, Mesin ATM merespon dengan meminta *password* atau PIN, dan selanjutnya.

Kita dapat melihat setiap fungsi atribut dan *event-event* apa saja yang terjadi. Sehingga melalui diagram sekuensial ini kita dapat merancang suatu program aplikasi yang baik, sehingga dalam menghadapi sebuah kasus yang benar-benar kompleks diagram sekuensial ini sangat membantu. Gambar contoh *sequence diagram* dapat dilihat pada gambar II. 8 berikut ini :

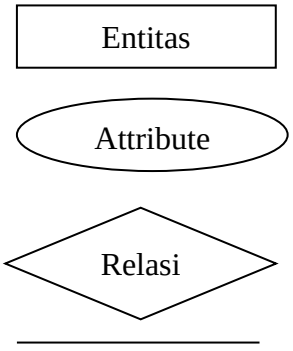


Gambar II.8. Contoh *Sequence Diagram*
(Adi Nugroho ; 2009 : 36)

II.8. *Database* Dan *ERD*

Menurut Yuniar Supardi (2008 : 9) *Desain database* merupakan pekerjaan yang penting dalam pembuatan atau pengembangan sistem, karena *desain database* akan mendapatkan susunan data atau *table* yang *efektif* dan *efisien*. Alat *desain database* yang popular ada dua, yaitu : *ERD* (*Entity Relationship Diagram*) dan *Normalisasi*. Jika memakai *Normalisasi* harus mendapatkan Data Dasar (Dokumen Dasar), sedangkan *ERD* tidak perlu. Dalam desain *ERD* terbagi dua tahapan yaitu: *Preliminary Desain* (*Disain* Awal) dan *Final Design* (*Disain* Akhir). Tetapi disain Akhir dari *ERD* juga berisi *Normalisasi*.

Menurut Rosa A.S dan M. Shalahuddin (2011 : 60), Struktur *logis* (skema *database* dapat ditunjukan secara *grafis* dengan *ERD* yang dibentuk dari komponen-komponen berikut ini :



Persegi panjang mewakili kumpulan *entitas*

Elips mewakili *attribute*

Belah ketupat mewakili *relasi*

Garis menghubungkan *atribut* dengan kumpulan entitas dengan *relasi*.

II.9. Data *Dictionary* (kamus data)

Dalam suatu rancangan *database*, data *dictionary* digunakan untuk menjelaskan atau mendeskripsikan kolom-kolom pada masing-masing tabel yang akan dibuat ke dalam *database*. Deskripsi kolom yang dimaksud di sini meliputi

tipe data, lebar karakter atau *digit*, serta keterangan tentang kunci *relasi* (Budi Raharjo : 2011 ; 59).

II.10. *Normalisasi*

Menurut Yuniar Supardi (2008 : 10) tahapan *normalisasi* terdiri dari beberapa bentuk, yaitu:

1. Bentuk Tak Normal (*UNF* / *Un Normal Form*).
2. Bentuk Normal Pertama (*1 NF* / *First Normal Form*).

   Bentuk Normal pertama memiliki ciri: Data berbentuk *file-file* (*file* datar), *record* disusun sesuai kedatangan, masih mungkin terjadi penyimpangan data (*anomali* data). *Anomali* data dapat berupa *insert anomali, delete anomali, update, anomali, dan redundancy data* (data *duplikat*).
3. Bentuk Normal Kedua (*2 NF* / *Second Normal Form*).

   Bentuk Normal kedua memiliki ciri; Tidak terjadi *anomali* data, setiap *field*/atribut bukan kunci harus tergantung fungsi (*Functional Depedency*) terhadap *field*/*atribut* kunci, masih mungkin terjadi *transitive dependency* (*field* bukan kunci tergantung pada *field* bukan kunci dalam satu *table*). Model objek mencapai bentuk normal kedua, sehingga penulis mendesain mulai bentuk normal ketiga dan bentuk *normal boyce codd*. Sedangkan untuk bentuk tak normal sudah dari dokumen dasar berupa Faktur, Nota, dan laporan *Stock of Name*.

4. Bentuk Normal Ketiga (*3 NF* / *Third Normal Form*).

   Table yang memenuhi Bentuk Normal Ketiga harus tidak terdapat *Transitive Depedency*. Bentuk normal ketiga dari sistem *inventory* :

5. Bentuk Normal *Boyce Codd* (*BCNF* / *Boyce Codd Normal Form*).

   Karena tak ada *field* bukan kunci tergantung secara *parsial* (bagian) kunci dalam satu tabel, maka bentuk normal ketiga juga merupakan bentuk *BCNF*.

6. Normal yang lebih tinggi.

   Setelah *3NF*, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Bentuk *Normal Boyce-Code (BCNF)* adalah versi *3NF* yang lebih teliti dan berhubungan dengan tabel *relasional* yang mempunyai banyak kunci *kandidat*, kunci *kandidat* gabungan, dan kunci *kandidat* yang saling tumpang tindih.