

BAB II

LANDASAN TEORI

II.1. Konsep Dasar Informasi

Sistem informasi dalam organisasi dapat dikatakan sebagai sistem yang menyediakan informasi bagi semua tingkatan dalam organisasi tersebut kapan saja diperlukan. Sistem ini menyimpan, mengambil, mengubah, mengolah dan mengkomunikasikan informasi yang diterima dengan menggunakan sistem informasi atau peralatan sistem lainnya. Sistem informasi akan bekerja pada fungsi-fungsi organisasi sesuai dengan derajat kebutuhan informasi pada tingkatan manajemen dalam organisasi. Karakteristik dan tipe-tipe informasi akan terkait dengan kebutuhan informasi pada tingkatan-tingkatan manajemen dalam organisasi. Sistem informasi berfungsi mentransformasikan data menjadi informasi yang bernilai bagi tiap tingkatan manajemen. Sistem informasi mendukung pembuatan keputusan sesuai tahap-tahap pembuatan keputusan manajemen dalam organisasi dan informasi dapat didefinisikan sebagai :

1. Kualitas Informasi

Istilah kualitas informasi (*quality of information*) terkadang juga dipakai untuk menyatakan informasi yang baik. Kualitas informasi sering kali diukur berdasarkan :

a). Relevansi

Relevan berarti bahwa informasi benar-benar memberikan manfaat bagi pemakai.

b). Ketepatan Waktu

Ketepatan waktu berarti informasi tersebut tidak usang / kadaluarsa ketika sampai ke penerima, sehingga masih ada waktu untuk menggunakan informasi tersebut sebagai bahan pengambilan keputusan.

c). Keakurasian

Akurasi berarti menyatakan derajat kebenaran terhadap informasi dan menentukan kehandalan atau reliabilitas informasi.

2. Nilai Informasi

Nilai informasi (*value of information*) ditentukan oleh dua hal, yaitu manfaat dan biaya untuk mendapatkannya. Suatu informasi dianggap bernilai kalau manfaatnya lebih efektif dibandingkan dengan biaya untuk mendapatkannya. Akan tetapi perlu diperhatikan bahwa informasi yang digunakan didalam suatu system informasi umumnya digunakan untuk beberapa kegunaan. Sebagian informasi tidak dapat ditaksir keuntungannya dengan suatu nilai tetapi dapat ditaksir dari nilai keefektifannya.

3. Teknologi Informasi

Information Teknologi (IT) adalah sebuah terminologi kontemporer yang mendeskripsikan kombinasi antara teknologi komputer (*hardware* dan *software*) dengan teknologi komunikasi (data, *image*, dan jaringan suara).

Peranan Teknologi Informasi :

1. Teknologi informasi menggantikan peran manusia. Dalam hal ini, teknologi informasi melakukan otomasi terhadap suatu tugas atau proses.
2. Teknologi memperkuat peran manusia, yakni dengan menyajikan informasi terhadap suatu tugas atau proses.

Teknologi informasi berperan dalam restrukturisasi terhadap peran manusia. Dalam hal ini, teknologi berperan dalam melakukan perubahan - perubahan terhadap sekumpulan tugas atau proses. (Dikatar : 2011)

II.2. Sistem Pendukung Keputusan

Menurut Hilyah M. (2012) Konsep Sistem Pendukung Keputusan pertama kali diperkenalkan pada awal tahun 1970-an oleh Michael S. Scott Morton dengan istilah *Management Decision System* (Sprague, 1982). Konsep pendukung keputusan ditandai dengan sistem interaktif berbasis komputer yang membantu pengambil keputusan memanfaatkan data dan model untuk menyelesaikan masalah-masalah yang tidak terstruktur. Pada dasarnya SPK dirancang untuk mendukung seluruh tahap pengambilan keputusan mulai dari mengidentifikasi masalah, memilih data yang relevan, menentukan pendekatan yang digunakan dalam prosen pengambilan keputusan, sampai mengevaluasi pemilihan alternatif.

Menurut Iskandar Z. Nasibu Vol. 2 No. 5 (2009) Sistem Pendukung Keputusan (*Decision Support System/DSS*) adalah sebuah sistem yang memberikan dukungan kepada seorang manajer, atau kepada sekelompok manajer yang relatif yang bekerja sebagai tim pemecah masalah, dalam memecahkan masalah semi terstruktur dengan memberikan informasi atau saran mengenai

keputusan tertentu. Informasi tersebut dapat diberikan dalam bentuk laporan berkala, laporan khusus, maupun output dalam model matematis. Model tersebut juga mempunyai kemampuan untuk memberikan saran dalam tingkat yang bervariasi.

Menurut Ahmad Khaidir (2014), Pengambilan keputusan merupakan proses pemilihan alternative tindakan untuk mencapai tujuan atau sasaran tertentu. Pengambilan keputusan dilakukan dengan pendekatan sistematis terhadap permasalahan melalui proses pengumpulan data menjadi informasi serta ditambah dengan faktor - faktor yang perlu dipertimbangkan dalam pengambilan keputusan.

Menurut Keen dan Scoot Morton : “Sistem Pendukung Keputusan merupakan penggabungan sumber-sumber kecerdasan individu dengan kemampuan komponen untuk memperbaiki kualitas keputusan. Sistem Pendukung Keputusan juga merupakan sistem informasi berbasis komputer untuk manajemen pengambilan keputusan yang menangani masalah-masalah semi struktur “.

Dengan pengertian diatas dapat dijelaskan bahwa sistem pendukung keputusan bukan merupakan alat pengambilan keputusan, melainkan merupakan sistem yang membantu pengambil keputusan dengan melengkapi mereka dengan informasi dari data yang telah diolah dengan relevan dan diperlukan untuk membuat keputusan tentang suatu masalah dengan lebih cepat dan akurat. Sehingga sistem ini tidak dimaksudkan untuk menggantikan pengambilan keputusan dalam proses pembuatan keputusan.

Alter (2002:38) mendefinisikan Sistem pendukung keputusan atau *Decision Support Systems* (DSS) adalah sistem informasi interkatif yang

menyediakan informasi, pemodelan, dan pemanipulasian data yang digunakan untuk membantu pengambilan keputusan pada situasi yang semiterstruktur dan situasi yang tidak terstruktur dimana tak seorang pun tahu secara pasti bagaimana keputusan seharusnya dibuat. Konsep DSS dikemukaakan pertama kali oleh Scoot Morton pada tahun 1971 (Turban, McLean, dan Wetherbe, 1999). Beliau mendefinisikan cikal bakal DSS tersebut sebagai : “*system* berbasis computer yang interaktif, yang membantu pengambil keputusan dengan menggunakan data dan model untuk memecahkan persoalan-persoalan tak terstruktur”

II.2.1. Tahap-Tahap Pengambilan Keputusan

Dalam Hilyah M. (2012), Menurut Herbert A. Simon (Kadarsyah Suryadi dan Ali Ramdhani, 2002, 15-16) model yang menggambarkan proses pengambilan keputusan. Proses Pengambilan Keputusan melibatkan 4 tahapan, yaitu:

1. Tahap *Intelligence*

Dalam tahap merupakan proses penelusuran dan pendeteksian dari lingkup problematika serta proses pengenalan masalah. Data masukan diperoleh, diproses, dan diuji dalam rangka mengidentifikasi masalah.

2. Tahap *Design*

Dalam tahap ini merupakan proses menemukan, mengembangkan dan menganalisis alternatif tindakan yang bisa dilakukan.

3. Tahap *Choice*

Dalam tahap ini dilakukan proses pemilihan diantara berbagai alternatif tindakan yang mungkin dijalankan.

4. Tahap *Implementation*

Dalam tahap ini pengambil keputusan menjalankan rangkaian aksi pemecahan yang dipilih di tahap *choice*. Implementasi yang sukses ditandai dengan terjawabnya masalah yang dihadapi, sementara kegagalan ditandai dengan tetap adanya masalah yang sedang dicoba untuk diatasi.

II.2.2. Komponen Sistem Pendukung Keputusan

Dalam Nila Susanti dan Sri Winiarti (2013) Sistem Pendukung Keputusan terdiri atas empat komponen utama, yaitu:

1. Subsistem manajemen data berfungsi sebagai memasukkan suatu *database* yang berisi data yang relevan untuk situasi dan dikelola oleh perangkat lunak yang disebut yang disebut sistem manajemen *database* (DBMS). *Knowledge Base* berisi semua fakta, ide, hubungan dan interaksi suatu domain tertentu.
2. Subsistem manajemen basis pengetahuan bertugas untuk mendukung semua subsistem lain atau bertindak sebagai suatu komponen independen. Ia memberikan intelegensi untuk memperbesar pengetahuan pengambil keputusan.
3. Subsistem manajemen model merupakan paket perangkat lunak yang memasukkan model keuangan statistik, ilmu manajemen atau model kuantitatif lainnya yang memberikan kapabilitas analitik dan manajemen perangkat lunak yang tepat.

4. Subsistem antarmuka pengguna (dialog) untuk mengimplementasikan sistem kedalam program aplikasi sehingga pengguna atau pemakai dapat berkomunikasi dengan sistem yang dirancang.

Dalam Heny Pratiwi (2014), Menurut Sudiyantoro (2005), komponen dalam sistem pendukung keputusan meliputi 8 bagian yaitu:

1. Perangkat Keras

Perangkat keras ini akan terhubung dengan komputer lain menggunakan sistem jaringan, sehingga memudahkan dalam pengambilan data pada organisasi tersebut.

2. Perangkat Lunak

Perangkat lunak sistem pendukung keputusan sering disebut juga dengan *DSS Generator*, berisi modul-modul untuk database, model dan *dialogue management*.

3. Sumber Data

Database sistem pendukung keputusan berisi data dan informasi yang diambil dari data organisasi, eksternal dan data manajer secara individu.

4. Sumber Model

Model ini berisi kumpulan model matematika dan teknik analisis yang disimpan dalam program dan berkas yang berbeda-beda. Komponen dari model ini dapat di kombinasikan dengan perangkat lunak tertentu untuk mendukung sebuah keputusan yang akan diambil.

5. Sumber Daya Manusia

Sistem pendukung keputusan dapat digunakan oleh para manajer dan staf khusus

untuk membuat keputusan ini juga dapat dikembangkan oleh penggunanya sesuai dengan keperluan para pengguna tersebut.

6. Model Sistem Pendukung Keputusan

Model merupakan komponen yang penting dalam SPK. Model memiliki pengertian yang berarti memisahkan dari dunia nyata dengan melukiskan komponen utama dan menghubungkannya dengan sistem dan kejadian lainnya.

7. Lembar Kerja Elektronik

Lembar kerja elektronik memudahkan pengguna membuat model dengan cara mengisi data dan menghubungkannya sesuai dengan format yang telah disediakan. Pengguna dapat melakukan beberapa perubahan dan mengevaluasi secara tampilan grafik.

8. Sistem Pendukung Keputusan Kelompok

Merupakan suatu sistem berbasis komputer yang mendukung kelompok-kelompok orang yang terlibat dalam suatu tugas atau tujuan bersama dan menyediakan tampilan antarmuka pada satu lingkungan.

II.2.3. Kriteria Sistem Pendukung Keputusan

Dalam Dita Monita (2013) Sistem pendukung keputusan dirancang secara khusus untuk mendukung seseorang yang harus mengambil keputusan-keputusan tertentu [1]. Berikut ini beberapa kriteria sistem pendukung keputusan, yaitu:

1. Interaktif

Sistem pendukung keputusan memiliki *user interface* yang komunikatif sehingga pemakai dapat melakukan akses secara cepat ke data dan memperoleh informasi yang dibutuhkan.

2. Fleksibel

Sistem pendukung keputusan memiliki sebanyak mungkin variable masukan, kemampuan untuk mengolah dan memberikan keluaran yang menyajikan alternatif-alternatif keputusan kepada pemakai.

3. Data Kualitas

Sistem pendukung keputusan memiliki kemampuan untuk menerima data kualitas yang dikuantitaskan yang sifatnya subyektif dari pemakainya, sebagai data masukan untuk pengolahan data. Misalnya terhadap kecantikan yang bersifat kualitas, dapat dikuantitaskan dengan pemberian bobot nilai seperti 75 atau 90.

4. Prosedur Pakar

Sistem pendukung keputusan mengandung suatu prosedur yang dirancang berdasarkan rumusan formal atau juga berupa prosedur kepakaran seseorang atau kelompok dalam menyelesaikan suatu bidang masalah dengan fenomena tertentu.

II.2.4. Struktur Keputusan dalam Sistem Pendukung Keputusan

Dalam Jurnal Heny Pratiwi (2014), Menurut Kusri (2007), keputusan yang diambil untuk menyelesaikan suatu masalah dilihat dari struktur masalahnya terbagi menjadi tiga yaitu :

1. Keputusan terstruktur (*Structured Decision*)

Keputusan yang dilakukan secara berulang-ulang dan bersifat rutin. Prosedur pengambilan keputusan sangat jelas. Keputusan tersebut dilakukan pada manajemen tingkat bawah (operasional). Misalnya, keputusan pemesanan barang dan keputusan penagihan piutang.

2. Keputusan semi terstruktur (*Semi Structured Decision*)

Keputusan yang memiliki dua sifat. Sebagian keputusan bisa ditangani oleh komputer sedangkan yang lain tetap harus dilakukan oleh pengambil keputusan. Produser dalam pengambilan keputusan tersebut secara garis besar sudah ada, tetapi beberapa hal yang masih memerlukan kebijakan dari pengambil keputusan. Prosedur dalam pengambilan keputusan tersebut secara garis besar sudah ada, tetapi beberapa hal yang masih memerlukan kebijakan dari pengambilan keputusan. Biasanya keputusan semacam ini diambil oleh manajer tingkat menengah (taktikal). Contoh keputusan jenis ini adalah evaluasi kredit, penjadwalan produksi dan pengendalian persediaan.

3. Keputusan Tak Terstruktur (*Unstructured Decision*)

Keputusan yang penanganannya rumit karena tidak selalu terjadi. Keputusan tersebut menurut pengalaman dan berbagai sumber yang bersifat eksternal. Keputusan tersebut umumnya terjadi pada manajemen tingkat atas (strategis). Contohnya adalah keputusan untuk pengembangan teknologi baru, keputusan bergabung dengan perusahaan lain dan perekrutan eksekutif.

II.3. Metode *TOPSIS* (*Technique For Others Reference by Similarity to Ideal Solution*)

Metode *TOPSIS* adalah salah satu metode pengambilan keputusan multikriteria yang pertama kali diperkenalkan oleh Yoon dan Hwang pada tahun 1981. Metode ini merupakan salah satu metode yang banyak digunakan untuk menyelesaikan pengambilan keputusan secara praktis. *TOPSIS* memiliki konsep dimana alternatif yang terpilih merupakan alternatif terbaik yang memiliki jarak terpendek dari solusi ideal positif dan jarak terjauh dari solusi ideal negatif [4]. Semakin banyaknya faktor yang harus dipertimbangkan dalam proses pengambilan keputusan, maka semakin relatif sulit juga untuk mengambil keputusan terhadap suatu permasalahan. Apalagi jika upaya pengambilan keputusan dari suatu permasalahan tertentu, selain mempertimbangkan berbagai faktor/kriteria yang beragam, juga melibatkan beberapa orang pengambil keputusan.

Permasalahan yang demikian dikenal dengan permasalahan multiple criteria decision making (MCDM). Dengan kata lain, MCDM juga dapat disebut sebagai suatu pengambilan keputusan untuk memilih alternatif terbaik dari sejumlah alternatif berdasarkan beberapa kriteria tertentu. Metode *TOPSIS* digunakan sebagai suatu upaya untuk menyelesaikan permasalahan multiple criteria decision making. Hal ini disebabkan konsepnya sederhana dan mudah dipahami, komputasinya efisien dan memiliki kemampuan untuk

mengukur kinerja relatif dari alternatif-alternatif keputusan. (Sumber : undip.ac.id)

Contohnya dapat dilihat pada tabel berikut :

- **Tabel II.1 Tabel Nilai Keputusan**

	HARGA TANAH	UMR	PAJAK	SUPLAI AIR	SDM
SEMARANG	2	2	2	4	5
KUDUS	4	4	3	3	2
KENDAL	2	3	2	4	4
JEPARA	3	4	5	5	3

(Sumber : Suharyaningsih, Juni : 2014)

- **Tabel II.2 Tabel Bobot Kriteria**

HARGA TANAH	UMR	PAJAK	SUPLAI AIR	SDM
4	5	3	5	2

(Sumber : Suharyaningsih, Juni : 2014)

II.3.1.Konsep Dasar Penggunaan Metode TOPSIS

Langkah-langkah metode TOPSIS :

1. Membangun *Normalized Decision Matrix*

Elemen r_{ij} hasil dari normalisasi decision matrix R dengan metode Euclidean length of a vector adalah:

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^m x_{ij}^2}} \dots\dots\dots(1)$$

(Sumber : Suharyaningsih, Juni : 2014)

2. Menentukan Solusi Ideal dan Solusi Ideal Negatif.

Solusi ideal dinotasikan A*, sedangkan solusi ideal negatif dinotasikan A- :

$$A^* = \{(\max V_{ij} | j \in J), (\min V_{ij} | j \in J'), \dots\dots\dots(2)$$

$$I = 1,2,3, \dots, m \} = \{V_1^*, V_2^*, \dots, V_n^*\} \dots\dots\dots(3)$$

$$A^- = \{(\min V_{ij} | j \in J), (\max V_{ij} | j \in J'), \dots\dots\dots(4)$$

$$i = 1,2,3, \dots, m \} = \{V_1, V_2, \dots, V_n\} \dots\dots\dots(5)$$

J = {j = 1,2,3n dan j merupakan *benefit criteria*}

J' {j = 1,2,3n dan j merupakan *cost criteria*}

(Sumber : Suharyaningsih, Juni : 2014)

3. Menghitung Separasi

Siparasi adalah jarak (dalam pandangan Euclidean) alternatif dari solusi ideal

didefinisikan sebagai :

$$S_i^* = \sqrt{\sum_{j=1}^n (V_{ij} - V_j^*)^2}, \text{ dengan } i = 1,2,3, \dots, m \dots\dots\dots(6)$$

$$S_i^* = \sqrt{\sum_{j=1}^n (V_{ij} - V_j^-)^2}, \text{ dengan } i = 1,2,3, \dots, m \dots\dots\dots(7)$$

(Sumber : Suharyaningsih, Juni : 2014)

Dan jarak terhadap solusi negatif-ideal didefinisikan sebagai:

4. Menghitung kedekatan relatif terhadap solusi ideal

$$C_{i^*} = \frac{S}{S_{i^*} + S_i} \quad \text{dengan } 0 < C_{i^*} < 1 \text{ dan } i = 1, 2, 3, \dots, m \quad \dots\dots\dots(8)$$

(Sumber : Suharyaningsih, Juni : 2014)

II.4. Siswa/i

Menurut pasal 1 ayat 4 UU RI No. 20 tahun 2013, mengenai sistem pendidikan nasional diana peserta didik atau siswa adalah anggota masyarakat yang berusaha mengembangkan diri mereka melalui proses pendidikan pada jalur dan jenjang dan jenis pendidikan tertentu. UUD pasal 1 ayat 4 No. 20 : 2013)

II.5. Pengertian MySQL

MySQL *database server* adalah RDBMS (*Relasional Database Management Sistem*) sekumpulan data yang tersusun dengan aturan tertentu dalam bentuk tabel. Adapun secara fungsi, database merupakan suatu tempat yang dipergunakan untuk menyimpan sekumpulan data dalam format tertentu. Saat ini

terdapat puluhan jenis database yang secara aktif dikembangkan dan dipergunakan dalam aplikasi.

Awalnya database MySQL ini dikembangkan oleh komunitas dengan diawasi dan dikembangkan oleh perusahaan yang dinamakan MySQL.AB. Setelah perusahaan tersebut diakuisisi oleh SUN Microsystem, secara otomatis lisensi dari MySQL juga berpindah ke perusahaan pengembang bahasa pemrograman Java tersebut. Belakangan SUN Microsystem diambil alih oleh Oracle. Perpindahan manajemen inilah yang semakin mengukuhkan database MySQL, baik bagi pengguna komunitas maupun pengguna enterprise.

(PAS : Microsoft Visual Basic 2010 dan MySQL untuk Aplikasi Point of Sales)

II.6. *Unified Modelling Language (UML)*

UML singkatan dari Unified Modeling Language yang berarti bahasa pemodelan standar. Chonoles mengatakan sebagai bahasa, berarti UML memiliki sintaks dan semantic. Ketika kita membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan lainnya harus mengikuti standar yang ada. UML bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. (Prabowo et al; 2011 : 6)

UML diaplikasikan untuk maksud tertentu, biasanya antara lain:

1. Merancang perangkat lunak
2. Sarana komunikasi antara perangkat lunak dengan proses bisnis.

3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

Blok pembangunan utama UML adalah diagram. Beberapa diagram ada yang rinci (jenis *timing diagram*) dan lainnya ada yang bersifat umum (misalnya diagram kelas). Para pengembangan sistem berorientasi objek menggunakan bahasa model untuk menggambar, 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 mensupport para pengembangan sistem saat ini. (Prabowo Pudjo Widodo, Herlawati; 2011 : 6-7)

II.6.1. Diagram-Diagram UML

Beberapa literatur menyebutkan bahwa UML menyediakan Sembilan jenis diagram, yang lain menyebutkan delapan karena ada beberapa yang digabung, misalnya diagram komunikasi, diagram urutan, dan diagram pewaktuan digabung menjadi diagram interaksi. Namun demikian model-model itu dapat dikelompokkan berdasarkan sifatnya yaitu statis atau dinamis. Jenis diagram itu antara lain:

1. Diagram Kelas, bersifat statis. Diagram ini memperlihatkan himpunan kelas-kelas, antarmuka-antarmuka, kolaborasi-kolaborasi, serta relasi-relasi. Diagram ini umum dijumpai pada pemodela sistem berorientasi objek . Meskipun bersifat statis, sering pula diagram kelas mmemuat kelas-kelas aktif.

2. Diagram paket (*Package Diagram*), bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas, merupakan bagian dari diagram komponen.
3. Diagram *use case*, bersifat statis. Diagram ini memperlihatkan himpunan *use case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram interaksi dan *sequence* (urutan), bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam suatu waktu tertentu.
5. Diagram komunikasi (*Communication Diagram*), bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi UML 14 yang menekankan organisasi structural dari objek-objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram*), bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*state*), transisi, kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi, dan terutama penting pada pemodelan sistem-sistem yang reaktif.
7. Diagram aktivitas (*Activity Diagram*), bersifat dinamis. Diagram aktivitas adalah tipe khusus dari diagram status yang memperlihatkan aliran dari suatu aktivitas lainnya dalam suatu sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi sistem dan member tekanan pada aliran kendali antar objek.
8. Diagram komponen (*Component Diagram*), bersifat statis. Diagram komponen ini memperlihatkan organisasi serta keberuntungan sistem / perangkat lunak pada

komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan dengan diagram kelas dimana komponen secara tipikal dipetakan ke dalam satu atau lebih kelas-kelas, antarmuka-antramuka serta kolaborasi-kolaborasi.

9. Diagram *Deployment (Deployment Diagram)*, bersifat statis. Diagram ini memperlihatkan konfigurasi saat aplikasi dijalankan (*run-time*). Memuat simpul-simpul beserta komponen-komponen yang ada di dalamnya. Diagram *deployment* berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*Distributed Computing*)

Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada UML dimungkinkan kita menggunakan diagram-diagram lainnya (misalnya *Data Flow Diagram, Entity Relationship Diagram* dan sebagainya). (Prabowo Pudjo Widodo, Herlawati; 2011 : 10-12)

- a. Diagram *Use Case (Use Case Diagram)*

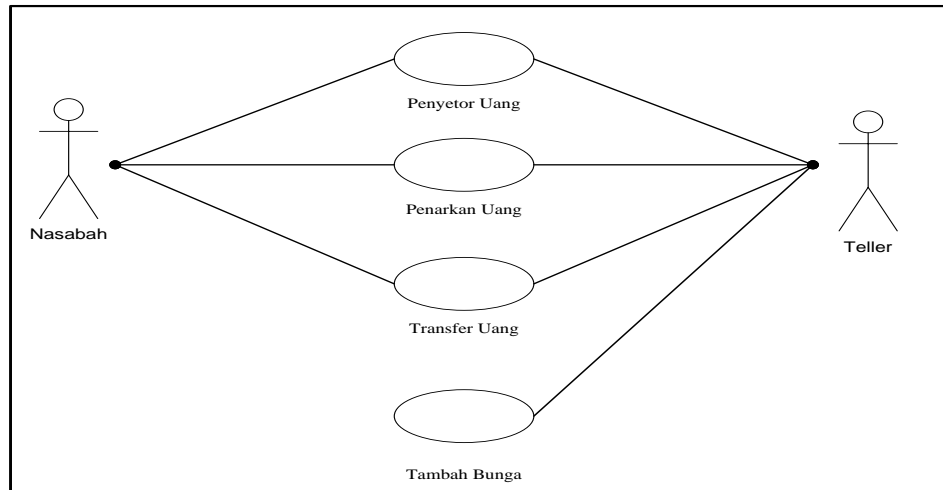
Use Case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Menurut Pooley (2005:15) mengatakan bahwa model *use case* dapat dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

Komponen pembentuk diagram *use case* adalah:

1. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.
2. *Use Case*, aktifitas/sarana yang disiapkan oleh bisnis / sistem.

3. Hubungan (*Link*), actor mana saja yang terlibat dalam *use case* ini.

Gambar dibawah ini merupakan salah satu contoh bentuk diagram *use case* yaitu :

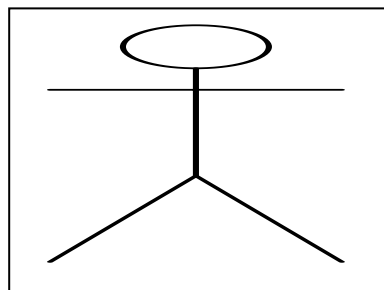


Gambar II.1. Diagram Use Case

Sumber : Prabowo Pudjo Widodo, Herlawati (2011 : 17)

a) Aktor

Menurut Chonoles (2003 : 17) menyarankan sebelum membuat *use case* dan menentukan aktornya, agar mengidentifikasi siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat biasanya dinamakan *stakeholder*.

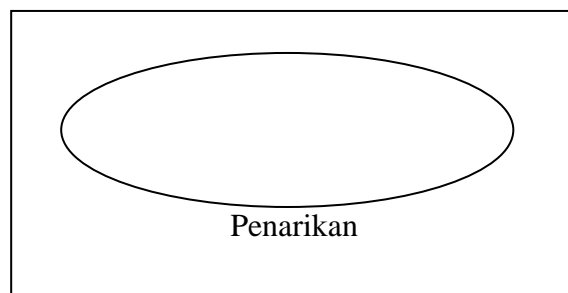


Gambar II.2 Diagram Aktor

(Sumber : Prabowo Pudjo Widodo, Herlawati 2011 : 17)

b) Use Case

Menurut Pilone (2005 : 21) use case menggambarkan fungsi tertentu dalam suatu sistem berupa komponen, kejadian atau kelas. Sedangkan menurut Whitten (2004 : 258) mengartikan use case sebagai urutan langkah-langkah yang secara tindakan saling terkait (scenario), baik terotomatisasi maupun secara manual, untuk tujuan melengkapi suatu tugas bisnis tunggal. Use case digambarkan dalam bentuk ellips/oval pada gambar II.3 sebagai berikut :



Gambar II.3 Simbol Use Case

(Sumber : Prabowo Pudjo Widodo, Herlawati 2011 : 22)

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu (Chonoles, 2003: 22) menawarkan cara untuk menghasilkan use case yang baik, yakni:

1) Pilihlah nama yang baik

Use case adalah sebuah *behavior* (perilaku), jadi seharusnya dalam frase kata kerja. Untuk membuat namanya lebih detil, tambahkan kata benda yang

mengidentifikasi dampak aksinya terhadap suatu kelas objek. Oleh karena itu diagram *use case* seharusnya berhubungan dengan diagram kelas.

2) Ilustrasikan perilaku dengan lengkap

Use case dimulai dari inisiasi oleh aktor primer dan berakhir pada actor dan menghasilkan tujuan. Jangan membuat *use case* kecuali Anda mengetahui tujuannya. Sebagai contoh, memilih jenis tempat tidur (*king size*, *queen size* atau *dobel*) saat tamu memesan tidak dapat dijadikan *use case* karena merupakan bagian dari *use case* pemesanan kamar dan tidak dapat berdiri sendiri (tidak mungkin tamu memesan kamar tidur jenis *king* tapi tidak memesan kamar hotel).

3) Identifikasi perilaku dengan lengkap

Untuk mencapai tujuan dan menghasilkan nilai tertentu dari aktor, *use case* harus lengkap. Ketika member nama pada *use case*, pilihlah frasa kata kerja yang implikasinya hingga selesai. Misalnya gunakan frasa *reserve a room* (pemesanan kamar) dan jangan *reserving a room* (memesan kamar) karena memesan kamar menggambarkan perilaku yang belum selesai.

4) Menyediakan *Use case* lawan (*inverse*)

Kita biasanya membutuhkan *use case* yang membatalkan tujuan misalnya pada *use case* pemesanan kamar, dibutuhkan pada *use case* pembatalan pesanan kamar.

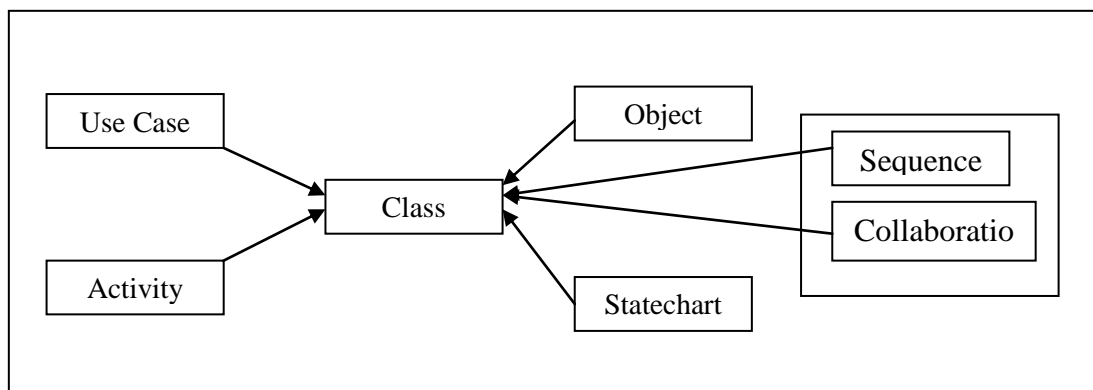
5) Batasi *Use case* hingga satu perilaku saja

Kadang kita cenderung membuat *use case* yang menghasilkan lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah *use case* kita hanya fokus pada satu hal. Misalnya, pengguna *use case* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda. Untuk menyediakan penjelasan detail

terhadap segala kemungkinan yang terjadi pada *use case*, apa yang terjadi dan apa respon sistem.

b. Diagram kelas (*Class Diagram*)

Diagram kelas adalah inti dari proses pemodelan objek, baik forward engineering maupun reverse engineering memanfaatkan diagram ini. Forward engineering adalah proses perubahan model menjadi kode program sedangkan reverse engineering sebaliknya merubah kode.



**Gambar II.4 Hubungan diagram kelas dengan diagram UML lainnya
(Sumber : Prabowo Pudjo Widodo, Herlawati 2011 : 38)**

C. Diagram Aktivitas (*Activity Diagram*)

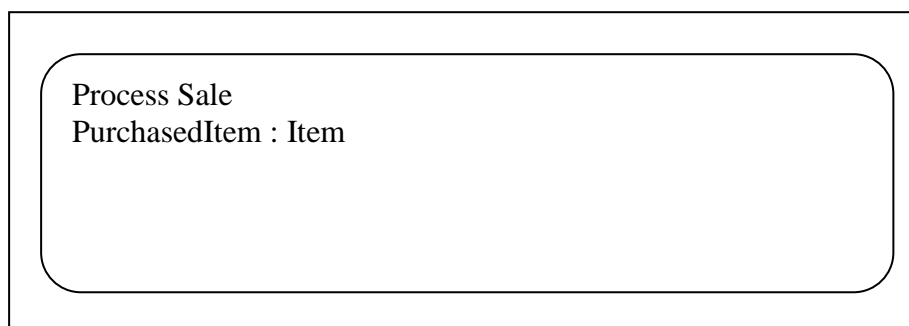
Diagram aktivitas lebih memfokuskan dari pada eksekusi dan alur sistem dari pada bagaimana sistem itu dirakit. Diagram ini tidak hanya memodelkan model bisnis juga. Diagram aktivitas menunjukkan aktivitas sistem dalam bentuk kumpulan aksi-aksi. Ketika digunakan dalam pemodelan software, diagram aktivitas mempresentasikan pemanggilan suatu fungsi tertentu misalnya *call*.

Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian di luar seperti pemesanan atau kejadian-kejadian internal misalnya proses penggajian tiap jumat sore.

Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi melakukan langkah sekali saja tidak boleh dipecah menjadi beberapa langkah lagi. Contoh aksi yaitu:

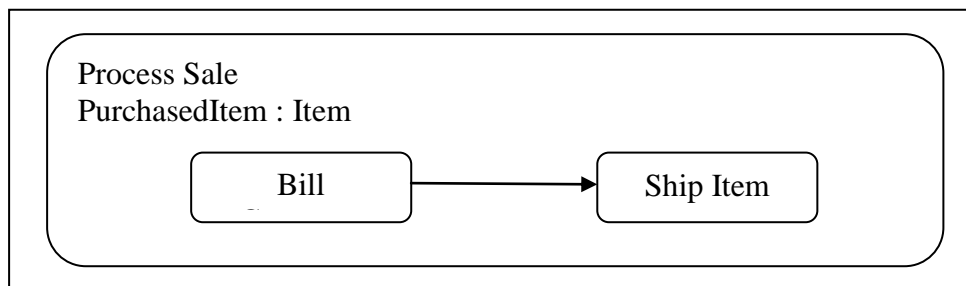
- 1) Fungsi matematika
- 2) Pemanggilan perilaku
- 3) Pemrosesan data

Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu classifier, classifier dikatakan konteks dari aktivitas. Aktivitas dapat mengakses atribut dan operasi classifier, tiap objek yang terhubung dan parameter-parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan untuk model proses bisnis, informasi itu biasanya disebut process-relevant data. Aktivitas diharapkan dapat digunakan ulang dalam suatu aplikasi, sedangkan aksi biasanya specific dan digunakan hanya untuk aktivitas tertentu. Aktivitas digambarkan dengan persegi panjang tumpu, namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



Gambar II.5. Aktivitas sederhana tanpa rincian
(Sumber : Prabowo Pudjo Widodo, Herlawati 2011 : 145)

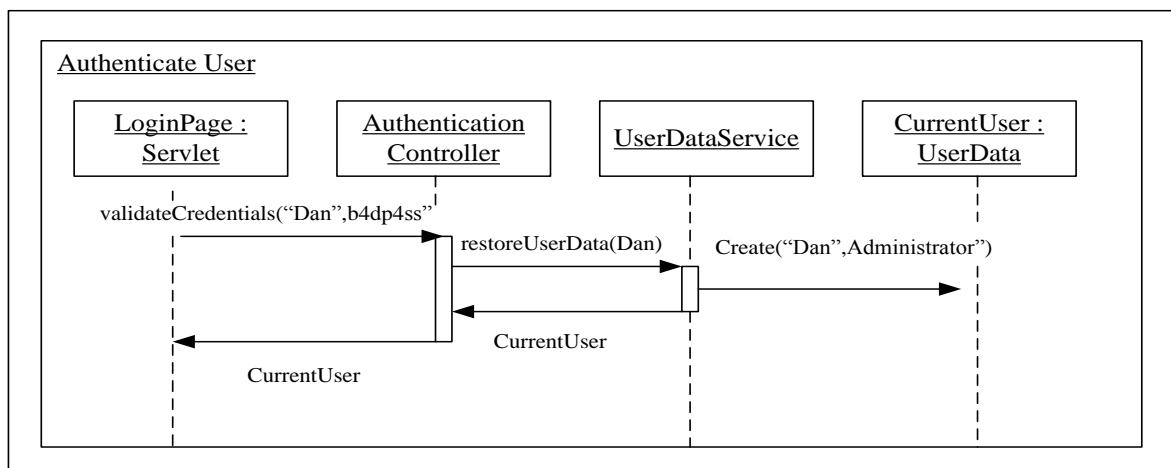
Detail aktivitas dapat dimasukkan di dalam kotak. Akan diperlihatkan dengan symbol yang sama dengan aktivitas dan namanya diletakkan didalam persegi panjang.



Gambar II.6. Contoh Diagram Komunikasi

(Sumber : Prabowo Pudjo Widodo, Herlawati 2011 : 174)

Menurut (Douglas, 2004 : 174) menyebutkan ada tiga diagram primer UML dalam memodelkan scenario interaksi, yaitu: diagram urutan (*sequence diagram*), diagram waktu (*timing diagram*) dan diagram komunikasi (*communication diagram*). Namun demikian (Pilone, 2005 : 174) menyatakan bahwa diagram yang paling banyak dipakai adalah diagram urutan. Gambar II.6. memperlihatkan contoh diagram urutan dengan notasi-notasinya yang akan dijelaskan nanti.



Gambar II.7. Contoh diagram urutan
(Sumber : Prabowo Pudjo Widodo, Herlawati 2011 : 175)

II.7. ERD (*Entity Relationship Diagram*)

Menurut Wahana Komputer; 2010:30. Pada dasarnya ERD (*Entity Relationship Diagram*) adalah sebuah diagram yang secara konseptual memetakan hubungan antar penyimpanan pada diagram DFD di atas. ERD ini digunakan untuk melakukan permodelan terhadap struktur data dan hubungannya. Penggunaan ERD ini dilakukan untuk mengurangi tingkat kerumitan penyusunan sebuah database yang baik.

Entity dapat berarti sebuah obyek yang dapat dibedakan dengan obyek lainnya. Obyek tersebut dapat memiliki komponen – komponen data (atribut atau field) yang membuatnya dapat dibedakan dari obyek yang lain. Dalam dunia database entity memiliki atribut yang menjelaskan karakteristik dari entity tersebut. Ada dua macam atribut yang dikenal dalam entity yaitu atribut yang berperan sebagai kunci primer dan atribut deskriptif. Hal ini berarti setiap entity memiliki himpunan yang diperlukan sebuah primary key untuk membedakan anggota – anggota dalam himpunan tersebut.

Atribut dapat memiliki sifat – sifat sebagai berikut :

- Atomic, atomik adalah sifat dari atribut yang menggambarkan bahwa atribut tersebut berisi nilai yang spesifik dan tidak dapat dipecah lagi. Contoh dari sifat

atomik adalah field status dari tabel karyawan yang hanya berisi menikah atau single.

- multivalued, sifat ini menandakan atribut ini bisa memiliki lebih dari satu nilai untuk tiap entity tertentu. Misalnya adalah field hobi, hobi dari tiap karyawan mungkin dan hampir pasti lebih dari satu. Misalnya karyawan A memiliki hobi : membaca, nonton TV dan bersepeda.
- Composite, atribut yang bersifat komposit adalah atribut yang nilainya adalah gabungan dari beberapa atribut yang bersifat atomik. Contohnya adalah atribut alamat yang dapat dipecah menjadi atribut atomik berupa alamat, kode pos, no telepon, dan kota.

Berikut ini adalah contoh atribut dan kunci dari entity mahasiswa, dosen, dan mata kuliah :

Entity	Atribut
Mahasiswa	<u>NIM</u>
	Nama Mahasiswa
	Tempat Tanggal Lahir
	Jenis Kelamin
	Agama
	Alamat
	Fakultas
	Jurusan
Dosen	<u>NIP</u>
	Nama dosen
	Jurusan
	Bidang keahlian
	Alamat
Mata Kuliah	<u>Kode matkul</u>
	Nama matkul
	SKS
	Semester

Keterangan : atribut yang bercetak tebal dan bergaris bawah merupakan primary key.

Contoh Kunci Entity

(Sumber : Andika Pratama, 31 Maret 2015 09:51)

ERD (*Entity Relationship Diagram*) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antar objek. Entitas adalah sesuatu atau objek dalam dunia nyata yang dapat dibedakan dari objek lain.

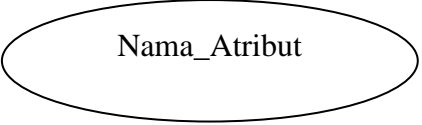


Sebagai tambahan, model ER menyajikan pula batasan dimana isi basis data harus menyesuaikan dengan batasan. Salah satu batasan yang penting adalah pemetaan kardinalitas (*mapping cardinalities*), yang menggambarkan jumlah entitas yang berhubungan dengan entitas lain melalui suatu relasi. (Janner Si ; 2010 : 60-61)

II.7.1. Simbol-simbol ERD (*Entity Relationship Diagram*)

Berikut Simbol **ERD** (*Entity Relationship Diagram*) :

Tabel II.3 Simbol-simbol ERD (*Entity Relationship Diagram*)

Simbol	Deskripsi
Entitas / <i>entity</i> <div style="text-align: center; border: 1px solid black; width: 150px; height: 40px; margin: 10px auto;"> Entitas </div>	Entitas merupakan data inti yang akan disimpan; bakal table pada basis data

<p>Atribut</p> 	<p>Fiel atau koilom data yang perlu disimpan dalam suatu entitas</p>
<p>Relasi</p> 	<p>Relasi yang menghubungkan antara entitas; relasi biasanya diawali dengan kata kerja</p>
<p>Asosiasi / <i>Association</i></p> 	<p>Penghubung antar relasi dan entitas dimana di kedua ujungnya memiliki <i>multiplicity</i> kemungkinan jumlah pemakaian</p>

(Sumber : Prabowo Pudjo Widodo, Herlawati 2011)

II.8. Normalisasi

Normalisasi adalah bagian perancangan basisdata. Tanpa Normalisasi, sistem basisdata menjadi tidak akurat, lambat, tidak efisien, serta tidak memberikan data yang diharapkan.

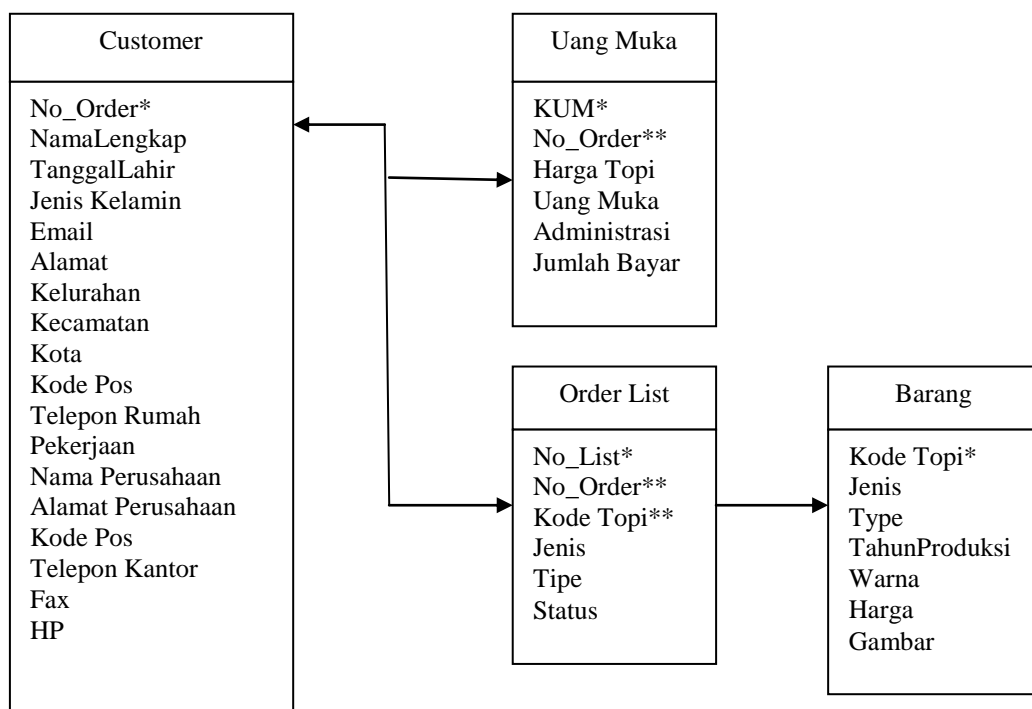
Pada waktu menormalisasi basisdata, ada empat tujuan yang harus dicapai, yaitu:

1. Mengatur data dalam kelompok-kelompok sehingga masing-masing kelompok hanya menangani bagian kecil sistem.
2. Meminimal jumlah data berulang dalam basisdata.

3. Membuat basisdata yang datanya diakses dan manipulasi secara cepat dan efisien tanpa melupakan integritas data.
4. Mengatur data sedemikian rupa sehingga ketika memodifikasi data, hanya mengubah satu tempat.

Normalisasi dimulai dengan menganalisa tabel dalam bentuk tidak normal, dan berikut contoh normalisasi.

Tabel II.4 Contoh Normalisasi



(Sumber : Iyan Gustiana ; 2011 : 7)

Tujuan Normalisasi adalah membuat kumpulan table relasional yang bebas dari sata berulang dan dapat dimodifikasi secara benar dan konsisten. Ini berarti bahwa semua table pada basisdata relasional harus berada pada bentuk normal ketiga (3NF). Sebuah table relasional berada pada 3NF jika dan hanya jika semua kolom bukan kunci adalah saling independen berarti bahwa tidak ada kolom bukan kunci yang tergantung pada sembarang kombinasi kolom lainnya. Dua bentuk normal pertama adalah langkah antara untuk mencapai tujuan, yaitu mempunyai semua table dalam 3NF. (Janner Si; 2010 : 77-78) Tahapan normalisasi terdiri dari beberapa bentuk yaitu sebagai berikut:

1. Bentuk normal pertama (1NF / *First Normal Form*)

Bentuk normal pertama memiliki ciri yaitu data berbentuk *flat file* (file datar), *record* disusun sesuai kedatangan, masih mungkin terjadi penyimpangan data (*anomaly data*). *Anomali* data dapat berupa *insert anomaly*, *delete anomaly*, *update anomaly* dan *redundancy data* (data duplikat). (Janner Si; 2010 : 79)

2. Bentuk Normal Kedua (2NF/ *Second Normal Form*)

Bentuk normal kedua memiliki ciri yaitu tidak terjadi *anomali data*, setiap *field / attribute* bukan kunci harus tergantung fungsi (*functional dependency*) terhadap

field / attribute kunci, masih mungkin terjadi *transitive dependency* (*field* bukan kunci tergantung pada *field* bukan kunci dalam satu table). (Janner Si; 2010 : 81)

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

Bentuk normal ketiga memiliki syarat yaitu table harus tidak terdapat *transitive dependency* (*field* bukan kunci tergantung pada *field* bukan kunci dalam satu tabel). (Janner Si; 2010 : 82)

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

Pada tahap ini menghilangkan ketergantungan *field* bukan kunci secara persial (bagian) kunci dalam satu tabel. Apabila pada normal ketiga tidak lagi ditemukan *field* bukan kunci tergantung secara persial dalam satu tabel, maka normal ketiga juga merupakan bentuk BCNF. (Janner Si; 2010 : 84)

5. Bentuk Normal Kelima (5NF / *Five Normal Form*)

Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*). Ketergantungan gabungan berarti bahwa sebuah tabel., setelah didekomposisi menjadi tiga atau lebih tabel yang lebih kecil, harus dapat digabungkan kembali untuk membentuk tabel asal. Dengan kata lain, 5NF menunjukkan ketika sebuah tabel tidak dapat didekomposisi lagi. (Janner Si ; 2010 : 85)

II.9. Mengenal Visual Basic

Menurut Edy Winarto; 2010:1. Visual Basic dibuat oleh microsoft, merupakan salah satu bahasa pemrograman berorientasi objek yang mudah

dipelajari. Selain menawarkan kemudahan, Visual Basic juga cukup andal untuk digunakan dalam pembuatan berbagai aplikasi, terutama aplikasi database. Visual basic merupakan bahasa pemrograman event drive, dimana program akan menunggu sampai ada respons dari user/pemakai program aplikasi yang dapat berupa kejadian atau event, misalnya ketika user mengklik tombol atau menekan enter. Jika kita membuat aplikasi dengan visual basic maka kita akan mendapatkan file yang menyusun aplikasi tersebut, yaitu :

1. File Project (*.vbp)

File ini merupakan kumpulan dari aplikasi yang kita buat. File project bisa berupa file *.frm, *.dsr atau file lainnya.

2. File Form (*.frm)

File ini merupakan file yang berfungsi untuk menyimpan informasi tentang bentuk form maupun interface yang kita buat.