

## **BAB II**

### **TINJAUAN PUSTAKA**

#### **II.1. Sistem**

Secara sederhana suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari suatu unsur, komponen, atau variabel yang terorganisir, saling tergantung satu sama lain dan terpadu. Teori sistem secara umum yang pertama kali diuraikan oleh Kennet Boulding, terutama menekankan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem. Kecenderungan manusia yang mendapat tugas memimpin suatu organisasi adalah terlalu memusatkan perhatian pada salah satu komponen saja dari sistem organisasi.

Teori sistem melahirkan konsep-konsep futuristik, antara lain yang terkenal adalah konsep sibernetika (*cybernetics*). Konsep atau dibidang kajian ilmiah ini berkaitan dengan upaya menerapkan berbagai ilmu yaitu ilmu perilaku, fisika, biologi, dan teknik. Oleh karena itu sibernetika biasanya berkaitan dengan usaha-usaha otomasi tugas-tugas yang dilakukan manusia, sehingga melahirkan studi-studi tentang robotika, kecerdasan buatan (*artificial intelegence*). Unsur-unsur yang mewakili suatu sistem secara umum adalah masukan (*input*), pengolahan (*processing*), dan keluaran (*output*). Selain itu, suatu sistem tidak bisa lepas dari lingkungan maka umpan balik (*feed back*) dapat berasal dari lingkungan sistem yang dimaksud. Organisasi dipandang sebagai suatu sistem yang tentunya akan memiliki semua unsur ini (Tata Sutabri, 2012).

Pengertian sistem dilihat dari elemen-elemennya adalah kumpulan elemen-elemen yang saling berkaitan dan bekerja sama dalam melakukan kegiatan untuk mencapai suatu tujuan, sedangkan pengertian sistem jika dilihat dari masukan dan keluarannya adalah suatu rangkaian yang berfungsi menerima input (masukan), mengolah input, dan menghasilkan output (keluaran). (V. Wiratna Sujarweni ; 2015. 1).

Dari kesimpulan diatas, Sistem adalah entitas atau satuan yang terdiri dari dua atau lebih komponen atau subsistem (sistem yang lebih kecil) yang saling terhubung dan terkait untuk mencapai suatu tujuan.

### **II.1.2. Informasi**

Informasi adalah data yang telah diklasifikasikan diolah atau diinterpretasi untuk digunakan dalam proses pengambil keputusan. Sistem pengolahan informasi mengolah data menjadi informasi atau tepatnya mengolah dari bentuk tak berguna menjadi berguna bagi penerimanya.

Informasi adalah data yang telah diklasifikasikan atau diinterpretasi untuk digunakan dalam pengambil keputusan (Tata Sutabri, 2012).

### **II.1.3. Sistem Informasi**

Sistem informasi adalah berupa suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan data transaksi harian yang mendukung operasi yang bersifat manajerial dengan kegiatan strategi suatu

organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan (Tata Sutabri, 2012)

## **II.2. Depo Peti Kemas**

Depo Peti Kemas adalah suatu tempat di dalam atau di luar Daerah Lingkaran Kerja Pelabuhan (DLKR) yang berfungsi untuk kegiatan penyimpanan, penumpukan, pembersihan / pencucian, perawatan, perbaikan peti kemas, pemuatan (Stuffing), pembongkaran (Stripping) serta kegiatan lain yang mendukung kelancaran penanganan peti kemas (full) dan atau peti kemas kosong (Empty).

### **II.2.1. Dasar Hukum**

1. Kepmenhub No. 47 Tahun 2008 Tentang Penyelenggaraan dan Pengusahaan Depo Peti Kemas.
2. UU No. 17 Tahun 2008 Bagian ke tujuh pasal 31-34 tentang Pelayaran.
3. PP No. 20 Tahun 2010 Bab VI pasal 79-86 Tentang Angkutan di Perairan.
4. Perda No. 1 Tahun 2012 Tentang Retribusi Daerah.

### **II.2.2. Asosiasi**

Asosiasi Depo Peti Kemas Indonesia (ASDEKI)

Visi :

- 1). Mewujudkan pelayanan transportasi yang berkualitas untuk mendukung peningkatan kesejahteraan masyarakat dan pembangunan daerah.

Misi :

- 1). Meningkatkan pemerataan pelayanan, dengan prioritas pada Wilayah Kepulauan dan masyarakat berpenghasilan rendah.
- 2). Meningkatkan pelayanan yang murah, mudah, aman, nyaman dan cepat.
- 3). Meningkatkan peran bidang Transportasi dalam percepatan dan pemerataan proses pembangunan daerah.
- 4). Mendorong partisipasi masyarakat dan dunia usaha dalam meningkatkan Pelayanan perhubungan.

<http://finance.detik.com/read/2012/05/06/174323/1910383/1016/setelah-ritz-carlton-tomy-winata-cs-bangun-hotel-mewah-baru-di-scbd>).

### **II.3. Normalisasi**

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relasional (*www. utexas. edu*).

### 1. Bentuk Normal Pertama (1 NF)

Contoh yang kita gunakan di sini adalah sebuah perancangan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri. Masing-masing pemasok bisa menyediakan banyak barang. Tabel relasionalnya dapat dituliskan sebagai berikut :

PEMASOK (P#, Status, Kota, b#, qty) di mana

p# : kode pemasok (kunci utama)

status : kode status kota

Kota : nama kota

b# : barang yang dipasok

qty : jumlah barang yang dipasok.

Sebuah tabel relasional secara defenisi selalu berada dalam bentuk normal pertama. Semua nilai pada kolom-kolomnya adalah atomi. Ini berarti kolom-kolom tidak mempunyai nilai berulang. Tabel II.1. menunjukkan tabel pemasok dalam 1 NF.

**Tabel II.1. Normalisasi Pertama Pemasok**

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

**Sumber : (Janner Simarmata, dkk, 2010).**

## 2. Bentuk Normal Kedua (2 NF).

Defenisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1 NF, tetapi tidak pada 2 NF, sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada 1 NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada kunci utama. Ini berarti bahwa setiap kolom bukan kunci harus tergantung pada seluruh kolom yang membentuk kunci utama. Tabel pemasok berada pada 1 NF, tetapi tidak pada 2 NF karena status dan kota tergantung secara fungsional hanya pada kolom p# dari kunci gabungan (p#, b#). Ini dapat digambarkan dengan membuat daftar ketergantungan fungsional.

P# → Kota, Status

Kota → Status

(P#, B#) → qty

Proses mengubah tabel 1 NF ke 2 NF adalah :

- a. Tentukan sembarang kolom penentu selain kunci gabungan dan kolom-kolom yang ditentukannya.
- b. Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom-kolom yang ditentukan.
- c. Pindahkan kolom-kolom yang ditentukan dari tabel asal ke tabel baru penentu akan menjadi kunci utama pada tabel baru.
- d. Hapus kolom yang baru dipindahkan dari tabel asal, kecuali penentu yang akan berfungsi sebagai kunci tamu.
- e. Tabel asal bisa diberi nama baru.

Pada contoh, kita memindahkan kolom p#, status, dan kota ke tabel baru yang disebut pemasok2. Kolom p# menjadi kunci utama tabel ini.

Tabel II.3. menunjukkan hasilnya

**Tabel II.2. Tabel Bentuk Normal Kedua (2NF).**

Pemasok2  
Barang

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

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

**Sumber : (Janner Simarmata, dkk, 2010).**

### 3. Bentuk Normal Ketiga (3 NF).

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional hanya pada kunci utama. Secara defenisi, sebuah tabel berada pada bentuk normal ketiga (3 NF) jika tabel sudah berada pada 2 NF dan setiap kolom yang bukan kunci tidak tergantung secara transistif pada kunci utamanya. Dengan kata lain, semua atribut bukan kunci tergantung secara fungsional hanya pada kunci utama. Tabel barang sudah dalam bentuk normal ketiga. Kolom bukan kunci, qty, tergantung sepenuhnya pada kunci utama (p#, b#). Pemasok masih berada pada 2 NF, tetapi belum berada pada 3 NF karena dia mengandung ketergantungan transitif. Ketergantungan transitif terjadi ketika sebuah kolom bukan kunci, yang ditentukan oleh kunci utama, menentukan kolom lainnya. Konsep ketergantungan transistif daopat digambarkan dengan menunjukkan ketergantungan fungsional pada pemasok2, yaitu :

Pemasok2. p#  $\longrightarrow$  Pemasok2, status

Pemasok2. p#  $\longrightarrow$  Pemasok2, kota

Pemasok2. kota  $\longrightarrow$  Pemasok2, status

Perlu dicatat bahwa pemasok2, status ditentukan, baik oleh kunci utama p#, maupun kolom bukan kunci, kota

Proses mengubah tabel menjadi 3 NF adalah :

- a. Tentukan semua penentu selain kunci utama dan kolom yang ditentukannya.

- b. Buat dan beri nama tabel baru untuk masing-masing penentu dan kolom yang ditentukannya.
- c. Pindahkan kolom yang ditentukan dari tabel asal ke tabel baru. Penentu menjadi kunci utama tabel baru.
- d. Hapus kolom yang baru saja dipindahkan dari tabel asal, kecuali penentu yang akan berfungsi sebagai kunci tamu.
- e. Tabel asal bisa diberi nama baru.

Untuk mengubah PEMASOK2 menjadi 3 NF, kita membuat tabel baru yang disebut KOTA\_STATUS dan memindahkan kolom kota dan status ke tabel baru. Status dihapus dari tabel diberi nama baru PEMASOK\_KOTA. Tabel II.3 menunjukkan hasilnya.

**Tabel II.3. Tabel Bentuk Normal Ketiga (3 NF)**

PEMASOK_KOTA		KOTA_STATUS	
P#	Kota	Kota	Status
P1	Yogyakarta	Yogyakarta	20
P2	Medan	Medan	10
P3	Medan	Bandung	30
P4	Yogyakarta	Semarang	40
P5	Bandung		

**Sumber : (Janner Simarmata, dkk, 2010).**

#### 4. Bentuk Normal Boyce Code (BCNF)

Setelah 3 NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3 NF sudah cukup karena sangat jarang entitas yang berada pada 3 NF bukan merupakan 4 NF

dan 5 NF. Lebih lanjut, mereka berpendapat bahwa keuntungan yang didapat mengubah entitas ke 4 NF dan 5 NF sangat kecil sehingga tidak perlu dikerjakan. Bentuk Normal Boyce- Code (BCNF) adalah versi 3 NF lebih teliti dan berhubungan dengan tabel relasional yang mempunyai (a) banyak kunci kandidat (b) kunci kandidat gabungan, dan (c) kunci kandidat yang saling tumpang tindih.

BCNF didasarkan pada konsep penentu. Sebuah kolom penentu adalah kolom di mana kolom-kolom lain sepenuhnya tergantung secara fungsional. Sebuah tabel relasional berada pada BCNF jika dan hanya setiap penentu adalah kunci kandidat.

#### 5. Bentuk Normal Keempat (4 NF)

Sebuah tabel relasional berada pada bentuk normal keempat (4 NF) jika dia dalam BCNF dan semua ketergantungan multivalued merupakan ketergantungan fungsional.

Bentuk normal keempat (4 NF) didasarkan pada konsep ketergantungan multivalued (MVD). Sebuah ketergantungan multivalued terjadi ketika dalam sebuah tabel relasional yang mengandung setidaknya tiga kolom, satu kolom mempunyai banyak baris bernilai sama, tetapi kolom lain bernilai berbeda. Defenisi secara formal diberikan oleh CJ. Date, yaitu :

Misalnya, ada sebuah tabel relasional R dengan kolom A, B dan C, Maka  $R.A \twoheadrightarrow R.B$  (kolom A menentukan kolom B). Adalah benar jika dan hanya jika himpunan nilai B yang cocok dengan pasangan nilai A

dan nilai C pada R hanya tergantung pada nilai A dan tidak tergantung pada nilai C.

MVD selalu terjadi dalam pasangan, yaitu  $R:A \twoheadrightarrow R:B$  dipenuhi jika dan hanya jika  $R:A \twoheadrightarrow R:C$  dipenuhi pula.

#### 6. Bentuk Normal Kelima (5 NF).

Sebuah tabel berada pada bentuk normal kelima jika dia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*). Ketergantungan gabungan berarti sebuah tabel, setelah dekomposisi menjadi tiga atau lebih tabel yang lebih kecil, harus dapat digabungkan kembali untuk membentuk tabel asal. Dengan kata lain 5 NF menunjukkan ketika sebuah tabel tidak dapat didekomposisi lagi (Janner Simarmata, 2012).

### II.4. Basis Data (*Database*)

Basis data menurut (Stephen dan Plew, 2000) adalah mekanisme yang digunakan untuk menyimpan informasi atau data. Kemudian (Silberchatz, dkk, 2002) mendefinisikan basis data sebagai kumpulan data berisi informasi yang sesuai untuk perusahaan. Sistem manajemen basis data (DBMS) adalah kumpulan data yang saling berhubungan dan kumpulan program untuk mengakses data. Tujuan utama sistem manajemen basis data adalah menyediakan cara menyimpan dan mengambil informasi basis data secara mudah dan efisien.

Menurut (Ramakrishnan dan Gehkre, 2003) menyatakan basis data sebagai kumpulan data, yang umumnya mendeskripsikan aktivitas satu organisasi atau lebih yang berhubungan (Janner Simarmata, dkk, 2010).

#### 1. Keuntungan DBMS (*Database Management System*)

DBMS memungkinkan perusahaan maupun pengguna individu untuk :

##### a. Mengurangi perulangan data.

Apabila dibandingkan dengan *file-file* komputer yang disimpan terpisah di setiap lokasi komputer. DBMS mengurangi jumlah total *file* dengan menghapus data yang terduplikasi di berbagai *file*. Data terduplikasi selebihnya dapat ditempatkan dalam satu *file*.

##### b. Mencapai independensi data.

Spesifikasi data disimpan dalam skema pada tiap program aplikasi. Perubahan dapat dibuat pada struktur data tanpa mempengaruhi program yang mengakses data.

##### c. Mengintegrasikan data beberapa *file*.

Saat *file* dibentuk sehingga menyediakan kegiatan logis, maka organisasi fisik bukan merupakan kendala. Organisasi logis, pandangan pengguna, dan program aplikasi tidak harus tercemin pada media.

##### d. Mengambil data dan informasi dengan cepat.

Hubungan-hubungan logis, bahasa manipulasi data, serta bahasa *query* memungkinkan pengguna mengambil data dalam hitungan detik atau menit.

e. Meningkatkan keamanan

DBMS *mainframe* maupun komputer mikro dapat menyertakan beberapa lapis keamanan seperti kata sandi (*password*), direktori pemakai, dan bahasa sandi (*encryption*) sehingga data yang dikelola akan lebih aman.

1. Kerugian DBMS

Keputusan menggunakan DBMS mengikat perusahaan atau pengguna untuk :

a. Memperoleh perangkat lunak.

DBMS *mainframe* masih sangat mahal. Walaupun harga DBMS berbasis komputer mikro lebih murah, tetapi tetap merupakan pengeluaran besar bagi suatu organisasi kecil.

b. Memperoleh konfigurasi perangkat keras yang besar.

DBMS sering memerlukan kapasitas penyimpanan dan memori lebih besar dari pada program aplikasi lain.

c. Mempekerjakan dan mempertahankan staf DBA.

DBMS memerlukan pengetahuan khusus agar dapat memanfaatkan kemampuannya secara penuh. Pengetahuan khusus ini disediakan paling baik oleh para pengguna basis data (DBA)

Baik basis data terkomputerisasi maupun DBMS bukanlah prasyarat untuk memecahkan masalah. Namun, keduanya memberikan dasar-dasar menggunakan komputer sebagai suatu sistem informasi bagi para spesialis informasi dan pengguna (Janner Simarmata, dkk, 2010).

## II.5. *Unified Modeling Language (UML)*

UML singkatan dari *Unified Modelling Language* yang berarti bahasa pemodelan standart. (Chonoles, 2003) 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 harus berhubungan satu dengan lainnya harus mengikuti standart yang ada. *UML* bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya? Bagaimana sistem mengatasi error yang terjadi? Bagaimana keamanan terhadap sistem yang ada kita buat? Dan sebagainya dapat dijawab dengan *UML*.

*UML* diaplikasikan untuk maksud tertentu, biasanya antara lain untuk :

1. Merancang perangkat lunak.
2. Sarana komunikasi antara perangkat lunak dengan bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.

*UML* telah diaplikasikan dalam investasi perbankan, lembaga kesehatan, departemen pertahanan, sistem terdistribusi, sistem pendukung alat kerja, retail, sales, dan supplier.

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 berorientasikan objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. *UML* memungkinkan para anggota team untuk bekerja sama dalam mengaplikasikan beragam sistem. Intinya, *UML* merupakan alat komunikasi yang konsisten dalam mensupport para pengembang sistem saat ini. Sebagai perancang sistem mau tidak mau pasti menjumpai *UML*, baik kita sendiri yang membuat sekedar membaca diagram *UML* buatan orang lain (Prabowo Pudjo Widodo Dan Herlawati, 2011).

### **II.5.1. 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, serta relasi-relasi diagram. Diagram ini umum dijumpai pada pemodelan sistem berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas.
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 waktu tertentu.
5. Diagram Komunikasi (*Communication Diagram*) bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi *UML* 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 sistem. Diagram ini terutama penting dalam pemodelan fungsi-fungsi suatu sistem dan member tekanan pada aliran kendali antar objek.
8. Diagram Komponen (*Component Diagram*) Bersifat Statis. Diagram komponen ini memperlihatkan organisasi serta kebergantungan

sistem/perangkat lunak pada komponen-komponen yang telah ada sebelumnya. Diagram ini berhubungan diagram kelas dimana komponen dipetakan kedalam satu atau lebih kelas-kelas. Antarmuka-antarmuka 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*).

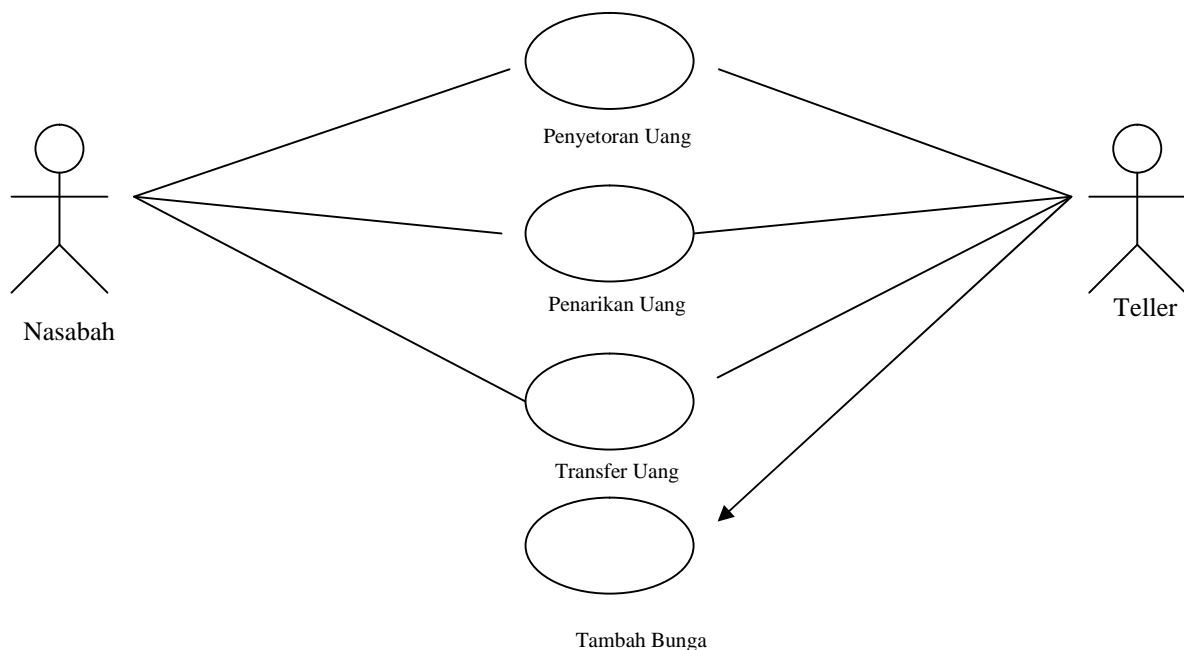
Kesembilan diagram ini tidak mutlak harus digunakan dalam pengembangan perangkat lunak, semuanya dibuat sesuai dengan kebutuhan. Pada *UML* dimungkinkan kita menggunakan diagram-diagram lainnya misalnya *Data Flow Diagram*, *Entity Relationship Diagram* dan sebagainya (Prabowo Pudjo Widodo, Dan Herlawati,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

1. *Diagram Use Case (Use Case Diagram)*

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

- a. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.
- b. *Use Case*, aktivitas/ sarana yang disiapkan oleh bisnis/sistem.
- c. Hubungan (*Link*), aktor mana saja yang terlibat dalam *use case* ini.

Gambar di bawah ini merupakan salah satu contoh bentuk diagram *use case* (Prabowo Pudjo Widodo Dan Herlawati, 2011).

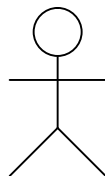


**Gambar II.1. Diagram *Use Case***

**Sumber : (Prabowo Pudjo Widodo Dan Herlawati, 2011)**

### 1. Aktor

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

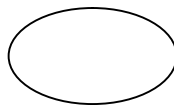


## Gambar II.2. Aktor

Sumber : (Prabowo Pudjo Widodo Dan Herlawati, 2011)

### 2. Use Case

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



## Gambar II.3. Simbol Use Case

Sumber : (Prabowo Pudjo Widodo Dan Herlawati, 2011)

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

#### a. Pilihlah Nama Yang Baik

*Use case* adalah sebuah *behaviour* (perilaku), jadi seharusnya dalam frase kata kerja. Untuk membuat namanya lebih detil tambahkan kata benda mengindikasikan dampak aksinya terhadap suatu kelas objek. Oleh karena itu diagram *use case* seharusnya berhubungan dengan diagram kelas.

#### b. Ilustrasikan Perilaku Dengan Lengkap.

*Use case* dimulai dari inisiasi oleh aktor primer dan berakhir pada aktor dan menghasilkan tujuan. Jangan membuat *use case* kecuali anda mengetahui tujuannya. Sebagai contoh memilih 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).

c. Identifikasi Perilaku Dengan Lengkap.

Untuk mencapai tujuan dan menghasilkan nilai tertentu dari aktor, *use case* harus lengkap. Ketika memberi 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 menggambarkan perilaku yang belum selesai.

d. Menyediakan Use Case Lawan (*Inverse*)

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

e. Batasi Use Case Hingga Satu Perilaku Saja.

Kadang kita cenderung membuat *use case* yang lebih dari satu tujuan aktivitas. Guna menghindari kerancuan, jagalah *use case* kita hanya fokus pada satu hal. Misalnya, penggunaan *use case check in* dan *check out* dalam satu *use case* menghasilkan ketidakfokusan, karena memiliki dua perilaku yang berbeda.

### 3. 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 program menjadi model (Prabowo Pudji Widodo Dan Herlawati, 2011)

### 4. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistem dari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukkan aktivitas sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, diagram aktivitas merepresentasikan pemanggilan suatu fungsi tertentu misalnya *call*. Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian diluar seperti pemesanan atau kejadian-kejadian internal misalnya penggajian tiap jumat sore (Prabowo Pudji Widodo, Dan Herlawati, 2011).

Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi nelakukan langka sekali saja tidak boleh dipecah menjadi beberapa langkah-langkah lagi.

Contoh aksinya yaitu :

- a. Fungsi Matematika.
- b. Pemanggilan Perilaku.
- c. Pemrosesan Data.

Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu *classifier* dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classifier*, tiap objek yang terhubung dan parameter-parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan dengan 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.

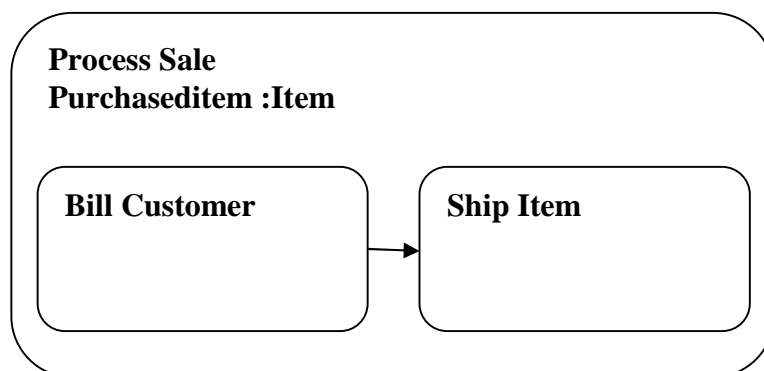
Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



**Gambar II.4. Aktivitas sederhana tanpa rincian**

**Sumber : (Prabowo Pudjo Widodo Dan Herlawati, 2011)**

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



**Gambar II.5. Aktivitas dengan detail rincian**  
**Sumber : (Prabowo Pudjo Widodo Dan Herlawati, 2011)**

5. *Sequence Diagram*

Menurut (Douglas, 2004) menyebutkan ada tiga diagram primer UML dalam memodelkan scenario interaksi, yaitu diagram urutan (*sequence diagram*), diagram waktu (*timing diagram*) dan diagram komunikasi (*communication diagram*).

Menurut (Pilone, 2005) menyatakan bahwa diagram yang paling banyak dipakai adalah diagram urutan. Gambar II.9. memperlihatkan contoh diagram urutan dengan notasi-notasinya yang akan dijelaskan nantinya (Prabowo Pudjo Widodo Dan Herlawati, 2011).

## **II.6. Bahasa Pemograman *Microsoft Visual Studio 2010***

*Microsoft Visual Studio 2008* merupakan kelanjutan dari *Microsoft Visual Studio* sebelumnya, yaitu *Visual Studio. Net 2003* yang diproduksi oleh Microsoft. Pada bulan Februari 2002 *Microsoft* memproduksi teknologi. *Net Framework* versi 1.0, teknologi. *Net* ini didasarkan atas susunan berupa *Net Framework*, sehingga setiap produk baru yang terkait dengan teknologi. *Net* akan selalu berkembang mengikuti perkembangan. *Net Frameworknya*.

Pada perkembangannya nantinya mungkin untuk membuat program dengan teknologi. *Net* memungkinkan para pengembang perangkat lunak akan dapat menggunakan lintas sistem operasi, yaitu dapat dikembangkan di sistem operasi

windows juga dapat dijalankan pada sistem operasi lain, misalkan pada sistem operasi *Linux*, seperti yang telah dilakukan pada pemograman *Java* oleh *Sun Microsystem*. Pada saat ini perusahaan-perusahaan sudah banyak mengupdate aplikasi lama yang dibuat *Microsoft Visual Basic 6.0* ke teknologi *.Net* karena kelebihan-kelebihan yang ditawarkan, terutama memungkinkan pengembang perangkat lunak secara cepat mampu membuat program *robust*, serta berbasiskan integrasi ke internet yang dikenal dengan *XML Web Service* (Erick Kurniawan, 2010)

## II.7. *MYSQL*

*Mysql* adalah salah satu software sistem manajemen *database* (DBMS) *structured Query Language (SQL)* yang bersifat *open source*. *SQL* adalah bahasa standart untuk mengakses *database* dan didefenisikan dengan standart *ANSI/ISO SQL*. *MYSQL* dikembangkan, disebarluaskan, dan didukung oleh *MYSQL AB*. *MYSQL AB* adalah perusahaan komersial yang didirikan oleh pengembang *MYSQL*. *MYSQL* merupakan aplikasi *Relational Database Management System* (RDBMS) yang digunakan untuk aplikasi *client server* atau sistem *embedded*.

*Mysql* mempunyai beberapa sifat yang menjadikannya sebagai salah satu software database yang banyak digunakan oleh pemakai diseluruh dunia. Sifat-sifat yang dimiliki oleh *MYSQL* antara lain :

- a. *Mysql* merupakan DBMS (*Database Management System*)
- b. *Database* adalah kumpulan data yang terstruktur. Data dapat berupa daftar belanja, kumpulan gambar, atau yang lebih luas yaitu informasi jaringan

perusahaan. Agar dapat menambah, mengakses, dan memproses data tersimpan pada sebuah komputer database, kita membutuhkan sistem manajemen *database* (DBMS) seperti *MYSQL Server*. Sejak komputer sangat baik menangani sejumlah besar data, *sistem manajemen database* (DBMS) memainkan peran utama dalam perhitungan baik sebagai peralatan yang berdiri sendiri maupun bagian aplikasi.

- c. *Mysql* merupakan RDBMS (*Relational Database Management System*).
- d. *Database relational* menyimpan data pada table-table yang terpisah, bukan menyimpan data dalam ruang penyimpanan yang besar. Hal ini menambah kecepatan *fleksibilitas*.
- e. *Mysql* merupakan *software open source*.
- f. *Open source* berarti setiap orang dapat menggunakan dan mengubah software yang bersangkutan. Setiap orang dapat mendownload *software MYSQL* dari internet dan menggunakan tanpa membayar. Bahkan jika mengkehendaknya anda bisa mempelajari kode sumber dan mengubah sesuai yang anda butuhkan (Wahana Komputer, 2010).