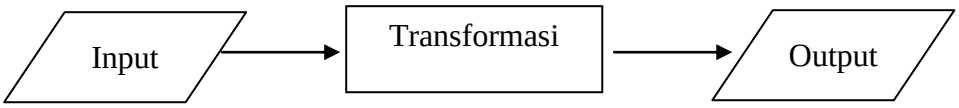
BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem

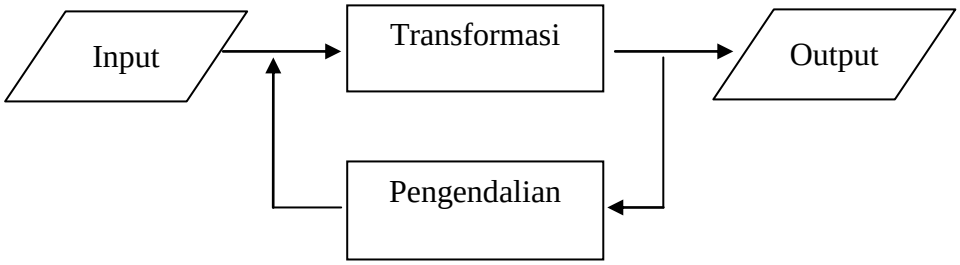
Sistem adalah suatu kesatuan usaha yang terdiri dari bagian – bagian yang berkaitan satu sama lain yang berusaha mencapai suatu tujuan dalam suatu lingkungan kompleks. Pengertian tersebut mencerminkan adanya beberapa bagian dan hubungan antar bagian, ini menunjukkan kompleksitas dari sistem yang meliputi kerja sama antara bagian yang interdependen satu sama lain. Selain itu, dapat dilihat bahwa sistem berusaha mencapai tujuan. Pencapaian tujuan ini menyebabkan timbulnya dinamika, perubahan yang terus menerus perlu dikembangkan dan dikendalikan. Definisi tersebut menunjukkan bahwa sistem sebagai gugus dari elemen – elemen yangs aling berinteraksi secara teratur dalam rangka mencapai tujuan atau subtujuan. Sifat – sifat dasar suatu sistem, antara lain :

1. Pencapaian tujuan, orientasi pencapaian tujuan akan memberikan sifat dinamis kepada sistem, memberi ciri perubahan yang terus – menerus dalam usaha mencapai tujuan.
2. Kesatuan usaha, mencerminkan suatu sifat dasar dari sistem, dimana hasil keseluruhan melebihi dari jumlah bagian – bagiannya atau sering disebut konsep sinergi.
3. Keterbukaan terhadap lingkungan, lingkungan merupakan sumber kesempatan maupun hambatan pengembangan. Keterbukaan terhadap lingkungan membuat penilaian terhadap suatu sistem menjadi relatif atau yang dinamakan *equifinality* atau pencapaian tujuan suatu sistem tidak mutlak harus dilakukan melalui berbagai cara sesuai dengan tantangan lingkungan yang dihadapi.
4. Transformasi, merupakan proses perubahan *input* menjadi *output* yang dilakukan oleh sistem, proses transformasi diilustrasikan pada gambar II.1.



Gambar II.1. Proses Transformasi Input Menjadi Output.
(Sumber : Prof. Dr. Ir. Marimin, M.Sc. ; 2011 : 2)

5. Hubungan antara bagian, kaitan antara subsistem inilah yang akan memberikan analisis sistem, suatu dasar pemahaman yang lebih luas.
6. Sistem ada berbagai macam, antara lain sistem terbuka, sistem tertutup, dan sistem dengan umpan balik.
7. Mekanisme pengendalian, mekanisme ini menyangkut sistem umpan balik yang merupakan suatu bagian yang memberi informasi kepada sistem mengenai efek dari perilaku sistem terhadap pencapaian tujuan atau pemecahan persoalan yang dihadapi. Skema proses transformasi sistem dengan mekanisme pengendalian disajikan pada gambar II.2.



Gambar II.2. Skema Proses Transformasi Sistem Dengan Mekanisme Pengendalian.
(Sumber : Prof. Dr. Ir. Marimin, M.Sc. ; 2011 : 3)

II.2. Pengertian Informasi

Informasi adalah data yang sudah diolah menjadi sebuah bentuk yang berarti bagi pengguna, yang bermanfaat dalam pengambilan keputusan saat ini

atau mendukung sumber informasi. Data belum memiliki nilai sedangkan informasi sudah memiliki nilai. Informasi dikatakan bernilai bila manfaatnya lebih besar dibanding biaya untuk mendapatkannya.

II.2.1. Kualitas Informasi

Informasi yang berkualitas memiliki 3 kriteria, yaitu :

1. Akurat (*accurate*)

   Informasi harus bebas dari keselahan, tidak bisa ataupun menyesatkan. Akurat juga berarti bahwa informasi itu harus dapat dengan jelas mencerminkan maksudnya.

2. Tepat pada waktunya (*timeliness*)

   Informasi yang datang pada penerima tidak boleh terlambat. Di dalam pengambilan keputusan, informasi yang sudah usang tidak lagi bernilai. Bila informasi datang terlambat dilakukan., hal itu dapat berakibat fatal bagi perusahaan.

3. Relevan

   Informasi yang disampaikan harus mempunyai keterikatan dengan masalah yang akan dibahas dengan informasi tersebut. Informasi bermanfaat bagi pemakaiannya. Disamping karakteristik, nilai informasi juga ikut menentukan kualitasnya. Nilai informasi (*value of information*) ditentukan oleh dua hal, yaitu manfaat dan biaya untuk mendapatkannya. Suatu informasi dikatakan bernilai bila manfaatnya lebih besar dibanding biaya untuk mendapatkannya. (Kusrini M.Kom ; 2010 : 7-8).

II.3. Pengertian Sistem Informasi

Untuk menghasilkan sistem informaasi yang berkualitas maka dibuatlah sistem informasi. Sistem informasi didefinisikan oleh Robert A. Laitch dan K. Roscoe Bavis sebagai berikut. “Sistem Informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian, mendukung operasi, bersifat manajerial dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan – laporan yang diperlukan. (Kusrini M.Kom ; 2010 : 8).

II.4. Pengertian Volume Penjualan

Penjualan merupakan tujuan utama dilakukannya kegiatan perusahaan. Perusahaan dalam menghasilkan barang/jasa mempunyai tujuan akhir yaitu untuk menjual barang/jasa tersebut kepada masyarakat dan untuk memperoleh laba. Penjualan merupakan pengalihan milik atas barang dengan imbalan uang sebagai gantinya dengan persetujuan untuk menyerahkan barang kepada pihak lain dengan menerima pembayaran. (Freddy Rangkuti ; 2010 : 57-58)

II.5. Mengenal Visual Basic 2010

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, (Edy Winarto ; 2010 : 1).

II.5.1. Mengetahui Antar Muka Visual Basic 2010

Saat anda menjalankan Visual Basic 2010 pertama kali, muncul jendela Choose Default Environmetn Setting. Disini anda bisa memilih apakah ingin memilih tipe antarmuka di visual studio, untuk programmer Visual Basic, lebih aik pilih Visual Basic Development Settings.

Gambar II.3. Pemilihan Default Environmet Settings
(Sumber : Edy Winarto ; 2010 : 1)

II.5.2 Keistimewaan Visual Basic

1. Menggunakan platform pembuatan program yang diberi naama *Developer Studio* , yang memiliki tampilan dan saran yang sama dengan visual C++ dan Visual J++ . Degan begitu anda dapat bermigrasi atau belajar pemograman lainnya dengan mudah dan cepat, tanpa harus belajar dari nol lagi.
2. Memiliki *compiler* andal yang dapat menghasilkan file *executable* yang lebih ce;at dan lebih efesien dari sebelumnya.
3. Memiliki beberapa sarana wizard yang baru. Wizard adalah sarana yang mempermudah didalam pembuatan aplikasi dengan mengotomatisasi tugas – tugas tertentu.
4. Tambahan kontrol – kontol baru yang lebih canggih.
5. Sarana akses data yang lebih cepadan handal untuk membuat aplikasi database yang berkemampuan tinggi. (Adi Kurniadi ; 2000 : 7)

II.5.3. Versi – versi Visual Basic

Seperti aplikasi lainnya, visual basic juga dipasarkan dalam berbagai jenis atau versi. Beberapa versi dari visual basic yang ada dipasaran antara lain :

1. *Standard Editon /Learning Edition :* Ini adalah versi standar yang sudah mencakup berbagai sarana dasar dari Visual Basic untuk mengembangkan aplikasi.
2. *Proffesional Edition :* versi ini memberikan berbagai sarana ekstra yang dibutuhkan oleh para programmer profesional. Misalnya seperti kontrol –

kontrol tambahan, dukungan utuk pemrograman internet, *compiler* untuk membuat *file help*, serta sarana pengembangan database yang lebih baik.

3. *Enterprise Edition :* versi ini dikhususkan untuk para programmer yang ingin mengembangkan aplikasi *remote computing / client/server*. Biasanya versi ini digunakan untuk membuat aplikasi pada jaringan.

II.6. Database

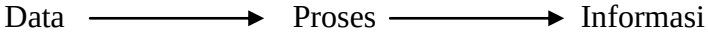
Banyak sekali definisi tentang database yang diberikan oleh para pakar dibidang ini. Database terdiri dari dua penggalan kata yaitu data dan base, yang artinya berbasiskan pada data. Tetapi secara konseptual, database diartikan sebuah koleksi atau kumpulan data yang saling berhubungan (relation), disusun menurut aturan tertentu secara logis, sehingga menghasilkan informasi. Sebuah informasi yang berdiri sendiri tidaklah dikatakan database. Contoh : Nomor telepon seorang pelanggan, disimpan dalam banyak tempat apakah itu di file pelanggan, di file alamat dan di lokasi yang lain. Antara file yang satu dengan file yang lainnya tidak saling berhubungan, sehingga apabila salah seorang pelanggan berganti nomor telepon dan anda hanya mengganti di file pelanggan saja, akibatnya akan terjadi ketidakcocokan data, karena di lokasi yang lain masih tersimpan data telepon yang lama.

Dalam sistem database hal ini tidak boleh dan tidak bisa terjadi, karena antara file yang satu dengan file yang lain saling berhubungan. Jika suatu data yang sama anda ubah, data tersebut di file yang lain akan otomatis berubah juga. Sehingga mampu menjadi informasi yang diinginkan dan dapat dilakukan proses

pengambilan, penghapusan, pengeditan, terhadap data secara mudah dan cepat (Efektif, Efisien dan Akurat).

Data adalah fakta, baik berupa sebuah objek, orang dan lain – lain yang dapat dinyatakan dengan suatu nilai tertentu (angka, simbol, karakter tertentu, dan lain – lain). Sedangkan informasi adalah data yang telah diolah sehingga bernilai guna dan dapat dijadikan bahan dalam pengambilan keputusan, (Yuhefizard ; 2010 : 2).

Hubungan data dan informasi dapat digambarkan sebagai berikut :



Gambar II.4. Data dan Informasi.
(Sumber : Yuhefizard ; 2010 : 2)

1. Keuntungan DBMS (*Database Managemen System)*

   DBMS memungkinkan perusahaan maupun pengguna individu untuk :

   a. Mengurangi perulangan data

      Apabila dibandingkan dengan file – file komputer yang disimpan terpisah disetiap lokasi komputer. DBMS mengurangi jumlsh total file dengan menghapus data yang 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

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

d. Mengambil data dan informasi lebih cepat

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

e. Meningkatkan keamanan

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

2. 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 informasi 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.

Baik basis data terkomputerisasi maupun DBMS bukanlah persyaratan untuk memecahkan masalah. Namun, keduanya memberikan dasar – dasar menggunakan komputer sebagai suatu sistem informasi bagi para spesialis informasi dan pengguna (janner Simarmata; 2008: 8-9).

II.7. SQL Server 2008

SQL *Server* 2008 adalah sebuah RDBMS (*Relational Database Management System*) yang di *develop* oleh *Microsoft*, yang digunakan untuk menyimpan dan mengolah data. Pada SQL *Server* 2008, kita bisa melakukan pengambilan dan modifikasi data yang ada dengan cepat dan efisien. Pada SQL *Server* 2008, kita bisa membuat *object – object* yang sering digunakan pada aplikasi bisnis, seperti membuat *database*, *table*, *fuction*, *stored procedure*, *trigger* dan *view*. Selain *object*, kita juga menjalankan perintah SQL (*Structured Query Language*) untuk mengambil data. (Cybertron Solution ; 2010 : 101).

11.7.1 Memulai SQL Server

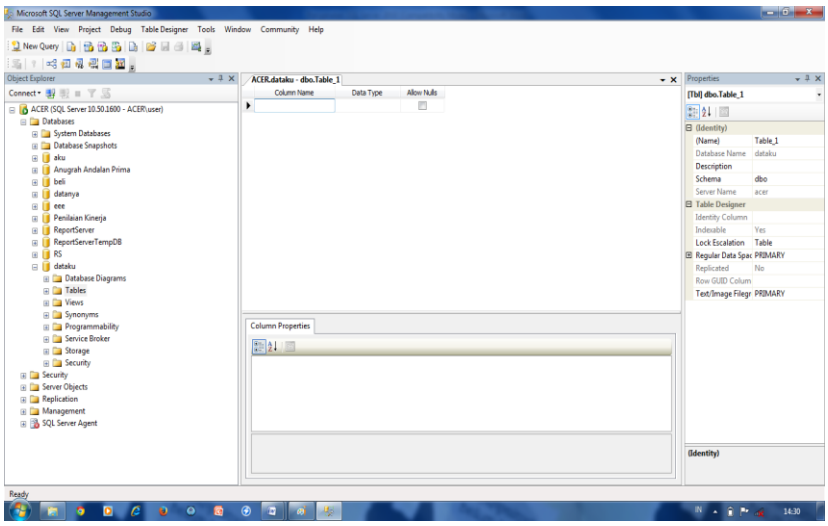
Untuk menggunakan SQL Server 2008, kita bisa membuka aplikasi tersebut melalui Start > Menu > All program > Microsoft SQL Server 2008 > SQL server Management Studio. Selanjutnya pada windows akan terbuka

aplikasai untuk melakukan aktivitas database SQL Server 2008. Untuk menggunakannya kita harus terlebih dahulu login.

Untuk melihat tampilan login SQL Server 2008 dapat dilihat pada gambar II.5 Sebagai berikut :

Gambar II.5. Tampilan Login SQL Server 2008
Sumber : Cybertron Solution ; 2010 : 102

Setelah berhasil login, kita bisa memulai aktivitas pada Database SQL Server 2008, seperti menuliskan perinah SQL Query melakukan *maintenance database* dan sebagainya.



Gambar II. 6. SQL Server Management Studio
(Sumber : Cybertron Solution ; 2010 : 102)

II.7.2. Interface SQL Server 2008

Ada 3 interface utama saat bekerja dengan SQL Server 2008 adalah sebagai berikut :

1. *Registered Server*

   Bila pada tampilan pertama anda tidak melihat panel ini maka anda akan dapat menampilkannya pada menu *view Registered Server* atau menekan kombinasi tombol CTRL + ALT + . Panel ini memungkinkan anda menjaga koneksi – koneksi dengan server – server yang pernah digunakan. Koneksi – koneksi ini dapat digunakan untuk memeriksa dari server tersebut (*online* atau *offline*) atau melakukan pada obyek – obyeknya (menggunakan pada panel object *explorer*). Setiap user memiliki daftar tersendiri dari *Registered Server* yang disimpan pada mesin lokal. Anda dapat melakukan penambahan atau pengurangan koneksi – koneksi ke server tersebut berdasarkan tipe servernya yaitu *Database Engine, Analysis Service, Reporting Service, Integration Service*.

2. *Object Explorer*

   Anda dapat melihat berbagai obyek yang ada pada sebuah server pada panel ini. Bila panel tidak dapat dilihat maka anda dapat menampilkannya dengan menu *View Object Explorer.* Apabila anda melakukan ekspansi dari sebuah cabang maka sebuah struktur logika dari sebuah obyek akan muncul. Anda dapat mengklik tanda + pada sebelah kiri untuk melakukan ekspansi cabang. Untuk melakukan koneksi pada sebuah server klik kanan

pada nama servernya kemudian pilih *Connect*, untuk melakukan *Disconnect*, klik kanan dan pilih *Disconnect*.

3. *Query Editor*

   Jendela ini digunakan untuk membuat, melakukan editing perintah – perintah T-SQL dan mengeksekusi perintah tersebut. Jendela ini akan muncul otomatis setiap kali anda melakukan kegiatan yang berhubungan dengan *query*. Apabila anda berminat membuat *Query* baru atau malakukan *editing file query* yang telah ada, maka jendela ini otomatis akan muncul. Anda dapat membuat *File Query With Current Connection* atau *Database Engine Query atau SQL Server Commpact Query* atau memanfaattkan tombol *New Query* pada toolbar (Wahana : 2008: 45 – 49).

II.7.3. Komponen – komponen SQL Server 2008

Ada 5 komponen – kompnen SQL Server 2008 adalah sebagai berikut :

1. *Literal Value*

   Yang termasuk *literal value* adalah huruf (a-z), numerik(0-9), danhexadesimal (0x). Literal Value juga dikenal konstanta. Sebuah *konstanta string* terdiri atas satu atau beberapa karakter yang diapit tanda kutip tunggal (*aposthrop*). Atau tanda petik ganda. Defenso konstanta string sebaiknya digunakan dengan tanda petik tunggal karena tanda petik ganda dalam *query* memiliki fungsi lain.

2. *Delimeter*

   Bahwa sebaiknya menggunakan tanda petik tunggal untuk mengawali dan mengakhiri sebuah konstanta string dari pada tanda petik ganda. Hal ini karena tanda petik ganda juga digunakan sebagai delimeter atau pemisah. Tanda kutip ganda tidak dapat digunakan sebagai pembuka atau penutup dari sebuah konstanta string perintah berikut ini diberikan yaitu *Set Queted Identifier On* Standar perintah tersebut adalah On. Perintah tersebut mengakibatkan tanda petik ganda diatur menjadi *Delimeter Identifier*. *Delimeter Identifier* adalah *identifier* khusus yang memungkinkan reserver word digunakana menjadi *identifier* dan juga membolehan adanya spasi pada nama *database*.

3. Komentar

   Komentar dalam pemograman ataupun *stripting* diperlukan untuk memberikan keterangan singkat tentang ode – kode yang ada dibawahnya. Sehingga sewaktu ada kerusakan, kesalahan, programmmer dapat dengan mudah mengerti apa kegunaan dari kode tersebut. pada SQL Server terdapat dua macam komentar yaitu :

   a. Komentar yang menggunakan tanda /**/. Dengan tanda ini komentar yang anda berikan dapat terdiri atas beberapa baris diawali dengan /* dan diakhiri dengan */.
   b. Komentar yang menggunakan tanda – untuk memberi komentar pada baris yang dimaksud saja. Biasanya digunakan untuk menerangkan *identifier* atau *reserved word.*

4. *Identifier*

   Dalam pemograman T- SQL untuk *query*, *identifier* digunakan untuk melakukan identifikasi database dan obyek – obyeknya seperti tabel dan index. Diidentifiksdi dengan string karakter dengan panjang maksimal 128 karakter. *Identifier* ini dapat terdiri atas berbagai macam huruf, angka dan karakter khusus (@,_#, $). Setiap identifier harus dimulai dengan huruf, atau karakter khusus tidak boleh dengan angka.

5. *Reserved Word*

   Reserved Word atau kata kunci adalah kata yang memiliki arti khusus dan harus dituliskan dengan aturan tertentu. Dalam bahasa T-SQL *reserved word* dan juga memilki banyak fungsi. *Reserved word* tidak dapat digunakan sebagai nama sebuah obyek kecuali obyek tersebut didefinisikan sebagai *delimeted identifier*. (Wahana Komputer ; 2008:84-86).

II.8. *Entity Relationship Diagram* (ERD)

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 :

a. *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.
b. *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.
c. *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, (Wahana Komputer ; 2010 : 30).

II.8.1. Komponen – komponen yang terdapat dalam Entity Relantionship Model

1. Entity

Adalah sesuatu yang dapat dibedakan dalam dunia nyata dimana informasi yang berkaitan dengannya dikumpulkan.

a. Entity Set adalah kumpulan entity yang sejenis.

Simbol yang digunakan untuk entity adalah persegi panjang.

Entity Set dapat berupa :

1. Entity yang bersifat fisik, yaitu entity yang dapat dilihat.

   Contohnya : rumah, kendaraan, mahasiswa, dosen, dan lain – lain.

2. Entity yang bersifat logic atau konsep, yaitu entity yang tidak dapat dilihat. Contohnya : pekerjaan, perusahaan, rencana, mata kuliah dan lain – lain.

   Simbol yang digunkan untuk entity adalah persegi panjang.

Gambar II.7. Entity
Sumber : Linda Marlinda (2004:17)

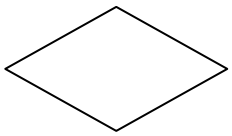
2. Relationship

a. Adalah hubungan yang terjadi antara satu atau lebih entity.

b. Relationship tidak mempunyai keberadaan fisik, kecuali yang mewarisi hubungan antara entity tersebut.

c. Relationship set adalah kumpulan relationship yang sejenis.

   Simbol yang digunakan adalah bentuk belah ketupat, diamond atau rectangle.



Gambar II.8. Relationship
Sumber : Linda Marlinda (2004:18)

3. Attribute

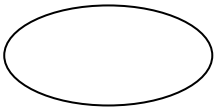
a. Adalah karakteristik dari entity atau relationship yang menyediakan penjelasan detail tentang relationship tersebut.

b. Attribute Value (nilai atribute) adalah suatu data aktual atau informasi yang disimpan di suatu attribute di dalam suatu entity atau relationship.

Terdapat dua jenis attribute yaitu :

1. Indentifer (key), untuk menetukan suatu entity secara unik.
2. Descriptor (nonkey attributte), untuk menetukan karakteristik dari suatu entity yang tidak unik.

Simbol yang digunakan adalah bentuk oval

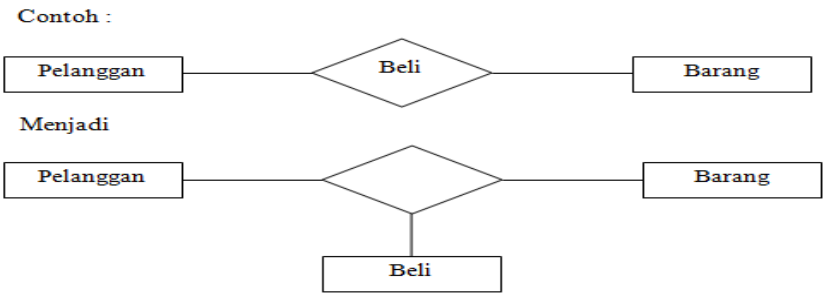


Gambar II.9. Relationship
Sumber : Linda Marlinda (2004:18)

4. Indicator Tipe

a. Indicator tipe associative object

Berfungsi sebagai suatu objek dan suatu relationship.

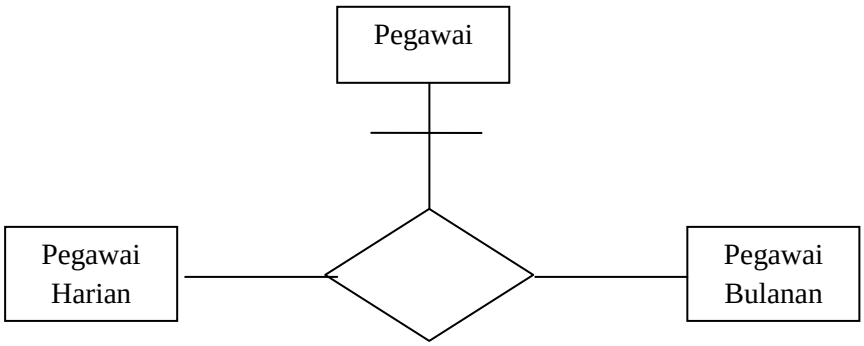


Gambar II.10. Indicator Tipe
Sumber : Linda Marlinda (2004:19)

b. Indicator Tipe Supertipe

Terdiri dari suatu objek dan sub kategori atau lebih yang dihubungkan dan berhubungan dengan relationship yang tidak bernama. (Linda Marlinda. S.Kom 2004:17-19).

Contoh :



Gambar II.11. Indicator Tipe Supertipe
Sumber : Linda Marlinda (2004:19)

II.9. Normalisasi

Normalisasi merupakan cara pendekatan dalam membangun desain logika basis data dan relasional yang tidak secara langsung berkaitan dengan model data, tetapi dengan menerapkan sejumlah aturan dan kriteria standar untuk menghasilkan struktur tabel yang normal.

II.9.1. Bentuk – bentuk Normalisasi

a. Bentuk tidak normal

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi. Data dikumpulkan apa adanya sesuai keadaanya.

b. Bentuk normal tahap pertama (1” Normal Form)

Definisi :

Sebuah table disebut 1NF jika :

- Tidak ada baris yang duplikat dalam tabel tersebut.
- Masing-masing cell bernilai tunggal

Catatan: Permintaan yang menyatakan tidak ada baris yang duplikat dalam sebuah tabel berarti tabel tersebut memiliki sebuah kunci, meskipun kunci tersebut dibuat dari kombinasi lebih dari satu kolom atau bahkan kunci tersebut merupakan kombinasi dari semua kolom.

Berikut ini akan dicontohkan normalisasi dari tabel kuliah yang memiliki atribut : kode_kul, nama_kul, sks, semester, waktu, tempat, dan nama_dos. Tabel kuliah tersebut tidak memenuhi normalisasi pertama, karena terdapat atribut waktu yang tergolong ke dalam atribut bernilai banyak. Agar tabel tersebut dapat memenuhi 1NF, maka solusinya adalah dengan mendekomposisi tabel kuliah menjadi :

- Tabel kuliah (kode_kul, nama_kul, sks, semester, nama_dos).
- Tabel jadwal (kode_kul, waktu, ruang).

c. Bentuk normal tahap kedua (2nd normal form)

Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam primary key memiliki ketergantungan fungsional pada primary key secara utuh.

Sebuah tabel dikatakan tidak memenuhi 2NF, jika ketergantungannya hanya bersifat parsial (hanya tergantung pada sebagian dari primary key). Bentuk normal kedua akan dicontohkan berikut.

Misal tabel nilai terdiri dari atribut kode_kul, nim dan nilai. Jika pada tabel nilai. Misalnya kita tambahkan sebuah atribut yang bersifat redundan, yaitu nama_mhs, maka tabel nilai ini dianggap melanggar 2NF.

Primary key pada tabel nilai adalah (kode_kul, nim).

Penambahan atribut baru (nama_mhs) akan menyebabkan adanya ketergantungan fungsional yang baru yaitu nim > nama_mhs. Karena atribut nama_mhs ini hanya memiliki ketergantungan persial pada primary key secara utuh (hanya tergantungn pada nim, padahal nim hanya bagian dari primary key). Bentuk normal kedua ini dianggap belum memadai karena meninjau sifat ketergantungan atribut terhadap atribut terhadap primary key saja.

d. Bentuk normal tahap ketiga (3rd normal form)

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X -> A, dimana A mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

- X haruslah superkey pada tabel tersebut.
- Atau A merupakan bagian dari primary key pada tabel tersebut.

Misalkan pada tabel mahasiswa, atribut alamat_mhs dipecah kedalam alamat_jalan, alamat_kota dan kode_pos. Bentuk ini tidak memenuhi 3NF,

karena terdapat ketergantungan fungsional baru yang muncul pada tabel tersebut, yaitu :

- alamat_jalan, nama_kota – kode_pos

Dalam hal ini (alamat_jalan, nama_kota) bukan superkey sementara kode_pos juga bukan bagian dari primary key pada tabel mahasiswa. Jika tabel mahasiswa didekomposisi menjadi tabel mahasiswa dan tabel alamat, maka telah memenuhi 3NF. Hal itu dapat dibuktikan dengan memeriksa dua ketergantungan fungsional pada tabel alamat tersebut, yaitu :

- alamat_jalan, nama_kota – kode_pos
- kode_pos – nama_kota

Ketergantungan fungsional yang pertama tidak melanggar 3NF, karena (alamat_jalan, nama_kota) merupakan superkey (sekaligus sebagai primary key) dari tabel alamat tersebut. Demikian juga dengan ketergantungan fungsional yang kedua meskipun (kode_pos) bukan merupakan superkey, tetapi nama_kota merupakan bagian dari primary key dari tabel alamat. Karena telah memenuhi 3NF, maka tabel tersebut tidak perlu di-dekomposisi lagi.

e. Bentuk Normal Tahap Keempat dan Kelima

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk Normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan

pengembangan dari ketergantungan fungsional. Adapun bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

f. Boyce Code Normal Form (BCNF)

- Memenuhi 1st NF
- Relasi harus bergantung fungsi pada atribut superkey (Kusrini, M.Kom ; 2010 : 41-43).

II.10. *Unified Modeling Language* (UML)

UML singkatan dari *Unified Modelling Langguage* yang berarti bahasa pemodelan standart. (Chonoles; 2003 : 6) 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 lainya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorentasikan 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 mensuport 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 Herlawati ; 2011 ; 6).

II.10.1. *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 dikelompokan 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 umu dijumpai pada pemodelan sistem berorentasi 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 adal;ah 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 terutam 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 berserta 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.

1. 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.

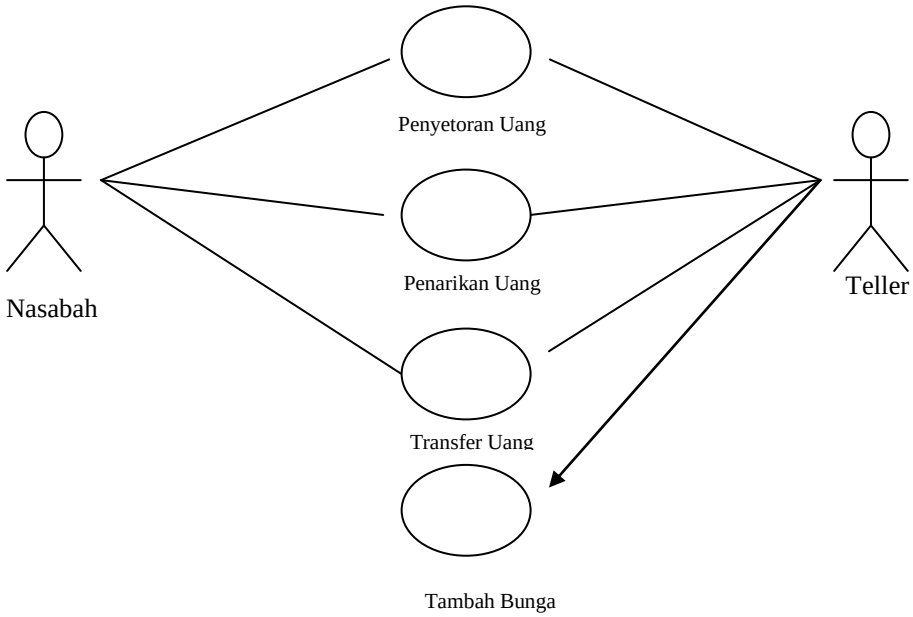
Komponen pembentuk diagram *use case* adalah :

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.

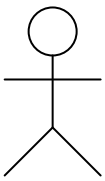
Gambar di bawah ini merupakan salah satu contoh bentuk diagram *use case.*



Gambar II.12. Diagram *Use Case*
Sumber : Probowo Pudjo Widodo (2011:17)

1. Aktor

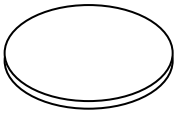
Menurut Chonoles (2003 :17) menyarankan sebelum mebuat use case dan menentukan aktornya, agar mengidentifikasi siapa saja pihak yang terlibat dalam sistem kita. Pihak yang terlibat biasanya dinamakan *stakeholder*.



Gambar II.13. Aktor
Sumber : Probowo Pudjo Widodo (2011:17)

2. *Use Case*

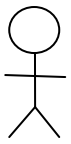
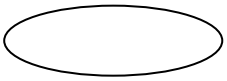
Menurut Pilone (2005 : 21) *use case* menggambarkan fungsi tertentu dalam suatu sistem berupa komponen kejadian atau kelas. Sedangkan menurut Whitten (2004 : 258) mengartikan *use case* sebagai urutan langkah-langkah yang secara tindakan saling terkait (skenario) baik terotomatisasi maupun secara manual,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

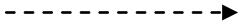
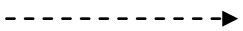


Gambar II.14. Simbol *Use Case*
Sumber : Probowo Pudjo Widodo (2011:22)

Berikut Simbol-simbol dalam *Use Case*, yaitu :

Tabel II.1 Tabel *Simbol-simbol Use case*

NAMA	KETERANGAN	SIMBOL
Actor	Seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang kita kembangkan	
Use Case	Peringkat Tertinggi dari fungsional yang dimiliki sistem.	

Relasi Asosiasi	Relasi yang terjadi antara aktor dengan use case biasanya berupa asosiasi.	
Include Relationship	Relasi cakupan memungkinkansuatu use case untuk menggunakan fungsionalitas yang disediakan oleh use case lainnya.	<<Include> 
Extends Relationship	Memungkinkan suatu use case memiliki kemungkinan untuk memperluas fungsional yang disediakan use case yang lainnya.	<<Extends 

Sumber : Adi Nugroho(2005:49)

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu Chonoles (2003:22-23) menawarkan cara untuk menghasilkan *use case* yang baik yakni :

a. Pilihlah nama yang baik

Use case adalah sebuah *behaviour* (prilaku), 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 membutukan *use case* yang membatalkan tujuan, misalnya pada us*e case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

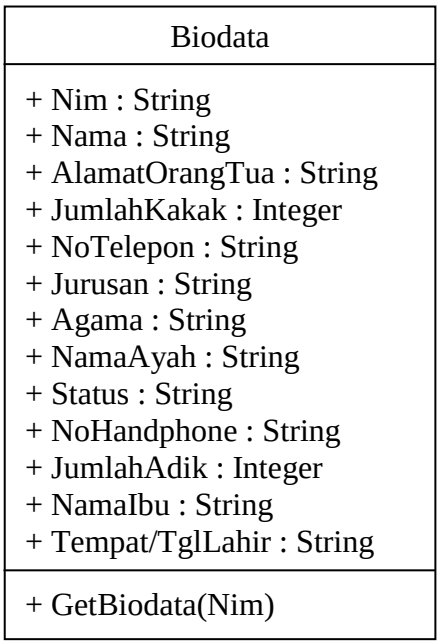
e. Batasi use case hingga satu perilaku saja.

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.

2. Diagram Kelas (*Class Diagram*)

Diagram kelas mempunyai dua jenis yaitu *domain class diagram* dan *design class diagram*. Fokus *domain class diagram* adalah pada sesuatu dalam lingkungan kerja pengguna, bukan pada *class* perangkat lunak yang nantinya akan anda rancang. Sedangkan *design class diagram* tujuannya adalah untuk mendokumentasikan dan menggambarkan kelas-kelas dalam pemrograman yang nantinya akan dibangun.

Gambar II.15. Notasi *Domain Diagram Class*
Sumber : E. Triandini dan G. Suardika (2012 : 49 -50)



Gambar II.16. Notasi *Design Diagram Class*
Sumber : E. Triandini dan G. Suardika (2012 : 49 -50)

3. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistemdari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukan 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 (Probowo Pudji Widodo ;2011 : 143-145).

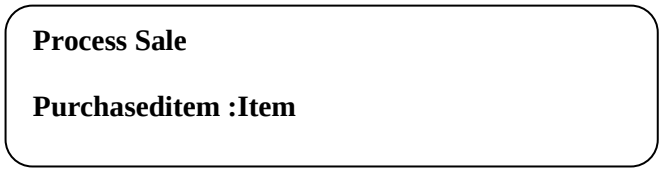
Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi nelakukan langka sekali saja tidak boleh dipecah menjadi beberapa langkah-langkah lagi. Contoh aksinya yaitu :

a. Fungsi Matematika

b. Pemanggilan Perilaku

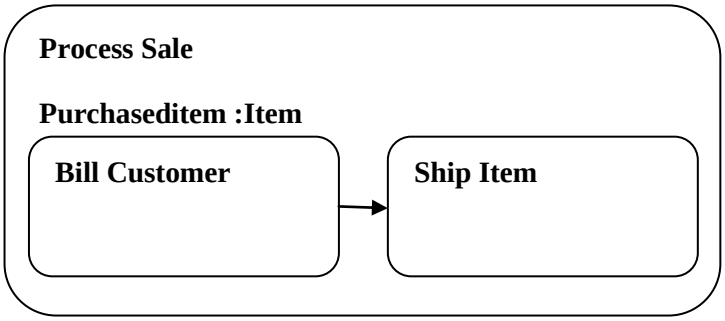
c. Pemrosesan Data

Ketika kita menggunakan diagram aktivitas untuk memodelkan perilaku suatu *classfier* dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classfier*, 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.



Gambar II.17. Aktivitas serderhana tanpa rincian
Sumber : Probowo Pudjo Widodo (2011:145)

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



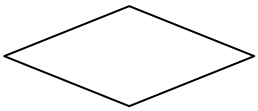
Gambar II.18. Aktivitas dengan detail rincian
Sumber : Probowo Pudjo Widodo (2011:145)

Berikut komponen-komponen dalam *activity diagram*, yaitu :

a. Action State, adalah langkah-langkah dalam sebuah *activity*. *Action* bisa terjadi saat memasuki *activity*, meninggalkan *activity*, atau pada *event* yang spesifik.

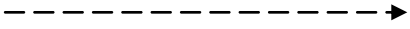
Gambar II.19. *Action State*
Sumber : Probowo Pudjo Widodo (2011:145)

b. Decision, menunjukkan dimana sebuah keputusan perlu dibuat dalam aliran kerja.



Gambar II.20. *Decision*
Sumber : Probowo Pudjo Widodo (2011:145)

c. Transition, menunjukkan bagaimana aliran kerja itu berjalan dari satu aktivitas ke aktivitas lainnya.



Gambar II.21. *Transition*
Sumber : Probowo Pudjo Widodo (2011:145)

d. Syncronization, menunjukkan dua atau lebih langkah dalam aliran kerja berjalan secara serentak *a Fork and Join.*

Gambar II.22. *Syncronization*
Sumber : Probowo Pudjo Widodo (2011:145)

e. Swimlane, menunjukkan siapa yang bertanggung jawab melakukan aktivitas dalam suatu diagram.

Gambar II.23. *Swimlane*
Sumber : Probowo Pudjo Widodo (2011:145)

f. Start State, memperlihatkan dimana aliran kerja berawal.

Gambar II.24. *Start State*
,Sumber : Adi Nugroho(2005:66)

g. End State, memperlihatkan dimana aliran kerja berakhir.

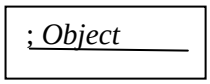
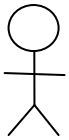
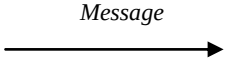
Gambar II.25. *End State*
Sumber : Adi Nugroho(2005:66)

4. Sequence Diagram

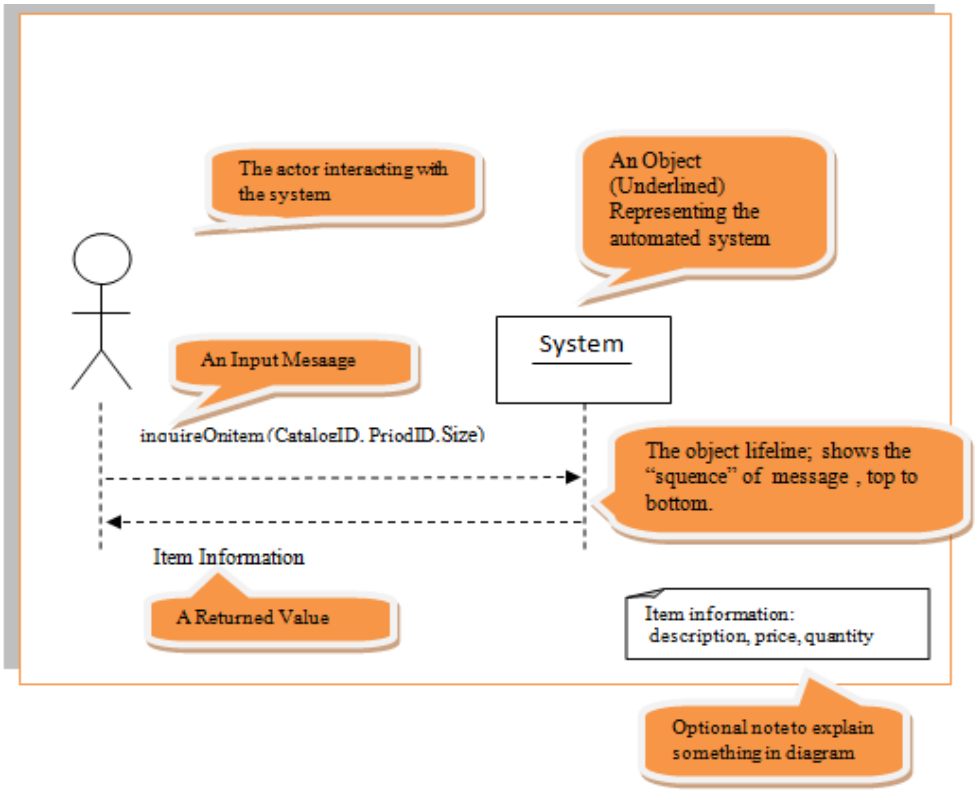
Menurut John Satzinger, 2010, dalam buku *System Analysis and Design in a Changing World*, "System Sequence Diagram (SSD) adalah diagram yang digunakan untuk mendefinisikan input dan output serta urutan interaksi antara pengguna dan sistem untuk sebuah use case (E. Triandini dan G. Suardika ; 2012 : 71).

Berikut adalah notasi-notasinya :

Tabel II.2. Notasi Sequence Diagram

Object	*Object* merupakan instance dari sebuah *class* dan dituliskan tersusun secara horizontal. Digambarkan sebagia sebuah *class* (kotak) dengan nama obyek di dalamnya yan g diawa dengan sebuah titik koma.	
Actor	*Actor* juga dapat berkomunikasi dengan objek, maka *actor* juga dapat diurutkan sebagai kolom. Simbol *Actor* sama dengan simbol pada *Actor Use Case Diagram.*	
Lifeline	*Lifeline* mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk *Lifeline* adalah garis putus-putus vertikal yang ditarik dari sebuah obyek.	
Activation	*Activation* dinotasikan sebagai kotak segi empat yang digambar pada sebuah *Lifeline. Activation* mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.	
Message	*Message* digambarkan dengan anak panah horizontal antara *activation. Mesaage* mengindikasikan komunikasi antara *object-object.*	

Sumber : Adi Nugroho (2005 : 92)



Gambar II.26. Notasi Sequence Diagram
Sumber : Evi Triandini dan Gede Suardika (2012 : 71)