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 menekan 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 : 10)

II.1.1. Karakteristik Sistem

Model umum sebuah sistem adalah input, proses, dan output. Hal ini merupakan konsep sebuah sistem yang sangat sederhana sebab sebuah sistem dapat mempunyai beberapa masukan dan keluaran. Selain itu sebuah sistem memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem. Adapun karakteristik yang dimaksud adalah sebagai berikut:

1. Komponen Sistem (*Component*)

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, artinya saling berkerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat dari sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar, yang disebut “supra sistem”.

2. Batas Sistem *(Boundary)*

Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem yang lain atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan Luar Sistem *(Environment)*

Bentuk apapun yang ada di luar lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut operasi lingkungan luar sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan dan dapat juga bersifat

merugikan sistem tersebut. Lingkungan yang menguntungkan merupakan bagian sistem tersebut. Dengan demikian, lingkungan luar tersebut harus dijaga dan dipelihara. Lingkungan luar yang merugikan harus dikendalikan. Kalau tidak maka akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Sistem *(Interface)*

Media yang menghubungkan sistem dengan subsistem lainnya disebut penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari subsistem lain. Bentuk keluaran dari satu subsistem akan menjadi masukan untuk subsistem lain melalui penghubung tersebut. Dengan demikian dapat terjadi suatu integrasi sistem untuk membentuk satu kesatuan.

5. Masukan Sistem *(Input)*

Energi yang dimasukan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input*). Contoh di dalam suatu sistem unit komputer. "program" adalah *maintenance* input yang digunakan untuk mengoperasikan komputernya dan "data" adalah *signal input* untuk diolah menjadi informasi.

6. Keluaran Sistem (*Output*)

Yaitu hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain. Contoh, sistem informasi. Keluaran yang dihasilkan adalah informasi. Informasi ini dapat digunakan sebagai masukan untuk pengambil keputusan atau hal-hal lain yang menjadi *input* bagi subsistem lain.

7. Pengolah Sistem (*Proses*).

Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Contoh, sistem akuntansi. Sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem (*Objective*)

Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat *deterministik*. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan (Tata Sutabri; 2012 :20-21).

II.1.2. Daur Hidup Sistem

Siklus hidup sistem (*system life cycle*) adalah merupakan proses evolusioner yang diikuti dalam menerapkan sistem atau subsistem informasi komputer. Siklus hidup sistem terdiri dari serangkaian tugas yang erat mengikuti langkah-langkah pendekatan sistem sistem karena tugas-tugas tersebut mengkuti pola yang teratur dan dilakukan secara *top down*. Siklus hidup sistem sering disebut sebagai pendekatan air terjun (*waterfall approach*) bagi pembangunan dan pengembangan sistem.

Pembangunan sistem hanyalah salah satu dari rangkaian daur hidup suatu sistem. Meskipun demikian, proses ini merupakan aspek yang sangat penting. Kita akan melihat beberapa fase/tahapan dari daur hidup suatu sistem.

1. Mengenali Adanya Kebutuhan.

Sebelum segala sesuatunya terjadi, timbul suatu kebutuhan atau problema yang harus dapat dikenali sebagaimana adanya. Kebutuhan dapat terjadi

sebagai hasil perkembangan dari organisasi dan volume yang meningkat melebihi kapasitas dari sistem yang ada. Semua kebutuhan ini harus dapat didefenisikan dengan jelas. Tanpa adanya kejelasan dari kebutuhan yang ada, pembangunan sistem akan kehilangan arah dan efektifitasnya.

2. Pembangunan Sistem

Suatu proses atau seperangkat prosedur yang harus diikuti untuk menganalisis kebutuhan yang timbul dan membangun suatu sistem untuk dapat memenuhi kebutuhan tersebut.

3. Pemasangan Sistem

Setelah tahap pembangunan sistem selesai, sistem kemudian akan dioperasikan. Pemasangan sistem merupakan tahap yang penting pula dalam daur hidup sistem. Peralihan dari tahap pembangunan menuju tahap operasional terjadi pemasangan sistem yang sebenarnya, yang merupakan langkah akhir dari suatu pembangunan.

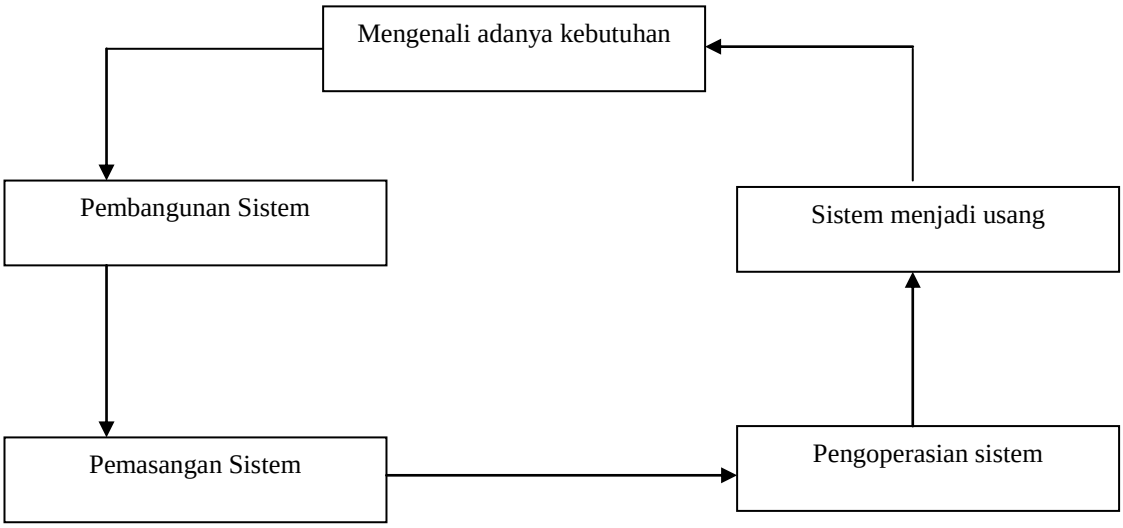
4. Pengoperasian Sistem

Program-program komputer dan prosedur-prosedur pengoperasian yang membentuk suatu sistem informasi semuanya bersifat statis. Sedangkan organisasi ditunjang oleh sistem informasi tadi, ia selalu mengalami perubahan-perubahan itu karena pertumbuhan kegiatan bisnis, perubahan pengaturan, dan kebijaksanaan ataupun kemajuan teknologi. Untuk mengatasi perubahan-perubahan tersebut, sistem harus diperbaiki atau diperbaharui.

5. Sistem Menjadi Usang

Kadang perubahan yang terjadi begitu drastis, sehingga tidak dapat diatasi hanya dengan melakukan perbaikan-perbaikan pada sistem yang berjalan. Tibalah saatnya secara ekonomis dan teknis sistem yang ada sudah tidak layak lagi untuk dioperasikan dan sistem yang baru perlu dibangun untuk menggantikannya.

Sistem informasi kemudian akan melanjutkan daur hidupnya. Sistem dibangun untuk memenuhi kebutuhan yang muncul. Sistem beradaptasi terhadap perubahan-perubahan lingkungannya dinamis. Sampailah pada kondisi dimana sistem tersebut tidak dapat lagi beradaptasi dengan perubahan-perubahan yang ada atau secara ekonomis tidak layak lagi untuk dioperasikan. Sistem yang baru kemudian dibangun untuk menggantikanya. Untuk dapat menggambarkan daur hidup sistem ini, lihat pada gambar II.1. sebagai berikut (Tata Sutabri; 2012 :26-28).



Gambar II.1 : Daur Hidup Sistem

(Sumber : Tata Sutabri ; 2012 : 29)

II.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 : 29).

II.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 :46)

II.3.1. Komponen Dan Jenis Sistem Informasi.

Sistem informasi terdiri dari komponen-komponen yang disebut blok bangunan (*building block*), yang terdiri dari blok masukan, blok model, blok keluaran, blok teknologi, blok basis data dan blok kendali. Sebagai suatu sistem, keenam blok tersebut masing-masing saling berinteraksi satu dengan yang lain membentuk satu kesatuan untuk mencapai sasaran.

1. Blok Masukan (*Input Block*)

Input mewakili data yang masuk ke dalam sistem informasi. *Input* di sini termasuk metode dan media untuk menangkap data yang akan dimasukan, yang dapat berupa dokumen-dokumen dasar.

2. Blok Model (*Model Block*)

Blok ini terdiri dari kombinasi prosedur, logika, dan model matematik yang akan memanipulasi data *input* dan data yang tersimpan di basis data dengan cara yang sudah tertentu untuk menghasilkan keluaran yang diinginkan.

3. Blok Keluaran (*Output Block*)

Produk dari sistem informasi adalah keluaran yang merupakan informasi yang berkualitas dan dokumentasi yang berguna untuk semua tingkatan manajemen serta pemakai sistem.

4. Blok Teknologi (*Technology Block*)

Teknologi merupakan "*tool box"* dalam sistem informasi. Teknologi digunakan untuk menerima *input*, menjalankan model, menyimpan dan mengakses data, menghasilkan dan mengirimkan keluaran dan membantu pengendalian dari sistem keseluruhan. Teknologi sistem terdiri dari 3 (tiga) bagian yaitu teknisi teknologi (*brainware*), perangakat lunak (*software*), dan perangkat keras (*hardware*).

5. Blok Basis Data (*Database Block*)

Basis data (*database*) merupakan kumpulan data yang saling berkaitan dan berhubungan satu dengan lainnya, tersimpan di perangkat keras komputer dan menggunakan perangkat lunak untuk memanipulasinya. Data perlu disimpan di basis data untuk keperluan penyediaan informasi lebih lanjut. Data di dalam basis data perlu diorganisasikan sedemikian rupa supaya informasi yang dihasilkan berkualitas. Organisasi basis data yang baik juga berguna untuk efesiensi

kapasitas penyimpanannya. Basis data diakses atau dimanipulasi menggunakan perangkat lunak paket yang disebut DBMS (*database management system*).

6. Blok Kendali (*Control Block*)

Banyak hal yang dapat merusak sistem informasi, seperti bencana alam, api, temperature, air, debu, kecurangan-kecurangan, kegagalan-kegagalan sistem itu sendiri, ketidakefesienan, sabotase, dan lain sebagainya. Beberapa pengendalian perlu dirancang dan diterapkan untuk meyakinkan bahwa hal-hal yang dapat merusak sistem dapat dicegah atau bila terlanjur terjadi kesalahan-kesalahan dapat langsung cepat diatasi (Tata Sutabri; 2012 :46-47).

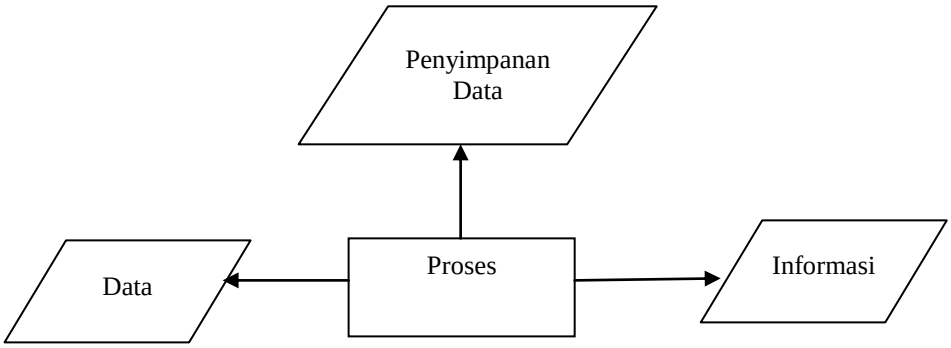
II.4. Sistem Informasi Akuntansi

Sistem Informasi Akuntansi adalah Sistem yang bertujuan untuk mengumpulkan dan memproses data serta melaporkan informasi yang berkaitan dengan transaksi keuangan. Misalnya, salah satu input dari Sistem Informasi Akuntansi pada sebuah toko baju, seperti pada contoh sebelumnya, adalah transaksi penjualan (Anastasia Diana , Lilis Setiawati;2011:4).

II.5. Data

Mengenai pengertian data, lebih jelas apa yang didefenisikan oleh Drs. Jhon J. Longkutoy dalam bukunya " Pengenalan Komputer" sebagai berikut : isitilah data adalah suatu istilah majemuk yang berarti fakta atau bagian dari fakta yang mengandung arti yang dihubungkan dengan kenyataan simbol-simbol, gambar-gambar, angka-angka, huruf-huruf, atau simbol-simbol yang

menunjukkan suatu ide, objek, kondisi, atau situasi dan lain-lain (Tata Sutabri, 2012 : 2).



Gambar II.2 : Pemrosesan Data

(Sumber : Tata Sutabri 2012 :2)

II.5.1. Pengolahan Data

Data merupakan bahan mentah untuk diolah, yang hasilnya kemudian menjadi informasi. Dengan kata lain, data yang telah diperoleh harus diukur dan dinilai baik buruknya, berguna atau tidak dalam hubungannya dengan tujuan yang akan dicapai. Pengolahan data terdiri dari kegiatan-kegiatan penyimpanan data dan penanganan data. Untuk lebih jelasnya akan diuraikan dibawah ini (Tata Sutabri; 2012 :6).

1. Penyimpanan Data (Data *Storage*)

Penyimpanan data meliputi pekerjaan pengumpulan (*filing*), pencarian (*searching*), dan pemeliharaan (*maintenance*). Data disimpan dalam suatu tempat yang lazim dinamakan "*file". File* dapat berbentuk map, *ordner, disket, tape, hard disk,* dan lain sebagainya. Sebelum disimpan, suatu data diberi kode menurut jenis kepentingannya. Pengaturan dilakukan sedemikian rupa sehingga mudah mencarinya. Pengkodean memegang peranan penting. Kode yang salah data akan mengakibatkan data masuk ke dalam *file* juga salah, yang selanjutnya akan

mengakibatkan kesulitan pencarian data tersebut apabila diperlukan. Jadi *file* diartikan suatu susunan data yang terbentuk dari sejumlah catatan (*record)* yang berhubungan satu sama lain (sejenis) mengenai suatu bidang dalam suatu unit usaha.

Sistem yang umum dalam penyimpanan data (*filing*) ialah berdasarkan lembaga, perorangan, produksi, atau lain-lainya, tergantung dari sifat organisasi yang bersangkutan. Kadang-kadang dijumpai kesulitan apabila menghadapi suatu data dalam bentuk surat misalnya yang menyangkut ketiga klasifikasi tadi. Metode yang terbaik adalah "referensi silang" (*cross reference*) antara *file* yang satu dengan *file* lain. Untuk memperoleh kemudahan dalam pencarian data (*searching*) di dalam *file*, maka *file* dibagi menjadi 2 (dua) jenis yaitu:

a. *File* Induk (*Master File*)

File induk ini berisi data-data permanen yang biasanya hanya dibentuk satu kali saja dan kemudian digunakan untuk pengolahan data selanjutnya.

Contoh : *file* kepegawaian, *file* gaji

b. *File* Transaksi (*Detail File*).

File transaksi berisi data-data temporer untuk suatu periode untuk suatu bidang kegiatan atau suatu periode yang dihubungkan dengan suatu bidang kegiatan.

Contoh : *file* lembur perminggu, *file* mutasi harian.

Pemeliharaan *file* (*file maintenance*) juga meliputi "peremajaan data" (data *updating*), yaitu kegiatan menambah catatan baru pada suatu data, mengadakan

perbaikan, dan lain sebagainya. Misalnya, dalam hubungan dengan *file* kepegawaian, sudah tentu sebuah organisasi, entah itu perusahaan atau jawatan, akan menambah pegawainya. Sementara itu, ada pula pegawai yang pensiun atau berhenti bekerja atau putus hubungan dengan organisasi. Dengan demikian, data mengenai pegawai yang bersangkutan akan dikeluarkan dari *file* tersebut. Tidak jarang pula harus dilakukan perubahan data terhadap data seorang pegawai, misalnya kenaikan pangkat, kenaikan gaji berkala, menikah, pindah alamat, dan lain sebagainya.

2. Penanganan Data (Data *Handling*).

Penanganan data meliputi berbagai kegiatan, seperti pemeriksaan (*verifying*), perbandingan (*comparing*), pemilihan (*sorting*), peringkasan (*extracting*), dan penggunaan (*manipulating*). Pemeriksaan data mencakup pengecekan data yang muncul pada berbagai daftar yang berkaitan atau yang datang dari berbagai sumber, untuk mengetahui berbagai sumber dan perbedaan atau ketidaksesuaian. Pemeriksaan ini dilakukan dengan kegiatan pemeliharaan *file* (*file maintenance*).

Pemilihan atau sorting dalam rangka kegiatan penanganan data mencakup pengaturan ke dalam suatu urutan yang teratur, misalnya daftar pegawai menurut pangkatnya, dari pangkat yang tertinggi sampai yang terendah atau daftar pelanggan dengan menyusun namanya menurut abjad dan lain sebagainya. Peringkasan merupakan kegiatan lain dalam penanganan data. Ini mencakup keterangan pilihan, misalnya daftar pegawai yang telah mengabdikan

dirinya kepada organisasi/perusahaan lebih dari 10 tahun atau daftar yang memesan beberapa hasil produksi sekaligus dan lain-lain.

Penggunaan data atau informasi "*data manipulation*" merupakan kegiatan untuk menghasilkan informasi. Kegiatan ini meliputi kompilasi tabel-tabel, stastik, ramalan mengenai perkembangan, dan lain sebagainya. Tujuan manipulasi ini adalah menyajikan informasi yang memadai mengenai apa yang terjadi pada waktu lampau guna menunjang manajemen, terutama membantu menyelidiki alternatif kegiatan mendatang. Jadi, hasil pengolahan data itu merupakan data untuk disimpan bagi penggunaan di waktu yang akan datang, yakni informasi yang akan disampaikan kepada yang memerlukan atau mengambil keputusan mengenai suatu hal (Tata Sutabri; 2005 :6-8).

II.6. Konsep Penjualan

Banyak organisasi mengikuti konsep penjualan yang menganggap konsumen tidak akan membeli cukup produknya untuk barang-barang yang tidak dicari, seperti ensiklopedi dan asuransi. Konsep penjualan juga dipraktikkan pada organisasi nirlaba. Penjualan kredit dilaksanakan oleh organisasi dalam hal ini adalah perusahaan dengan cara mengirimkan barang hasil produksi kepada konsumen sesuai order pembeli yang disepakati sebelumnya.

Kebanyakan perusahaan menerapkan konsep penjualan ketika kelebihan kapasitas. Tujuan mereka menjual apa yang mereka hasilkan dan bukan menghasilkan apa yang diminta pasar. Pemasaran merupakan upaya getol penjualan dengan penuh resiko. Dipusatkan kepada terciptanya transaksi penjualan dan bukan hanya membangun hubungan yang menguntungkan dengan konsumen. Diasumsikan konsumen yang dibujuk untuk membeli menyukai produknya atau kalau mereka tidak suka mereka melupakan kekecewaannya dan nantinya akan membeli lagi.

Tentu saja itu adalah asumsi yang lemah tentang konsumen. Beberapa studi menunjukkan bahwa pelanggan tidak puas sehingga bakal membeli lagi. Yang lebih buruk lagi, kalau seorang pelanggan yang puas bercerita hanya kepada tiga orang, maka pelanggan yang tidak puas bercerita kepada sepuluh orang mengenai pengalaman buruknya (Nembah F. Hartimbul Ginting; 2011 : 27)

II.7. Metode Flat Rate

Sistem Flat rate merupakan bunga kredit yang dikenakan kepada debitur setiap bulan jumlahnya tetap, walaupun jumlah pokok kredit telah menurun karena telah diangsur setiap bulannya.

Rumusnya adalah :

$$A = \frac{P + i(Pxn)}{n}$$

Keterangan :

A = Angsuruan pokok kredit + bunga kredit (dimana A jumlahnya tetap meskipun pokoknya telah diangsur).

P = Jumlah Pokok kredit awal/plafond sebelum diangsur.

n = Jangka waktu kredit (dalam bulan) (Linda Iresa, dkk : 2011 : 3-4)

II.8. Basis Data (*Database*)

Basis data menurut Stephen dan Plew adalah (2000) adalah mekanisme yang digunakan untuk menyimpan informasi atau data.

Kemudian Silberchatz, dkk (2002) mendefenisikan 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 efesien.

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).

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. Mengintergrasikan 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.

2. Kerugian DBMS (*Database Management System*)

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 oraganisasi 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 mengunakan komputer sebagai suatu sistem informasi bagi para spesialis informasi dan pengguna (Janner Simarmata, dkk; 2010 : 8-9).

II.8.1. *Entity Relationship Diargam* (ERD)

Entity Relationship (ER) 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 objek dalam dunia nyata yang dapat dibedakan dari objek lain. Sebagai contoh, masing-masing mahasiswa adalah entitas dan mata kuliah dapat dianggap sebagai entitas.

Entitas digambarkan dalam basis data dengan kumpulan atribut. Misalnya atribut nim, nama, alamat, dan kota bisa menggambarkan data mahasiswa tertentu dalam

suatu universitas. Atribut-atribut membentuk entitas mahasiswa. Demikian pula, atribut kodeMK, namaMK, dan SKS mendeskripsikan mata kuliah.

Atribut NIM digunakan sebagai untuk mengidentifikasikan mahasiswa secara unik karena dimungkinkan terdapat dua mahasiswa dengan nama, alamat, dan kota yang sama. Pengenal unik harus diberikan pada masing-masing mahasiswa.

Relasi adalah hubungan antara beberapa entitas. Sebagai contoh relasi menghubungkan mahasiswa dengan mata kuliah yang diambilnya. Kumpulan semua entitas bertipe sama disebut kumpulan entitas (*entitas sel*), sedangkan kumpulan semua relasi bertipe sama disebut dengan kumpulan relasi (*relationship sel*).

Struktur logis (skema database) dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut pada dilihat pada tabel II.1.

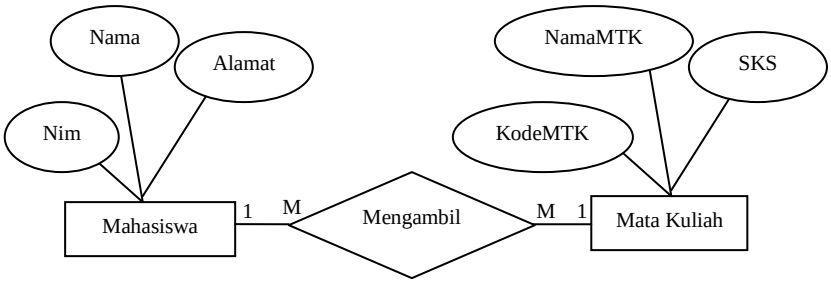
Tabel II.1. Komponen-Komponen Diagram ER

Entitas	Persegi Panjang mewakili kumpulan entitas
Atribut	Elips Mewakili Atribut
Relasi	Belah Ketupat Mewakili Relasi
	Garis Menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi

(Sumber : Janner Simarmata, dkk ; 2010 : 60)

Masing-masing komponen diberi nama entitas atau relasi yang diwakilinya.

Sebagai ilustrasinya bayangkan anda mengambil bagian sistem basis data universitas yang terdiri dari mahasiswa dan mata kuliah. gambar II.3. menunjukkan diagram ER dari contoh. Diagram menunjukkan bahwa ada dua kumpulan entitas yaitu mahasiswa dan mata kuliah dan bahwa relasi mengambil contoh mahasiswa dan mata kuliah (Janner Simarmata; 2010 : 59-60).



Gambar II.3 : Diagram ER

(Sumber : Janner Simarmata, dkk ; 2010 : 60)

II.8.2. 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 Nornal 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.2. menunjukkan tabel pemasok dalam 1 NF

Tabel II.2. Normalisasi Pertama Pemasok

P#	Status	Kota	B#	Qty
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 :80)

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.3. Tabel Bentuk Normal Kedua (2NF).

Pemasok2

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

Barang

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 : 82)

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 dapat digambarkan dengan menunjukkan ketergantungan fungsional pada pemasok2, yaitu :

Pemasok2. p# ⟶ Pemasok2, status

Pemasok2. p# ⟶ Pemasok2, kota

Pemasok2. kota ⟶ 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.4 menunjukkan hasilnya

Tabel II.4. Tabel Bentuk Normal Ketiga (3 NF)

PEMASOK_KOTA

P#	Kota
P1	Yogyakarta
P2	Medan
P3	Medan
P4	Yogyakarta
P5	Bandung

KOTA_STATUS

Kota	Status
Yogyakarta	20
Medan	10
Bandung	30
Semarang	40

(Sumber : Janner Simarmata, dkk ; 2010 : 83)

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 multivalue merupakan ketergantungan fungsional.

Bentuk normal keempat (4 NF) didasarkan pada konsep ketergantungan multivalue (MVD). Sebuah ketergantungan multivalue terjadi ketika dalam sebuah tabel relasional yang mengandung setidaknya tiga kolom, satu kolom mempunyai banyak baris bernilai sama, tetapi kolom lain bernilai berebeda.

Defenisi secara formal diberikan oleh CJ. Date, yaitu :

Misalnya, ada sebuah tabel relasional R dengan kolom A, B dan C, Maka R.A $\longrightarrow$ 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 $\longrightarrow$ R.B dipenuhi jika dan hanya jika R.A $\longrightarrow$ 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 deskomposisi 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 dideskomposisi lagi (Janner Simarmata; 2012 : 77 - 86)

II.9. *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 Dan Herlawati; 2011 : 6-7).

II.9.1. 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 umum 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 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 memberi 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. Pada *UML* dimungkinkan kita menggunakan diagram-diagram lainya misalnya *Data Flow Diagram, Entity Relationship Diagram* dan sebagainya (Prabowo Pudjo Widodo,Dan Herlawati; 2011 : 10-12).

II.9.3. Diagram Use Case (*Use Case Diagram*)

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

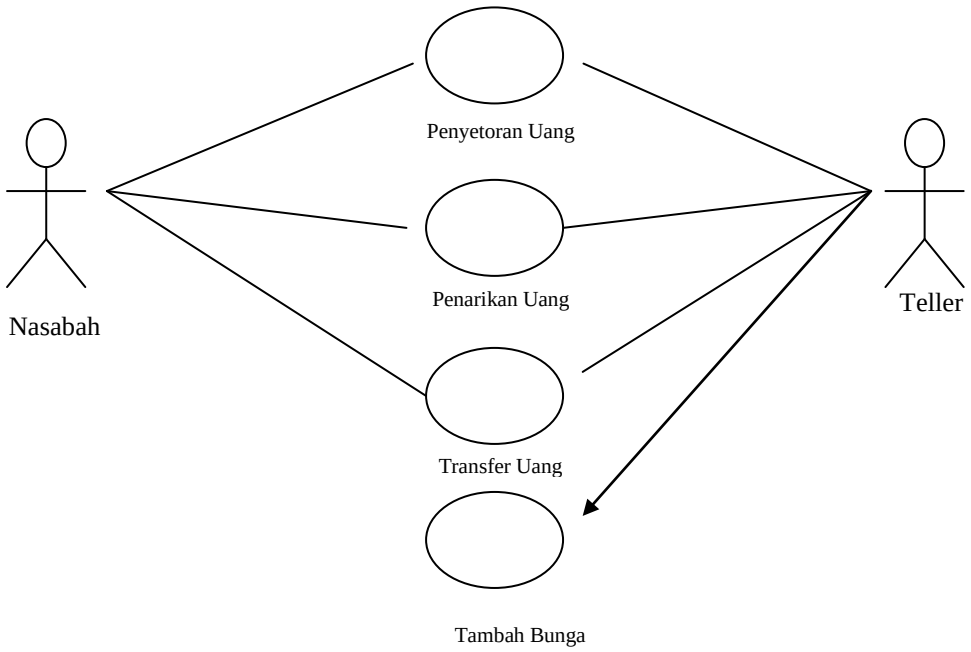
Use Case menggambarkan *external view* dari sistem yang akan kita buat modelnya. Menurut Pooley (2005:15) mengatakan bahwa model *use case* dapat

dijabarkan dalam diagram, tetapi yang perlu diingat, diagram tidak 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 : 16-17).

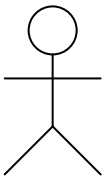


Gambar II.4 : Diagram *Use Case*

(Sumber : Prabowo Pudjo Widodo Dan Herlawati ; 2011 : 17)

2. 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*.

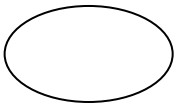


Gambar II.5 : Aktor

(Sumber : Prabowo Pudjo Widodo Dan Herlawati ; 2011 : 17)

3. *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.6 : Simbol *Use Case*

(Sumber : Prabowo Pudjo Widodo Dan Herlawati ; 2011 : 22)

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 u*se case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

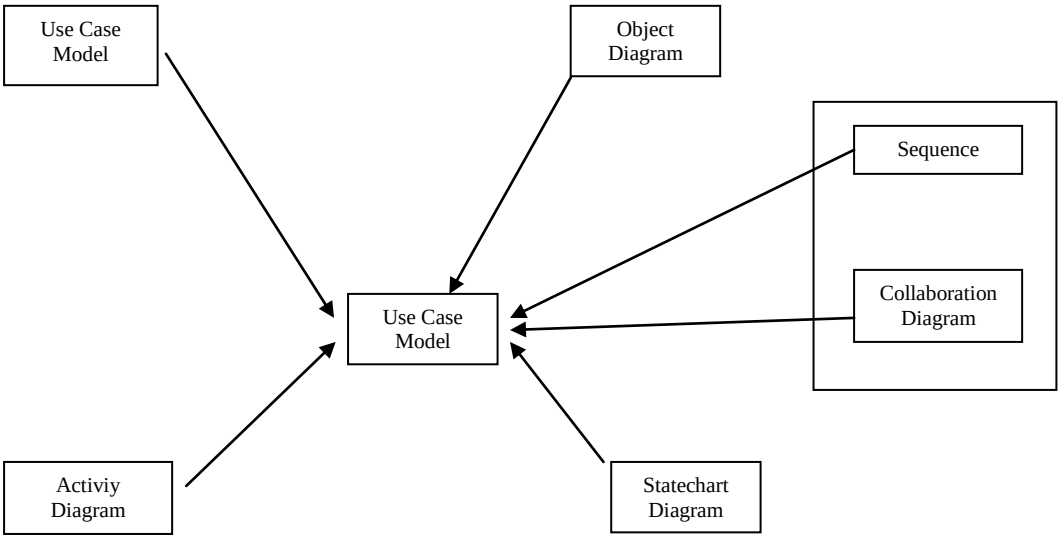
e. Batasi Use Case Hingga Satu Perilaku Saja.

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.

4. 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 : 37).



Gambar II.7 : Hubungan Diagram Kelas Dengan Diagram *UML* lainya

(Sumber : Prabowo Pudjo Widodo Dan Herlawati ; 2011 : 38)

5. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur system dari 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

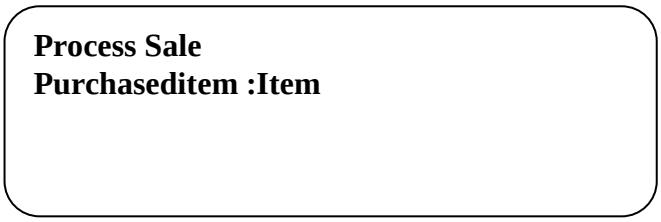
software, diagram aktivitas merepresentasikan pemanggilan suatu fungsi tertentu misalnya *call.* Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian diluar seperti pemesanan atau kejadian-kejadian internal misalnya penggajian tiap jumat sore (Probowo Pudji Widodo, Dan Herlawati ;2011 : 143-145).

Aktivitas merupakan kumpulan aksi-aksi. Aksi-aksi nelakukan langkah 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.

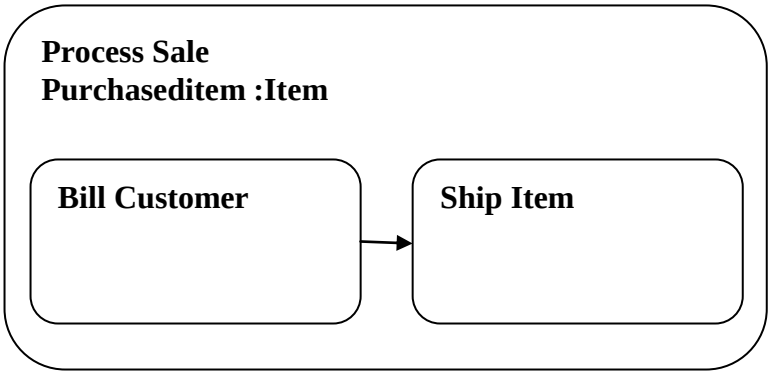
Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



Gambar II.8 : Aktivitas serderhana tanpa rincian

(Sumber : Prabowo Pudjo Widodo Dan Herlawati ; 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.9 : Aktivitas dengan detail rincian

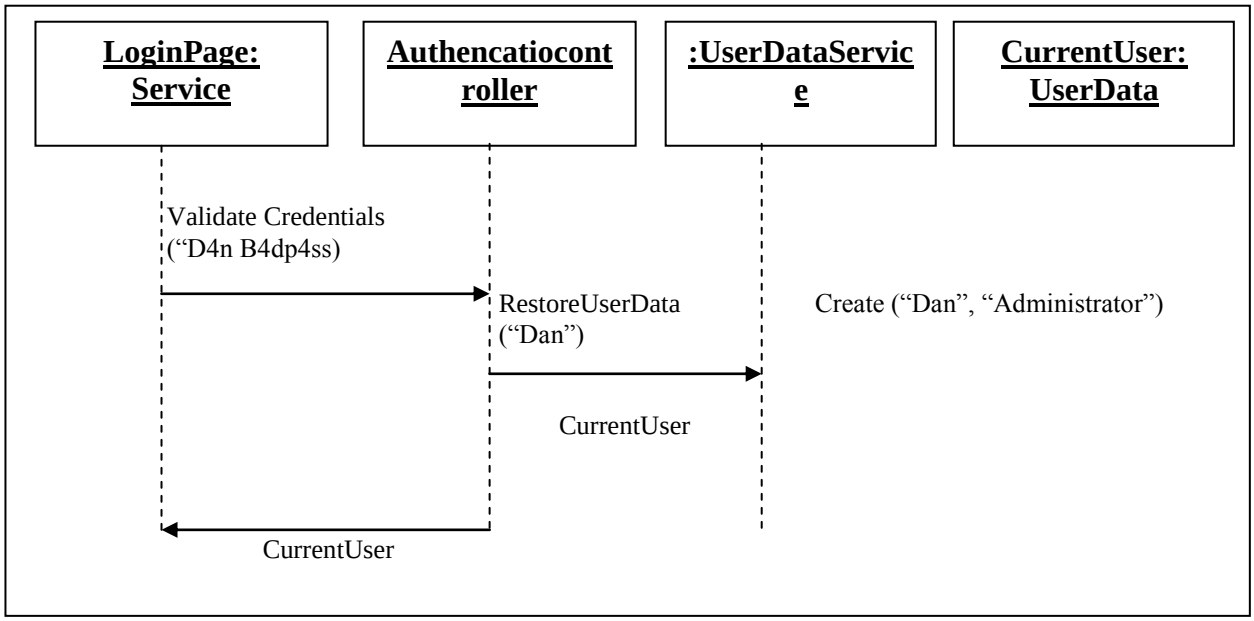
(Sumber : Prabowo Pudjo Widodo Dan Herlawati ; 2011 : 145)

6. *Sequence Diagram*

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).*

Menurut Pilone (2005 : 174) 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 : 174 – 175).



Gambar II.10 : Diagram Urutan

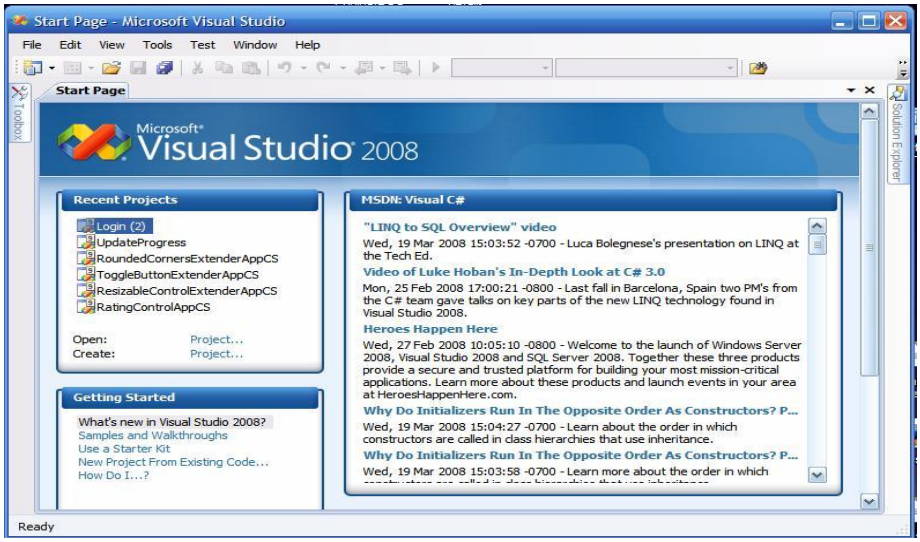
(Sumber : Prabowo Pudjo Widodo Dan Herlawati ; 2011 : 175)

II.10. Bahasa Pemograman *Microsoft Visual Studio 2008*

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 meng*update* aplikasi lama yang dibuat *Microsoft Visual Basic* 6.0 ke teknologi. *Net* karena kelebihan-kelebihan yang ditawarkan, terutama memungkinkan pengembang perngkat lunak secara cepat mampu membuat program *robust*, serta berbasiskan integrasi ke internet yang dikenal dengan *XML Web Service* (Ketut Darmayuda ; 2008 : 1)

Untuk melihat tampilan visual studio 2008 dapat dilihat pada gambar II.11. sebagai berikut :



Gambar II.11 : Tampilan Utama Visual Studio 2008

(Sumber : Ketut Darmayuda ; 2008 : 12)

II.11. *Microsoft SQL Server*

SQL Server Management Studio membantu anda mengatur database dengan mudah. Anda dapat melakukan pengaturan atas beberapa pada sebuah komputer saja atau melakukan pengaturan *server* secara *remote*. Anda dapat juga membuat *database*, *table*, *index*, dan melakukan manipulasi data terhadap *database* dan tabel-tabelnya.

SQL Server Management Studio memiliki beberapa komponen penting yang mewakili kegunaanya dalam perancangan *database*, dan melakukan pengaturan sistem secara keseluruhan. Komponen-komponen tersebut itu adalah :

a. *Registered Server*

b. *Object Explorer*

c. *Query editor*

Adapun tampilan Microsoft SQL Server 2008 dapat dilihat pada gambar.11.12 sebagai berikut (Wahana Komputer; 2008 : 40)

Gambar II.12 : SQL Server 2008

(Sumber : Wahana Komputer ; 2008 : 40)

II.11.1. *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 dapat menampilkannya pada menu *View Regristed Servers* atau menekan kombinasi tombol CTRL + ALT + G. 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 *regristered server* yang disimpan pada mesin lokal. Anda dapat melakukan penambahan atau pengurangan koneksi ke *server*. Anda dapat mengkelompokan koneksi – koneksi ke server tersebut berdasarkan tipe servernya yaitu *Database Engine, Analysis Service, Reporting Service, Intergration Service*.

2. *Object Explorer*

Anda dapat melihat berbagai obyek yang ada pada sebuah server pada panel ini. Bila panel ini tidak terlihat maka anda dapat menampilkannya dengan menu *View Object Explorer*. Apabila anda melakukan ekpansi 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 melakukan *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 memanfaatkan tombol *New Query* pada *toolbar* (Wahana Komputer ; 2008 : 45-49).

II.11.2. Komponen –Komponen *SQL Server* 2008

Ada 5 komponen-komponen SQL Server 2008 adalah sebagai berikut :

1. *Literal Value*

Yang termasuk dengan *literal value* adalah huruf (a – z), *numerik* (0-9), dan hexadesimal (0x). *Literal value* juga dikenal dengan konstanta. Sebuah *konstanta string* terdiri atas satu atau beberapa karakter yang diapit tanda kutip tunggal *(apostroph)* atau tanda petik ganda. Defenisi *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 dan 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 *Delimeted Identifier*. *Delimeted Identifier* adalah *identifier* khusus yang memungkinkan reserver word digunakan menjadi *identifier* dan juga membolehkan adanya spasi pada nama *database.*

3. Komentar

Komentar dalam pemograman ataupun *scripting* diperlukan untuk memberikan keterangan singkat tentang kode-kode yang ada dibawahnya. Sehingga sewaktu ada kerusakan, kesalahan, *programmer* 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 di awali dengan / * dan di akhiri dengan */.
b. Komentar yang menggunakan tanda – untuk memberi komentar pada baris yang dimaskud saja. Biasanya digunakan untuk menerangkan *identifier* atau *reserved word.*

4. *Identifier*

Dalam pemograman bahasa *T-SQL* untuk *query*, *identifier* digunakan untuk melakukan identifikasi *database* dan obyek-obyeknya seperti tabel dan *index*. Diidentifikasi 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 memilikiarti khusus dan harus dituliskan dengan aturan tertentu. Dalam bahasa *T-SQL reserved word* dan juga memiliki banyak fungsi. Reserved word tidak dapat digunakan sebagai nama sebuah obyek kecuali obyek tersebut didefenisikan sebagai *delimeted identifier*. (Wahana Komputer ; 2008: 84-86).

DAFTAR PUSTAKA

Sutabri Tata, 2012, ***"Konsep Sistem Informasi",*** Andi, Yogyakarta.

Ginting Nembah F Hartimbul, 2011, ***"Manajemen Pemasaran",*** Yrama Widya, Bandung.

Linda Iresa, dkk, ***"Aplikasi Pencatatan Kredit Berbasis Desktop",*** http://repository.politekniktelkom.ac.id/Proyek%20Akhir/KA/JURNAL%20PA%20APLIKASI%20PENCATATAN%20KREDIT%20%BERBASIS%20DESKTPOP.pdf (diakses tanggal 3 Agustus 2013)

Simarmata Janner, dkk, 2010, ***"Basis Data",*** Andi Offset, Yogyakarta.

Diana Anastasia dan Setiawati Lilis,2011, ***"Sistem Informasi Akuntansi",*** Andi , Yogyakarta.

Widodo Pudjo Prabowo dan Herlawati, 2011, ***"Menggunakan UML",*** Informatika, Bandung.

Ketut Darmayuda , 2009, ***"Pemrograman Aplikasi Database dengan Microsoft Visual Basic.Net 2008",*** Informatika, Bandung.

Wahana Komputer , 2010, ***"SQL Server 2008 Express",*** Andi, Yogyakarta