

BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Terdapat dua kelompok pendekatan didalam mendefenisikan sistem, yaitu yang menekankan pada prosedur dan elemennya atau komponennya. Pendekatan sistem yang lebih menekankan pada prosedur menurut Jerry FitzGerald, Ardra F. FitzGerald, Warren D. Stallings, Jr mendefinisikan sistem sebagai berikut : Suatu sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu.

Pendekatan sistem yang merupakan jaringan kerja dari prosedur lebih menekankan urutan operasi didalam sistem. Sedangkan prosedur (*procedure*) menurut Richard F. Neuschel mendefenisikan prosedur sebagai berikut : Suatu prosedur adalah suatu urutan operasi klerikal (tulis menulis), biasanya melibatkan beberapa orang di dalam satu atau lebih departemen, yang diterapkan untuk menjamin penanganan yang seragam dari transaksi-transaksi bisnis yang terjadi.

Pendekatan sistem yang menekankan pada komponen akan lebih mudah dalam mempelajari suatu sistem untuk tujuan analisis dan perancangan suatu sistem. Untuk menganalisis dan merencanakan suatu sistem, analisis dan perancangan sistem harus mengerti terlebih dahulu mengenai komponen-komponen atau elemen-elemen atau subsistem-subsistem dari sistem tersebut.

Suatu sistem mempunyai maksud tertentu, ada yang menyebutkan maksud dari suatu sistem adalah untuk mencapai suatu tujuan (*goal*) dan ada yang menyebutkan untuk mencapai sasaran (*objectives*). Goal biasanya dihubungkan dengan ruang lingkup yang lebih luas dan sasaran dalam ruang lingkup yang lebih sempit. Bila merupakan suatu sistem utama, seperti misalnya sistem bisnis, maka istilah *goal* lebih tepat diterapkan. Untuk sistem akuntansi atau sistem-sistem yang lainnya yang merupakan bagian atau subsistem dari sistem bisnis, maka istilah *objectives* yang lebih tepat. Jadi tergantung dari ruang lingkup dari mana memandang sistem tersebut, seringkali tujuan (*goal*) dan sasaran (*objectives*) digunakan bergantian dan tidak dibedakan.

Sistem adalah kumpulan dari elemen-elemen yang berinteraksi untuk mencapai suatu tujuan (Jogiyanto, HM; 2005: 1-2).

II.1.1. Karakteristik Sistem

Sistem mempunyai beberapa karakteristik atau sifat-sifat tertentu, antara lain :

1. **Komponen Sistem (*Component*)**

Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk suatu komponen sistem atau bagian-bagian dari sistem.

2. **Batas Sistem (*Boundary*)**

Merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lain atau dengan lingkungan kerjanya. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem (*Environment*)

Suatu sistem yang ada di luar dari batas sistem yang dipengaruhi oleh operasi sistem.

4. Penghubung Sistem (*Interface*)

Media penghubung antara suatu subsistem dengan subsistem lain. Adanya penghubung ini memungkinkan berbagai sumber daya mengalir dari suatu subsistem ke subsistem lainnya.

5. Masukan Sistem (*Input*)

Energi yang masuk ke dalam sistem, berupa perawatan dan sinyal. Masukan perawatan adalah energi yang dimasukkan supaya sistem tersebut dapat berinteraksi.

6. Keluaran Sistem (*Output*)

Hasil energi yang diolah dan diklasifikasikan menjadi keseluruhan yang berguna dan sisa pembuangan.

7. Pengolahan Sistem (*Process*)

Suatu sistem dapat mempunyai suatu bagian pengolahan yang akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem (*Object*)

Tujuan yang ingin dicapai oleh sistem, akan dikatakan berhasil apabila mengenai sasaran atau tujuan (Jogiyanto, HM; 2005: 4-5).

II.1.2. Klasifikasi Sistem

Suatu sistem dapat diklasifikasikan menjadi seperti berikut :

1. Sistem diklasifikasikan sebagai sistem abstrak (*Abstract System*) dan sistem fisik (*Physical System*)
2. Sistem abstrak adalah suatu sistem yang berupa pemikiran atau ide-ide yang tidak tampak secara fisik, sedangkan sistem fisik adalah sistem yang ada secara fisik.
3. Sistem alamiah dan sistem buatan manusia Sistem alamiah adalah sistem yang terjadi melalui proses alam sedangkan sistem buatan manusia adalah sistem yang dirancang oleh manusia.
4. Sistem tertentu dan sistem tak tentu Sistem tertentu adalah suatu sistem yang operasinya dapat diprediksi secara tepat sedangkan sistem tak tertentu adalah sistem dengan perilaku ke depan yang tidak dapat diprediksi.
5. Sistem tertutup dan sistem terbuka Sistem tertutup adalah sistem yang tidak terpengaruh oleh lingkungan luar atau otomatis, sedangkan sistem terbuka adalah sistem yang berhubungan dan terpengaruh oleh lingkungan luar (Jogiyanto, HM; 2005: 6-7).

II.2. Informasi

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya. (Jogiyanto, HM; 2005: 8).

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).

Dari definisi-definisi tersebut dapat disimpulkan bahwa informasi adalah data yang telah diklasifikasikan atau diinterpretasi untuk digunakan dalam pengambil keputusan.

II.2.1. Sistem Informasi

Sistem informasi adalah suatu sistem di dalam 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 (Jogiyanto, HM; 2005: 11).

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)

Dari definisi-definisi tersebut dapat disimpulkan bahwa sistem informasi adalah suatu sistem di dalam 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.

II.2.2. Komponen-Komponen Sistem Informasi

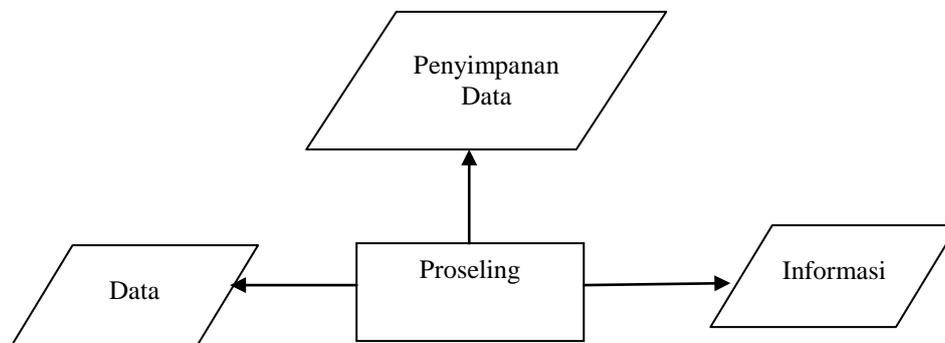
Dalam suatu sistem informasi terdapat komponen-komponen sebagai berikut :

1. Perangkat keras (*hardware*), mencakup berbagai peranti fisik seperti komputer dan printer.
2. Perangkat lunak (*software*) atau program, yaitu sekumpulan instruksi yang memungkinkan perangkat keras memproses data.
3. Prosedur, yaitu sekumpulan aturan yang dipakai untuk mewujudkan pemrosesan data dan pembangkitan keluaran yang dikehendaki.
4. Orang, yaitu semua pihak yang bertanggung jawab dalam pengembangan sistem informasi, pemrosesan dan penggunaan keluaran sistem informasi.
5. Basis data (*database*), yaitu sekumpulan tabel, hubungan dan lain-lain yang berkaitan dengan penyimpanan data.
6. Jaringan komputer dan komunikasi data, yaitu sistem penghubung yang memungkinkan sumber (*resource*) dipakai secara bersama atau diakses oleh sejumlah pemakai (Jogiyanto, HM; 2005: 12-13).

II.3. Data

Mengenai pengertian data, lebih jelas apa yang didefinisikan oleh Drs. Jhon J. Longkutoy dalam bukunya “ Pengenalan Komputer” sebagai berikut : istilah 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.1. Pemrosesan Data

Sumber : (Tata Sutabri; 2012 : 2).

II.3.1. Siklus 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 :

1. Penyimpanan data (*data storage*)

Penyimpanan data meliputi pekerjaan pengumpulan (*filig*), 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 :

1. *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.

2. *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, statistik, 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; 2012 : 6-8).

Dari pendapat diatas diambil suatu kesimpulan 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.

II.4. Proyek

Proyek adalah kegiatan-kegiatan yang dapat direncanakan dan dilaksanakan dalam suatu bentuk kesatuan dengan mempergunakan sumber-sumber untuk benefit. Kegiatan-kegiatan tersebut dapat berbentuk investasi baru seperti pembangunan pabrik, pembuatan jalan raya, atau kereta api, irigasi, bendungan, perkebunan, pembukaan hutan, pendirian gedung-gedung sekolah atau rumah sakit, survai atau penelitian, perluasan atau perbaikan program-program yang sedang berjalan, dan sebagainya. Suatu proyek dapat diselenggarakan oleh

instansi pemerintahan, badan-badan swasta, atau organisasi sosial maupun oleh perorangan.

Sumber-sumber yang dipergunakan dalam pelaksanaan proyek dapat berbentuk barang-barang modal, tanah, bahan-bahan setengah jadi, bahan-bahan mentah, tenaga kerja dan waktu. Sumber-sumber tersebut sebagian atau seluruhnya, dapat dianggap sebagai barang atau jasa konsumsi yang dikorbankan dari penggunaan masa sekarang untuk memperoleh benefit yang lebih besar di masa yang akan datang.

Benefit tersebut dapat berbentuk tingkat konsumsi yang lebih besar, penambahan kesempatan kerja, perbaikan tingkat pendidikan atau kesehatan, dan perubahan/ perbaikan suatu sistem atau struktur. Suatu proyek dinyatakan berakhir bila sudah pasti atau diduga tidak memberikan benefit lagi.

Kegiatan-kegiatan dilaksanakan dalam suatu bentuk satu kesatuan berarti bahwa baik sumber-sumber yang dipergunakan dalam suatu proyek maupun hasil-hasil proyek tersebut dapat dipisahkan dari sumber-sumber yang dipergunakan oleh dan hasil-hasil dari kegiatan yang lain (Cilve Gray dkk ; 2005 : 1).

II.5. Entity Relationship Diagram (ERD)

Merupakan suatu model untuk menjelaskan hubungan antar dua dalam basis data berdasarkan suatu persepsi bahwa *real word* terdiri dari *object-object* dasar yang mempunyai hubungan atau antar *object-object* tersebut. Relasi antar *object* dengan menggunakan symbol-simbol grafis tertentu. Model *entity relationship* adalah suatu penyajian dengan menggunakan *entity* dan *relationship*. Diperkenalkan pada tahun 1976 oleh P.P. Chen.

II.5.1. Komponen-komponen yang terdapat didalam *Entity Relationship Model*

1. *Entity*

- a. Adalah sesuatu yang dapat dibedakan dalam dunia nyata dimana informasi yang berkaitan dengannya dikumpulkan.
- b. *Entity set* adalah kumpulan *entity* yang sejenis.
- c. Symbol yang digunakan untuk *entity* adalah persegi panjang.
- d. *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 konsep atau logic, yaitu *entity* yang tidak dapat dilihat. Contohnya : pekerjaan, perusahaan, rencana, mata kuliah, dan lain-lain.
- e. Simbol yang digunakan untuk *entity* adalah persegi panjang.



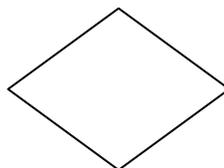
Gambar II.2. *Entity*

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



Gambar II.3. Relationship

Sumber : (Linda Marlinda; 2004:18)

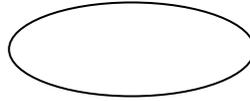
Contoh :



3. Attribute

- a. Adalah karakteristik dari *entity* atau *relationship* yang menyediakan penjelasan detail tentang atau *relationship* tersebut.
- b. Attribute value (nilai atribut) adalah suatu data aktual atau informasi yang disimpan di suatu atribut di dalam suatu *entity* atau *relationship*.
- c. Terdapat dua jenis atribut, yaitu :
 1. *Indetifer (key)*, untuk menentukan suatu *entity* secara unik.
 2. *Descriptor (nonkey attribute)*, untuk menentukan karakteristik dari suatu *entity* yang tidak unik.

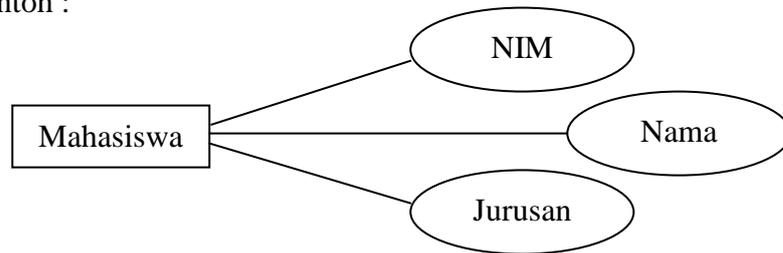
- d. Simbol yang digunakan adalah bentuk oval



Gambar II.4. Attribute

Sumber : (Linda Marlinda; 2004:18)

Contoh :

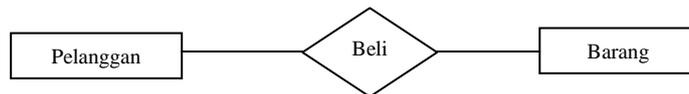


4. Indicator Tipe

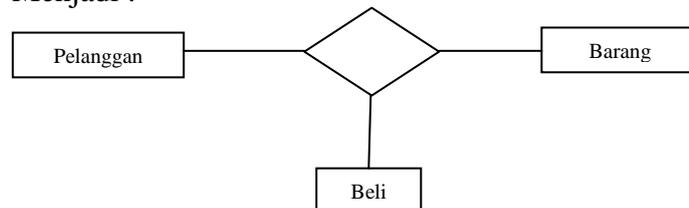
- a. *Indicator tipe associative object*

Berfungsi sebagai suatu objek dan suatu *relationship*

Contoh :



Menjadi :



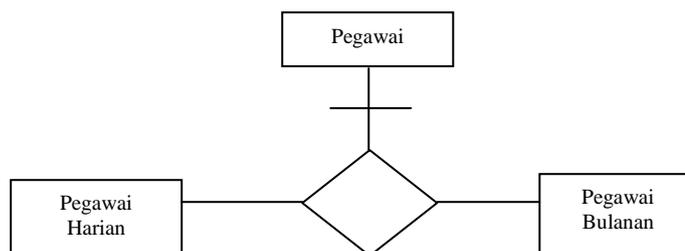
Gambar II.5. Indicator Tipe

Sumber : (Linda Marlinda; 2004:19)

b. *Indicator tipe supertipe*

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

Contoh :



Gambar II.6. Indicator Tipe SuperTipe

Sumber : (Linda Marlinda; 2004:19)

II.6. Kamus Data

Kamus data (KD) atau data dictionary (DD) yang disebut juga dengan system dictionary data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan demikian KD, analisis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. KD dibuat pada pada tahap analisis sistem dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem. Pada tahap analisis, KD dapat digunakan sebagai alat komunikasi antara analisis sistem dengan pemakai sistem tentang data yang mengalir di sistem. Pada tahap perancangan sistem KD digunakan untuk merancang input, merancang laporan-laporan dan database. KD dibuat berdasarkan arus data yang ada di DAD. Arus data di DAD sifatnya global, hanya

ditunjukkan nama arus datanya saja (Prof. Dr. Jogiyanto HM, MBA, Akt; 2005 : 725).

II.6.1. Isi Kamus Data

KD harus dapat mencerminkan keterangan yang jelas tentang data yang dicatatnya. Untuk maksud keperluan ini maka KD harus memuat hal-hal berikut ini :

1. Nama Arus Data

Karena KD dibuat berdasarkan arus data yang mengalir di DAD, maka nama dari arus data juga harus dicatat di KD, sehingga mereka yang membaca DAD dan memerlukan penjelasan lebih lanjut tentang suatu arus data tertentu di DAD dapat langsung mencarinya dengan mudah di KD.

2. Alias

Alias atau nama lain dari data dapat dituliskan bila nama lain ini ada. Alias perlu ditulis karena data yang mempunyai nama yang berbeda untuk orang atau departemen satu dengan yang lainnya. Misalnya bagian pembuat faktur dan langganan menyebut bukti penjualan sebagai faktur, sedang bagian gudang menyebutnya sebagai tembusan permintaan persediaan barang. Baik faktur dan tembusan permintaan persediaan ini mempunyai struktur data yang sama tetapi mempunyai struktur yang berbeda.

3. Bentuk data

Telah diketahui bahwa arus data mengalir.

- a. Dari kesatuan luar ke suatu proses, data yang mengalir ini biasanya tercatat di suatu dokumen atau formulir.
- b. Hasil dari suatu proses ke kesatuan luar, data yang mengalir biasanya terdapat di media laporan atau *query* tampilan layar atau dokumen hasil cetakan komputer.
- c. Hasil dari suatu proses yang lain, data yang mengalir biasanya dalam bentuk variabel atau parameter yang dibutuhkan oleh proses penerimanya.
- d. Hasil dari suatu proses yang direkamkan ke simpanan data, data yang mengalir ini biasanya berbentuk suatu variabel.
- e. Dari simpanan data dibaca oleh suatu proses, data yang mengalir ini biasanya berupa suatu *field* (item data).

Dengan demikian bentuk dari data yang mengalir dapat berupa :

1. Dokumen dasar atau formulir
2. Dokumen hasil cetakan komputer
3. Laporan tercetak.
4. Tampilan dilayar monitor.
5. Variabel.
6. Parameter
7. *Field*

Bentuk dari data ini perlu dicatat di KD, karena dapat digunakan untuk mengelompokkan KD ke dalam kegunaanya, sewaktu perancangan sistem KD yang mencatat data yang mengalir dalam

bentuk dokumen dasar atau formulir yang digunakan untuk merancang bentuk input sistem. KD yang mencatat data yang mengalir dalam bentuk laporan tercetak dan dokumen hasil cetakan komputer akan digunakan untuk merancang *output* yang akan dihasilkan oleh sistem. KD yang mencatat data yang mengalir dalam bentuk tampilan layar monitor akan digunakan juga untuk merancang tampilan layar yang akan dihasilkan oleh sistem. KD yang mencatat data yang mengalir ke dalam bentuk parameter dan variabel akan digunakan untuk merancang proses dari program. KD yang mencatat data yang mengalir ke dalam bentuk dokumen, formulir, laporan, dokumen cetakan komputer, tampilan di layar monitor, variabel, dan *field* akan digunakan untuk merancang database.

4. Arus data

Arus data menunjukkan dari mana data yang mengalir dan kemana data akan menuju. Keterangan arus data ini perlu dicatat di KD supaya memudahkan mencari arus data di DAD (Prof. Dr. Jogiyanto, HM, Akt; 2005 : 726-727).

II.7. Normalisasi

Normalisasi adalah proses pengkelompokkan attribute-attribute dan suatu relasi sehingga membentuk *Well- Structure Relation*. Normalisasi merupakan proses pengkelompokkan elemen data menjadi suatu tabel-tabel menunjukkan entity dan relasinya. Normalisasi ditemukan pada tahun 1970 oleh E. F. CODD.

II.7.1. Well-Structure Relation

Well- Structure Relation adalah sebuah *relation* dengan jumlah kerangkapan datanya sedikit (*Minimum amount of redundancy*), serta memberikan kemungkinan bagi user untuk melakukan *Insert*, *Delete*, dan *Modify* terhadap baris-baris data pada *relation* tersebut, yang tidak berakibat terjadinya *Error* atau INKONSETENSIDATA, yang disebabkan oleh operasi-operasi tersebut.

Contoh :

Terdapat sebuah *Relation Course* dengan ketentuan sebagai berikut :

- a. Setiap mahasiswa hanya boleh mengambil satu mata kuliah saja.
- b. Setiap mata kuliah mempunyai uang kuliah yang standar (tidak tergantung pada mahasiswa yang mengambil mata kuliah tersebut).

Tabel II.1. *Relation Course*

STUDENT-ID	KODE-MTK	BIAYA
92130	CS-200	75
92200	CS-300	100
99250	CS-300	75
92425	CS-400	150
92500	CS-300	100
92575	CS-500	50

Sumber : (Linda Marlinda; 2004 :115)

Relation Course diatas merupakan sebuah *relation* yang sederhana dan terdiri dari 3 kolom/attribute (Linda Marlinda; 2004 : 115).

a. Bentuk tidak normal (*Unnormalized Form*)

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti suatu format tertentu. Dapat saja data tidak lengkap atau terduplikasi. Data dikumpulkan apa adanya dengan saat menginput. Contoh data :

Tabel II.2. Bentuk tidak normal (*Unnormalized Form*)

No_Siswa	Nama	Pa	Kelas 1	Kelas 2	Kelas 3
22890100	Shandy	Linda	1234	1543	1543
22890101	Susi	Riska	1234	1775	

Sumber :(Linda Marlinda; 2004 : 122)

Siswa yang punya nomor siswa, nama, dan pa mengikuti 3 mata pelajaran/kelas. Di sini ada perulangan kelas 3 kali ini bukan bentuk tidak 1 NF.

b. Bentuk Normal Kesatu (1 NF/ *Fisrt Normal Form*)

Suatu relasi 1 NF dan hanya jika sifat dan setiap relasi atributnya bersifat *atomic*. Atom adalah zat terkecil yang masih memiliki sifat induknya. Bila dipecah lagi maka ia tidak memiliki sifat induknya.

Ciri-ciri 1 NF :

1. Setiap data dibentuk kedalam *flat file* data terbentuk per satu *record* nilai dan field berupa "*atomic value*".
2. Tidak ada *set attribute* yang berulang atau bernilai ganda,
3. Tiap *field*
4. Hanya satu pengertian.

Tabel II.3. Bentuk Normal Ke Satu (1 NF/ First Normal Form)

No_Siswa	Nama	Pa	Kelas 1
22890100	Shandy	Linda	1234
22890100	Shandy	Linda	1543
22890101	Susi	Riska	1234
22890101	Susi	Riska	1775
22890101	Susi	Riska	1543

Sumber : (Linda Marlinda; 2004 : 122)

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

Bentuk normal kedua mempunyai syarat yaitu bentuk data telah memenuhi kriteria bentuk normal kesatu. *Attribute* bukan kunci haruslah bergantung secara fungsi pada *primary key*. Jadi, untuk membentuk normal kedua haruslah sudah ditentukan kunci-kunci *field*. Kunci *field* haruslah unik dan dapat mewakili *attribute* lain yang menjadi anggotanya. Misal : dari contoh relasi siswa pada 1 NF terlihat bahwa *primary key* adalah nomor siswa. Nama siswa dan pa bergantung fungsi pada no_siswa, tetapi kode_kelas bukanlah fungsi dan siswa dipecah menjadi 2 relasi :

- Relasi siswa

Tabel II.4. Bentuk Normal Kedua Relasi Siswa (2 NF/ *Second Normal Form*)

No_Siswa	Nama	Pa
22890100	Shandy	Linda
22890101	Susi	Riska

Sumber : (Linda Marlinda; 2004 : 123)

- Relasi ambi_Kelas

Tabel II.5. Bentuk Normal Kedua Relasi ambil_Kelas (2 NF/ *Second Normal Form*)

No_Siswa	Kode Kelas
22890100	1234
22890100	1543
22890101	1234
22890101	1775
22890101	1543

Sumber : (Linda Marlinda; 2004 : 123)

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

Untuk menjadi bentuk normal ketiga maka relasi dalam bentuk normal kedua dan semua *attribute* bukan primer tidak punya hubungan yang transistif. Dengan kata lain, setiap *attribute* bukan

kunci haruslah bergantung hanya pada *primary key* dan *primary key* secara menyeluruh.

Contoh pada bentuk normal kedua diatas termasuk juga bentuk normal ketiga karena seluruh attribute yang ada bergantung penuh pada kunci primernya.

e. ***Boyce-Cood Normal Form (BCNF)***

BCNF mempunyai paksaan lebih kuat dan bentuk normal ketiga untuk menjadi BCNF, relasi harus dalam bentuk normal kesatu dan setiap *attribute* harus bergantung fungsi pada *attribute superkey*.

Pada contoh di bawah ini terdapat relasi seminar dengan ketentuan sebagai berikut :

1. Kunci primer adalah no_siswa +seminar
2. Siswa boleh mengambil satu atau dua seminar.
3. Setiap siswa dibimbing oleh salah satu di antara 2 instruktur seminar tersebut.
4. Setiap instruktur boleh hanya mengambil satu seminar saja.

Pada contoh ini no_siswa dan seminar menunjuk seorang instruktur :

Relasi seminar

Tabel II.6. Boyee-Cood Normal Form (BCNF)

No_Siswa	Seminar	Instruktur
22890100	2281	Si doel
22890101	2281	Pak tile
22890102	2291	Mandra
22890101	2291	Basuki
22890109	2291	Basuki

Sumber : (Linda Marlinda; 2004 : 124)

Bentuk relasi seminar adalah bentuk normal ketiga, tetapi tidak BCNF karena nomor seminar masih tergantung fungsi pada instruktur. Jika setiap instruktur dapat mengajar hanya pada satu seminar saja, maka seminar bergantung fungsi pada satu *attribute* bukan *superkey* seperti disyaratkan oleh BCNF. Maka relasi seminar haruslah dipecah menjadi dua yaitu :

Relasi Pengajar

Tabel II.7. Boyee-Cood Normal Form (BCNF)

Instruktur	Seminar	No_Siswa	Instruktur
Si doel	2281	22890100	Si doel
Pak tile	2281	22890101	Pak tile
Mandra	2291	22890102	Mandra
Basuki	2291	22890101	Basuki
		22890109	Basuki

Sumber : (Linda Marlinda; 2004 : 124)

f. Bentuk Normal Keempat (4NF)

Relasi R adalah bentuk 4 NF jika dan hanya jika relasi tersebut juga termasuk BCNF dan semua ketergantungan *multivalued* adalah juga ketergantungan fungsional.

g. Bentuk Normal Kelima (5 NF)

Disebut juga PINF (*Projection Join Normal Form*) dan 4 NF dilakukan dengan menghilangkan ketergantungan *join* yang merupakan kunci kandidat (Linda Marlinda; 2004 : 122-125).

II.7.2. Unified Modeling Language (UML)

UML singkatan dari *Unified Modelling Language* 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 lainnya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorientasikan objek menggunakan bahasa model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. *UML* memungkinkan para anggota team untuk bekerja sama dalam mengaplikasikan beragam sistem. Intinya, *UML* merupakan alat komunikasi yang konsisten dalam mensupport para pengembang sistem saat ini. Sebagai perancang sistem mau tidak mau pasti menjumpai *UML*, baik kita sendiri yang membuat sekedar membaca diagram *UML* buatan orang lain (Prabowo Pudji Widodo, Herlawati; 2011 : 6-7).

II.7.3. 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 berorientasi objek. Meskipun bersifat statis, sering pula diagram kelas memuat kelas-kelas.
2. Diagram paket (*Package Diagram*) bersifat statis. Diagram ini memperlihatkan kumpulan kelas-kelas merupakan bagian dari diagram komponen.
3. Diagram *Use Case* bersifat statis. Diagram ini memperlihatkan himpunan *use-case* dan aktor-aktor (suatu jenis khusus dari kelas). Diagram ini terutama sangat penting untuk mengorganisasi dan memodelkan perilaku suatu sistem yang dibutuhkan serta diharapkan pengguna.
4. Diagram interaksi dan *Sequence* (urutan). Bersifat dinamis. Diagram urutan adalah diagram interaksi yang menekankan pada pengiriman pesan dalam waktu tertentu.
5. Diagram komunikasi (*Communication Diagram*) bersifat dinamis. Diagram sebagai pengganti diagram kolaborasi *UML* yang menekankan organisasi *structural* dari objek-objek yang menerima serta mengirim pesan.
6. Diagram *Statechart* (*Statechart Diagram*) bersifat dinamis. Diagram status memperlihatkan keadaan-keadaan pada sistem, memuat status (*State*), transisi kejadian serta aktifitas. Diagram ini terutama penting untuk memperlihatkan

sifat dinamis dari antarmuka (*interface*), kelas, kolaborasi dan 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 beserta komponen-komponen yang ada di dalamnya. Diagram *Deployment* berhubungan erat dengan diagram komponen dimana diagram ini memuat satu atau lebih komponen-komponen. Diagram ini sangat berguna saat aplikasi kita berlaku sebagai aplikasi yang dijalankan pada banyak mesin (*distributed computing*).

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

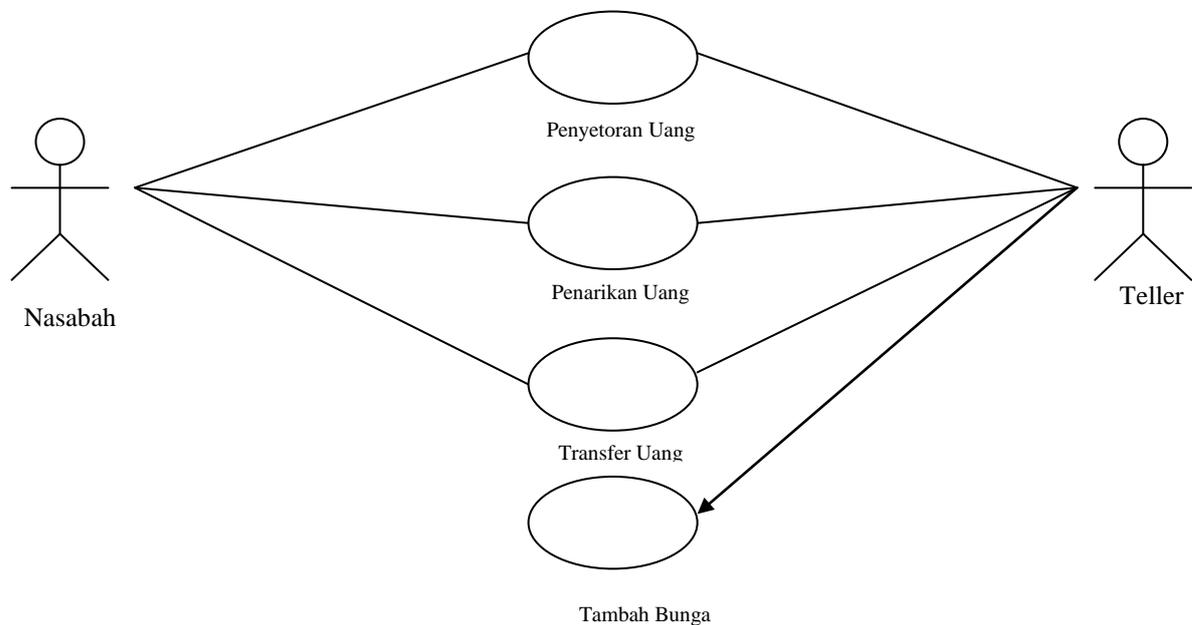
1. *Diagram Use Case (use case diagram)*

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 disediakan 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 Pudji Widodo, Herlawati; 16-17).

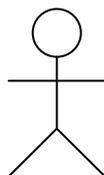


Gambar II.7. Diagram Use Case

Sumber : Prabowo Pudjo Widodo, 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*.

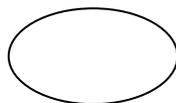


Gambar II.8. Aktor

Sumber : Prabowo Pudjo Widodo, 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.9. Simbol Use Case

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

Use case sangat menentukan karakteristik sistem yang kita buat, oleh karena itu Chonoles (2003:22-23) 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 frase kata kerja yang implikasinya hingga selesai. Misalnya gunakan frase *reserve a room* (pemesanan kamar) dan jangan *reserving a room* (memesan kamar) karena memesan menggambarkan perilaku yang belum selesai.

d. Menyediakan *use case* lawan (*inverse*)

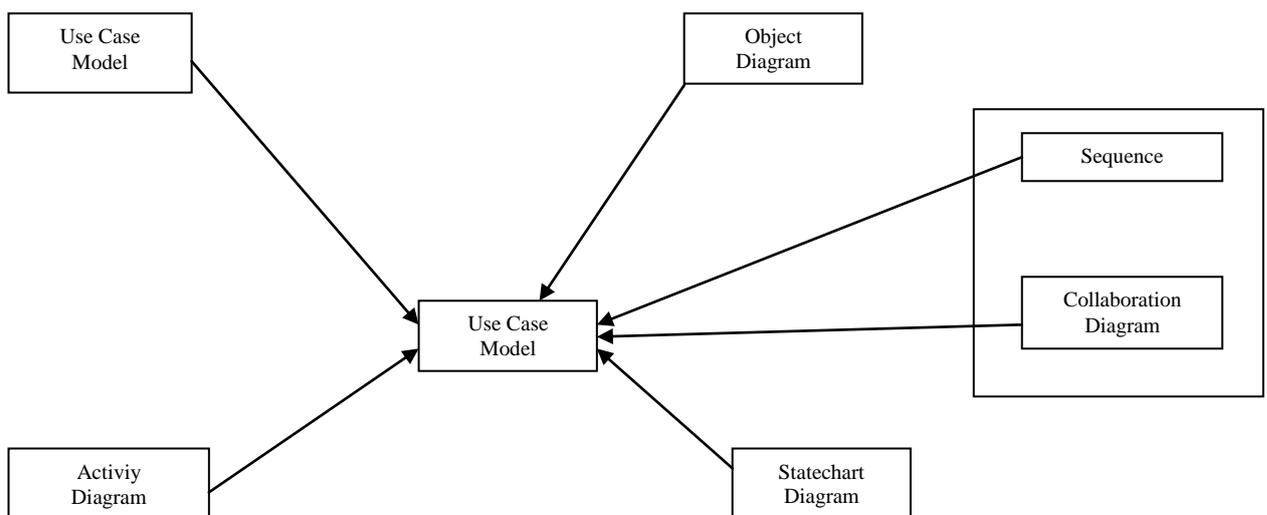
Kita biasanya membutuhkan *use case* yang membatalkan tujuan, misalnya pada *use case* pemesanan kamar, dibutuhkan pula *use case* pembatalan pesanan kamar.

e. Batasi use case hingga satu perilaku saja.

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

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 (Probowo Pudji Widodo; 2011 : 37).



Gambar II.10. Hubungan Diagram Kelas Dengan Diagram UML lainnya

Sumber : Prabowo Pudjo Widodo, Herlawati (2011 : 38)

5. Diagram Aktivitas (*Activity Diagram*)

Diagram aktivitas lebih memfokuskan diri pada eksekusi dan alur sistem dari pada bagaimana sistem dirakit. Diagram ini tidak hanya memodelkan software melainkan memodelkan bisnis juga. Diagram aktivitas menunjukkan aktivitas sistem dalam kumpulan aksi-aksi. Ketika digunakan dalam pemodelan *software*, diagram aktivitas merepresentasikan pemanggilan suatu fungsi tertentu misalnya *call*. Sedangkan bila digunakan dalam pemodelan bisnis, diagram ini menggambarkan aktivitas yang dipicu oleh kejadian-kejadian diluar seperti pemesanan atau kejadian-kejadian internal misalnya penggajian tiap jumat sore (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 *classifier* dikatakan kontek dari aktivitas. Aktivitas dapat mengakses atribut dan operasi *classifier*, tiap objek yang terhubung dan parameter-parameter jika aktivitas memiliki hubungan dengan perilaku. Ketika digunakan dengan model proses bisnis, informasi itu biasanya disebut *process-relevant data*. Aktivitas diharapkan dapat digunakan ulang dalam suatu aplikasi, sedangkan aksi biasanya *specific* dan digunakan hanya untuk aktivitas tertentu.

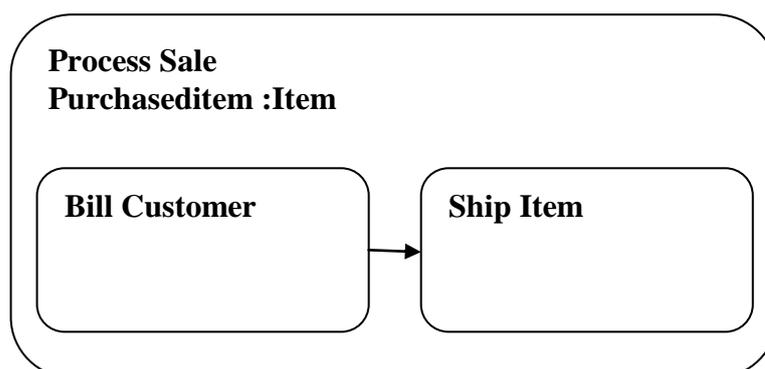
Aktivitas digambarkan dengan persegi panjang tumpul. Namanya ditulis di kiri atas. Parameter yang terlibat dalam aktivitas ditulis dibawahnya.



Gambar II.11. Aktivitas sederhana tanpa rincian

Sumber : Prabowo Pudjo Widodo, Herlawati (2011:145)

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



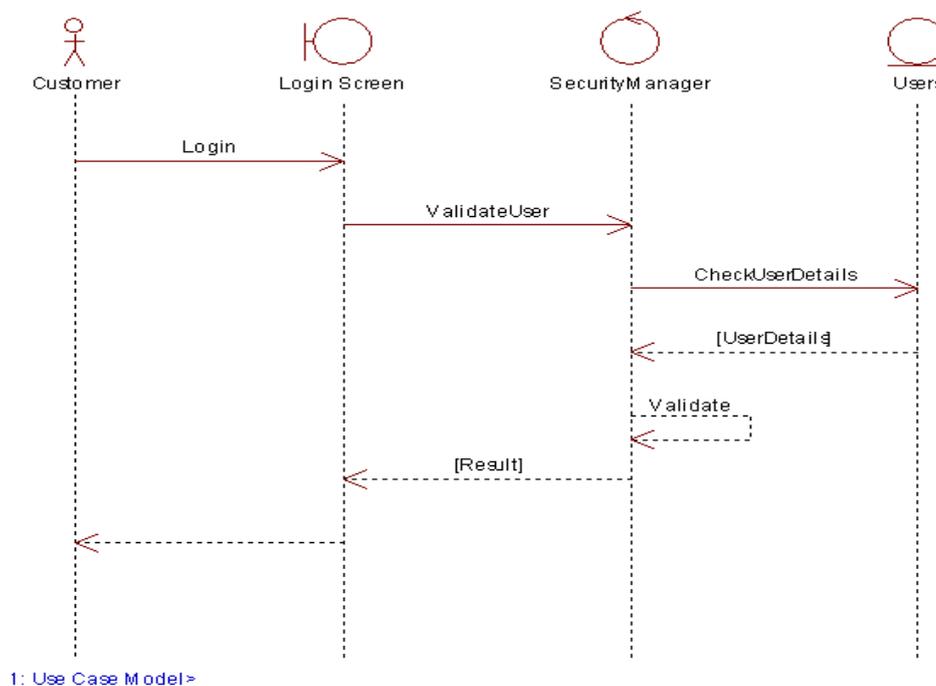
Gambar II.12. Aktivitas dengan detail rincian

Sumber : Prabowo Pudjo Widodo, 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.13. memperlihatkan contoh diagram urutan dengan notasi-notasinya yang akan dijelaskan nantinya.



Gambar II.13. Diagram Urutan

Sumber : Prabowo Pudjo Widodo, Herlawati (2011:175)

II.8. Bahasa Pemrograman *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 mengupdate 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 ; 2009 : 1)

Untuk melihat tampilan visual studio 2008 dapat dilihat pada gambar II.10. sebagai berikut :



Gambar II.14. Tampilan Utama Visual Studio 2008

Sumber : Ketut Darmayuda (2009 : 12)

II.9. MYSQL

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

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

- a. *Mysql* merupakan DBMS (*Database Management System*)
- b. *Database* adalah kumpulan data yang terstruktur. Data dapat berupa daftar belanja, kumpulan gambar, atau yang lebih luas yaitu informasi jaringan perusahaan. Agar dapat menambah, mengakses, dan memproses data tersimpan pada sebuah komputer *database*, kita membutuhkan sistem manajemen *database* (DBMS) seperti *MYSQL Server*. Sejak komputer sangat baik menangani sejumlah besar data, *sistem manajemen database* (DBMS) memainkan peran utama dalam perhitungan baik sebagai peralatan yang berdiri sendiri maupun bagian aplikasi.
- c. *Mysql* merupakan RDBMS (*Relational Database Management System*).

- d. *Database relational* menyimpan data pada table-tabel yang terpisah, bukan menyimpan data dalam ruang penyimpanan yang besar. Hal ini menambah kecepatan *fleksibilitas*.
- e. *Mysql* merupakan *software open source*.
- f. *Open source* berarti setiap orang dapat menggunakan dan mengubah software yang bersangkutan. Setiap orang dapat mendownload *software MYSQL* dari internet dan menggunakan tanpa membayar. Bahkan jika mengkehendaknya anda bisa mempelajari kode sumber dan mengubah sesuai yang anda butuhkan (Wahana Komputer; 2010 : 26-27).

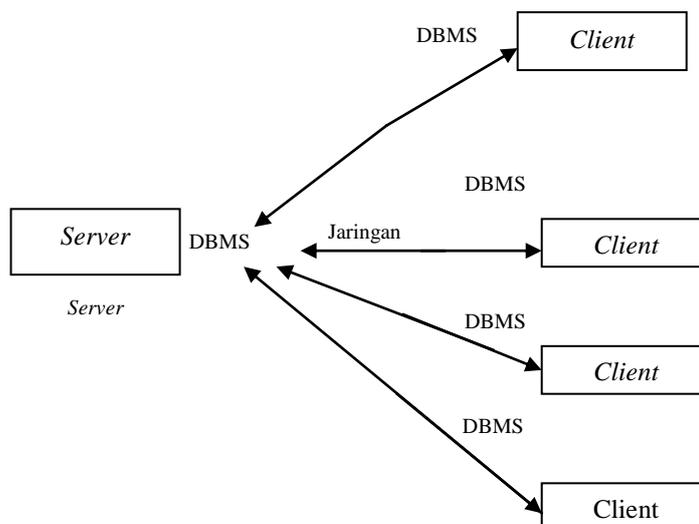
II.9.1. Client Server

Client server adalah satu model komunikasi 2 komputer atau lebih yang berfungsi melakukan pembagian tugas. *Client* bertugas untuk melakukan input, update, dan menampilkan data sebuah *database*. Sementara *server* bertugas untuk menyediakan pelayanan untuk melakukan manajemen yaitu : menyimpan dan mengolah *database* (Wahana Komputer; 2010 : 5).

II.9.2. Arsitekur Client Server

1. Arsitektur StandAlone (1-Tier)

Model pertama aplikasi pemrograman *database client server* adalah standalone atau 1 *tier* (1 -tingkat) adalah sebuah komputer yang mengakses sebuah *database* dari komponen sendiri. Dengan kata lain, aplikasi antarmuka *user* dan aplikasi DBMS terdapat pada komputer yang sama.



Gambar II.15. Arsitektur StandAlone (1-Tier)

Sumber : Wahana Komputer (2010 : 6)

Adapun karakteristik arsitektur 1 tier sebagai berikut :

- a. Beban jaringan menjadi tinggi karena yang diminta adalah file database secara keseluruhan pada komputer *server client* melalui jaringan.
- b. Setiap komputer pada jaringan harus mempunyai DBMS tersendiri untuk menyimpan hasil salinan dari *server* sehingga mengurangi sumber daya yang dimiliki oleh komputer *client* terutama *memory*.
- c. Komputer *client* harus mempunyai kemampuan proses yang tinggi untuk mendapatkan waktu respon yang baik saat komputer *server* mengirimkan *file* yang diminta.
- d. Arsitektur *1-tier* cocok untuk bisnis kecil yang hanya membutuhkan data sebuah komputer untuk memproses dan

menyimpan data sekaligus, tetapi kurang tepat diterapkan pada model jaringan (Wahana Komputer; 2010 : 7).

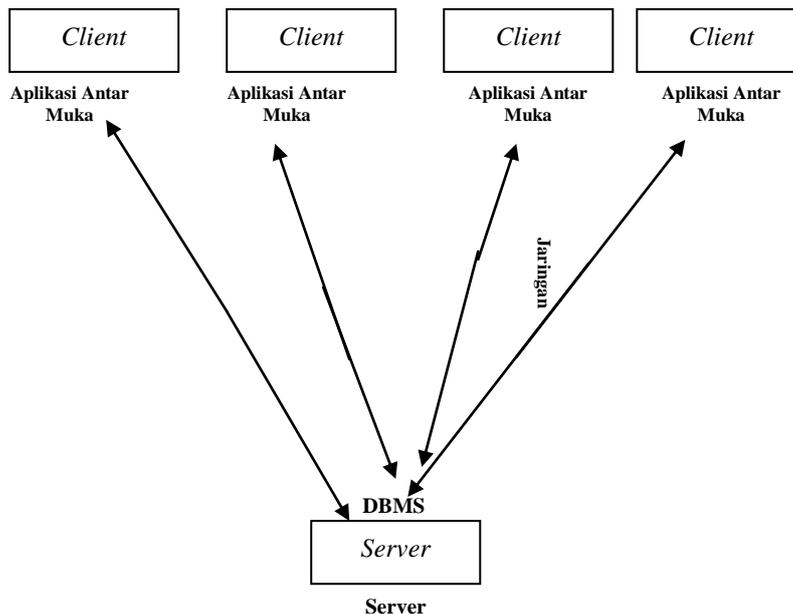
2. Arsitekur 2-Tier

Model kedua sebuah pemograman *database* adalah model 2-tier. Arsitektur pada model demikian membagi tugas antara komputer *client* server. Komputer *client* bertugas menyediakan antar muka untuk *user*, permintaan (*request data*) ke DBMS Server, serta pemrosesan data (mencakup logika penyajian data, logika pemrosesan data, dan logika atau bisnis) komputer *client* hanya mengirimkan sebuah *statement* untuk menambah (*insert*) data, mengubah (*update*), menghapus (*delete*), dan yang terakhir meminta (*select*) data untuk ditampilkan melalui antarmuka yang dibuat oleh *programmer*. Sedangkan *server* bertanggung jawab terhadap penyimpanan, pengelolaan, melayani permintaan akses data, dan pemrosesan *client*.

Karakteristik arsitektur 2-tier adalah :

- a. 2-tier terjadi pada jaringan dan melakukan pemodelan programan *database* dalam 2 tingkat. Tingkat pertama adalah *client* dan tingkat kedua adalah *server*.
- b. Tingkat pertama komputer *client* sebagai penyedia aplikasi antarmuka untuk mengolah *database*, baik menampilkan data kedalam *user interface*, menambah, menghapus data, maupun logika bisnis (*bussines logic*)

- c. Tingkat kedua adalah *server* yang menyediakan aplikasi untuk mengelola *database* serta menyediakan pula *query stored procedure*, dan *triggers*, yang dapat dipanggil *client* untuk mengolah data.
- d. Komputer *client* hanya mengirimkan sebuah *statement sql* untuk meminta data ke *server*.
- e. *Server* hanya memberikan data yang diminta melalui *statement* bersangkutan.
- f. Komputer *server* dituntut untuk memiliki kemampuan pemrosesan yang tinggi karena harus melayani permintaan banyak komputer *client* yang mengakses satu atau lebih DBMS.
- g. Beban jaringan menjadi ringan karena data yang berjalan pada jaringan hanya data yang diminta oleh *client*.
- h. Otentifikasi pemakai, pemeriksaan integritas, dan pemeliharaan kamus data dilakukan pada sisi *server*.
- i. Sederhana dan mudah untuk diterapkan, khususnya pada bisnis kecil yang hanya terdapat pada satu gedung (Wahana Komputer; 2010 : 7-8).



Gambar II.16. Arsitekur 2-Tier

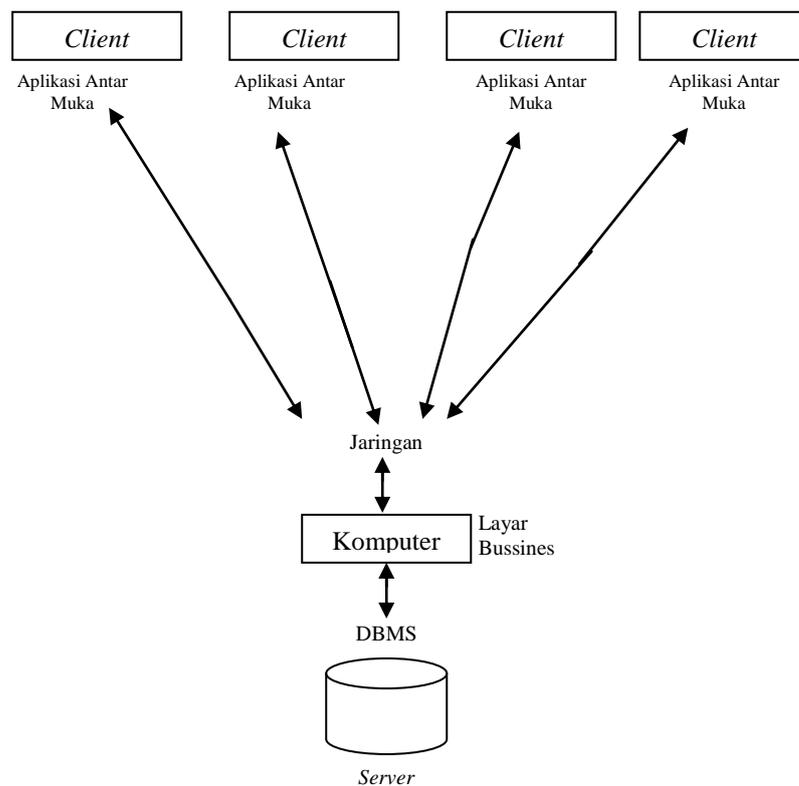
Sumber : Wahana Komputer (2010 : 8)

3. Arsitekur N-Tier

Arsitektur *n-tier* berarti membagi komponen menjadi *n* entitas yaitu 1 tier *client* dan *n-1 tier server*. Seperti pada model sebelumnya *client* bertugas menyediakan antarmuka aplikasi, sedangkan bertugas menyediakan data. Pada model *n-tier* (sebagai contoh adalah 3-tier), server dibagi 2 menjadi, yaitu satu *server* yang dipakai sebagai *bussines object (middle tier)* dan satu *server* yang hanya menyimpan *database (server tier)*.

Secara nyata model 3-tier adalah pada jaringan internet yang hanya memanfaatkan *database*. Internet lapisan pertama adalah komputer *client* yang menampilkan halaman *web*, tempat kontent atau data halaman *web* berasal dari *database*. Lapisan kedua adalah *web* atau

HTTP server yang menterjemahkan *script server side* (*PHP, JSP, ASP*, dan lainnya) dari komputer *client* untuk meminta data pada *database*. Kemudian lapisan ketiga adalah komputer *database server* yang menyediakan *database* yang diminta oleh *web* atau *HTTP server* (Wahana Komputer; 2010: 6-9).



Gambar II.17. Arsitekur N-Tier

Sumber : Wahana Komputer (2010 : 9)