BAB II

TINJAUAN PUSTAKA

II.1. Konsep Dasar Sistem

Konsep dasar sistem akan menguraikan beberapa pengertian sistem, karakteristik sistem, pengertian dan Komponen Sistem Informasi.

II.1.1. Pengertian Sistem

Kata sistem mempuyai beberapa pengertian, tergantung dari sudut mana kata tersebut didefinisikan. Secara garis besar ada dua kelompok pendekatan, yaitu :

1. Pendekatan sistem yang lebih menekankan pada elemen-elemen atau kelompoknya yang dalam hal ini sistem itu didefinisikan sebagai suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu aturan tertentu.
2. Pendekatan sistem sebagai jaringan kerja dari prosedur yang lebih menekankan urutan operasi didalam sistem. Prosedur (*procedure*) didefinisikan oleh Richard F. Neushl sebagai urutan operasi kerja (tulis-menulis), yang lebih departemen, yang diterapkan untuk menjamin penanganan yang seragam dari transaksi bisnis yang terjadi.(Kusrini, 2007 ; 5)

II.1.2. Karakteristik sistem

Menurut Kusrini dan Andri Kuniyo (2007 : 6-7), sistem mempunyai beberapa karakteristik atau sifat-sifat tertentu, antara lain :

1. Komponen Sistem (*System Component*)

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

2. Batasan Sistem (*Boundary*)

   Merupakan daerah yang membatasi suatu sistem dengan sistem yang lain atau dengan lingkungan kerjanya.

3. Subsistem

   Bagian-bagian dari sistem yang beraktivitas dan berinteraksi satu sama lain untuk mencapai tujuan dengan sasarannya masing-masing.

4. Lingkungan Luar sistem (*Environment*)

   Suatu sistem yang ada diluar dari batas sistem yan dipengaruhi oleh operasi sistem.

5. Penghubung sistem (*Interface*)

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

6. Masukan Sistem (*Input*)

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

7. Keluaran Sistem (*Output*)

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

8. Pengolahan Sistem (*Process*)

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

9. Sasaran Sistem (*Object*)

   Tujuan yang ingin dicapai oleh sistem akan dikatakan berhasil apabila mengai sasaran atau tujuan.

II.1.3. Pengertian Sistem Informasi

Untuk memahami pengertian sistem informasi, harus dilihat keterkaitan antara data dan informasi sebagai entitas penting pembentuk sistem informasi. Data merupakan nilai, keadaan, atau sifat yang berdiri sendiri lepas dari konteks apapun. Sementara informasi adalah data yang telah diolah menjadi sebuah bentuk yang berarti bagi penerimanya dan bermanfaat dalam pengambilan keputusan saat ini atau mendatang. Dari pengertian di atas kita dapat mendefenisikan sistem informasi sebagai integrasi antara orang, data, alat dan prosedur yang bekerja sama dalam mencapai suatu tujuan. Jadi sistem informasi adalah data yang telah diproses, atau data yang memiliki arti yang didalamnya terdapat elemen orang, alat, data dan prosedur atau cara. (Eko Nugroho, 2008 : 17)

Tujuannya adalah untuk menyajikan informasi guna pengambilan keputusan pada perencanaan, pemrakarsaan, pengorganisasian, pengendalian kegiatan operasi subsistem suatu perusahaan.

II.1.4. Komponen Sistem Informasi

Dalam suatu sistem informasi terdapat komponen-komponen sebagai berikut :

1. Perangkat keras (*hardware*), mencakup berbagai piranti 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 di kehendaki.
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 lain0lain 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.

   (Kusrini, 2007; 9)

II.2. 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 contoh sebelumnya, adalah transaksi penjualan. Kita memperoleh transaksi penjualan dengan mencatat penjualan tersebut kedalam jurnal penjualan, mengklasifikasikan transaksi dengan menggunakan kode rekening dan memposting transaksi ke dalam jurnal. Kemudian, secara periodik sistem informasi akuntansi akan menghasilkan output berupa laporan keuangan yang terdiri dari neraca dan laporan laba rugi. (Anastasia Diana, Lilis Setiawati; 2011 : 4)

Menurut Kusrini (2007) Sistem informasi akuntansi merupakan sebuah informasi yang mengubah data transaksi bisnis menjadi informasi keuangan yang berguna bagi pemakainya,

Tujuan dari sistem informasi akuntansi adalah :

1. Mendukung operasi sehari-hari
2. Mendukung pengambilan keputusan manajemen
3. Memenuhi kewajiban yang berhubungan dengan pertanggung jawaban.

Komponen-komponen yang terdapat dalam sistem informasi akuntasi adalah sebagai berikut :

1. Orang-orang yang mengoperasikan sistem tersebut.

2. Prosedur-prosedur, baik manual maupun yang terotomatisasi yang dilibatkan dalam pengumpulan, pemrosessan dan penyimpaan data aktivitas-aktivitas organisasi.
3. Data tentang proses-proses bisnis.
4. Software yang dipakai untuk memproses data organisasi.
5. Infrastruktur teknologi informasi.

Didalam organisasi, sistem informasi akuntansi berfungsi untuk :

1. Mengumpulkan dna menyimpan aktivitas yang dilaksanakan disuatu organisasi, sumber daya yang dipengaruhi oleh aktivitas-aktivitas tersebut dan para pelaku aktivitas tersebut.
2. Mengubah data menjadi inforamasi yang berguna bagi manajemen.
3. Menyediakan pengendalian yang memadai. (Kusrini, 2007 ; 10-11)

II.2.1. Sisa Hasil Usaha (SHU)

Sisa hasil usaha (SHU) adalah gabungan dari hasil partisipasi neto dan laba atau rugi kotor dengan non anggota, ditambah atau dikurangi dengan pendapatan dan beban lain serta beban perkoperasian dan pajak penghasilan badan koperasi.

Sisa Hasil Usaha Koperasi merupakan pendapatan koperasi yang diperoleh dalam satu tahun buku dikurangi dengan biaya, penyusutan dan kewajiban lainnya termasuk pajak dalam tahun buku yang bersangkutan.(C A P S, 2011 ; 61)

II.2.2. Metode Langsung

Metode langsung menganalisa sumber kas (arus kas masuk) dan penggunaan kas (arus kas keluar) dari nilai pendapatan operasi dan beban – beban sehubungan operasi yang tersaji dalam laporan laba/rugi. (Glorida Karyawati P, 2013; 104)

II.3. Visual Basic 2010

Visual Basic 2010 merupakan lingkungan pengembangan terintegrasi atau biasa disebut IDE yang dikembangkan berbasis bahasa pemrograman *BASIC*. Bahasa *BASIC* sendiri sebenarnya sudah lama dikembangkan oleh *Microsoft Corporation* dengan nama *Microsoft Quick Basic*. Kesederhanaan sintaks dan fleksibelitas Bahasa Basic yang menyebabkan bahasa pemrograman in berlaku fenomenal sehingga banyak disukai dan pakai oleh *programmer* diseluruh dunia.

Berbekal kepopuleran tersebut *Microsoft* mengembangkan bahasa Basic in menjadi produk yang sangat terkenal di kalangan *programmer*, yaitu mulai *Microsoft Visual Basic 6.0* sampai sekarang, yaitu *Microsoft Visual Basic 2010*. Perkembangan teknologi dan penambahan banyak sekali fitur pada *Visual Basic 2010* tidak mengakibatkan adanya perubahan sintaks-sintaks dasar yang terdapat didalamnya. Sehingga dapat dikatakan untuki menjadi *programmer Visual Basic 2010* yang sebenarnya, anda diharuskan menguasai dan mengerti bagaimana menggunakan dan mengimplementasikansintaks dasar dalam bahasa pemrograman *basic*. (Wahana Komputer, 2010 ; 36)

II.4. Basis Data

Istilah basis data banyak menimbulkan interprestasi yang berbeda. Pada saat maraknya perangkat lunak dBASE II dan dBase II plus. Sebuah berkas (dengan ekstensi .DBF) biasa disebut basis data. Istilah yang tidak tepat ini meskipun telah merasuk ke sejumlah pemrograman, akhirnya diluruskan kembali oleh pencipta perangkat lunak basis data yang lain. Chou mendefinisikan basis data sebagai kumpulan informasi bermanfaat yang diorganisasikan ke dalam tata cara yang khusus. Menurut Fabbri dan Schwab, basis data adalah sistem berkas terpadu yang dirancang terutama untuk meminimalkan pengulangan data. Menurut Date, basis data dapat dianggap sebagai tempat untuk sekumpulan berkas data terkomputerisasi.

Menurut Date, sistem basis data pada dasarnya sistem terkomputerisasi yang tujuan utamanya adalah memelihara informasi dan membuat informasi tersebut tersebut tersedia saat dibutuhkan. (Abdul Kadir; 2006; 9)

II.4. 1. Kamus Data

Karena DBMS menyimpan kumpulan beberapa item data yang terpisah yang dapat digunakan pemakai pada beberapa aplikasi secara bersama-sama, adalah penting bahwa beberapa mekanisme digunakan untuk menyimpankan informasi mengenai mengenai beberapa item data bersangkutan. Itu adalah fungsi dari kamus data.

Kamus data adalah suatu file yang terpisah yang menyimpan informasi seperti :

a. Nama setiap item/jenis/kolom data.
b. Struktur data untuk tiap item.
c. Program yang menggunakan tiap item
d. Tingkat keamanan untuk setiap item

(Sumber : Zulkifli A.M; 2005; 382)

II.4.2. Normalisasi

Redudansi data cenderung melebihi ukuran dari basis data dan itu menjadi sebuah masalah yang sangat serius dalam media basis data yang besar. Selain itu, redudansi data bisa mendorong kea rah yang tidak normal (*anomalies*). Untuk memahami permasalahan yang bisa mengakibatkan pemborosan (*redudansi*), kita harus mengetahui pengertian redudansi lebih dekat lagi.

Sebelumnya, mari kita mulai dengan memgamati bahwa atribut dari skema tabel bisa digolongkan ke dalam tiga kelompok :

1. Atribut yang digunakan untuk tujuan identifikasi.
2. Atribut yang digunakan untuk tujuan informasi
3. Atribut yang digunakan untuk tujuan keduanya, baik identifikasi maupun informasi.

a. Bentuk Tidak Normal.

untuk membuat perancangan basis data, Anda pasti sudah mengenal sejumlah bentuk khusus, sifat-sifat, atau batasan skema tabel yang dimiliki untuk mencapai suatu tujuan yang diinginkan, misalnya memperkecil redudansi. Bentuk itu disebut “Bentuk Normal”

ada enam bentuk normal yang dikenal yaitu :

1. Bentuk Normal Pertama (1NF)
2. Bentuk Normal Kedua (2NF)
3. Bentuk Normal Ketiga (3NF)
4. Bentuk Normal Boyce Codd (BCNF)
5. Bentuk Normal Keempat (4NF)
6. Bentuk Normal Kelima (5NF)

Bentuk normal pertama sampai ketiga (dibuat oleh E.F. Codd) merupakan bentuk normal yang umum dipakai. Maksudnya, pada kebanyakan relasi, persoalan anomaly tidak akan muncul lagi bila ketiga bentuk normal tersebut telah terpenuhi.

b. Bentuk Normal Pertama (1 NF)

Bentuk normal pertama sangat sederhana. skema tabel disebut dalam bentuk normal pertama jika nilai atribut tidak terpisahkan. Untuk mengilustrasikannya, semua penulis buku di dalam atribut tunggal disebut penulis.

Berikut sebuah contoh :

ISBN = 979 -763-120-6

Judul = Basis Data

Penulis = Janner Simarmata Dan Iman Prayudi

Penerbit = Andi Yogyakarta

Oleh karena itu skema dalam kasus ini mengizinkan lebih dari satu nama penulis sebagai atribut penulis, maka skema bukan dalam bentuk normal pertama. Tentu saja, salah satu dari permasalahan yang jelas nyata dengan atribut penerbit adalah ketidakmungkinan mengurutkan data menggunakan nama penulis individu. Akan lebih sulit juga, sebagai contoh, bila menyiapkan label surat untuk setiap penulis dan seterusnya.

c. Bentuk Normal Kedua (2 NF)

Berdasarkan skema tabel T memiliki bentuk normal kedua jika semua atribut informasi (atribut yang tidak memiliki kunci mana pun) adalah atribut dari entitas lain di dalam skema tabel dan bukan dari kelas entitas lainnya. Dengan kata lain, atribut informasi menyediakan informasi secara rinci tentang entitas di dalam kelas entitas itu dan bukan beberapa entitas lain.

Berikut ilustrasi disertai sebuah contoh. Pertimbangkanlah skema table yang disederhanakan yang dirancang untuk menyimpan alamat rumah. Satu kemungkinan adalah :

{Kota, Jalan, NomorRumah, WarnaRumah, PopulasiKota}

Atribut PopulasiKota tidaklah pada tempatnya kerena merupakan atribut dari kota, bukan alamat rumah. Lelbih rincinya, **PopulasiKota** lebih kuat bila menjadi sebuah atribut informasi (bukan untuk identifikasi rumah)’ Atribut tersebut memberikan informasi tentang kota, bukan alamat rumah. Kemudian, skema table bukan berada dalam bentuk normal kedua. Maksud dari bentuk normal kedua kurang lebih adalah sebagai berikut.

Mengacu pada contoh sebelumnya, kita memiliki ketergantungan :

{Kota} => {PopulasiKota}

d. Bentuk Normal Ketiga (3 NF)

Bentuk normal kedua adalah baik, tetapi kita bisa melakukan yang lebih baik lagi. Kita sudah melihat bahwa jika skema tabel berbentuk normal kedua, maka bukan informasinya yang kuat, atribut tergantung pada subset yang sesuai dengan kunci. Bagaimanapun juga, ada kemungkinan lain yang tidak diinginkan. Lihat contoh berikut :

Asumsi dan skema tabel berikut adalah untuk tujuan ilustrasi, yakni bahwa tidak ada dua buku dengan judul yang sama dengan penerbit yang sama :

{Judul, KdPenerbit, JmlHalaman, Harga}

Satu-satunya kunci untuk skema tabel itu adalah **{Judul, KdPenerbit, JmlHalaman,** dan **Harga** hanya lah atribut informasi.

e. Bentuk Normal Boyce Codd (BCNF)

BCNF merupakan bentuk normal sebagai perbaikan terhadap 3 NF. Suatu relasi BCNF selalu memenuhi 3NF, tetapi tidak seaiknya. Suatu relasi yang memenuhi 3 NF belum tentu memenuhi BCNF. Dalam banyak literature disebutkan bentuk normal ketiga pun mungkin masih mengandung anomali

sehingga masih perlu dinormalisasikan selanjutnya.

Dependensi berikut mungkin bisa menentukan skema table di dalam bentuk normal ketiga, tetapi masih tetap memiliki redundansi. Pertimbangkan skema table {Kota, NmJalan, KodePos} dengan dependensi :

{Kota, NmJalan} => {KodePos}

Dan

{KodePos} => {Kota}

(Dalam kehidupan nyata, suatu kode pos bisa terbagi menjadi dua kota yang berbeda. Kita akan mengasumsikannya tidak untuk tujuan ilustrasi). Skema table itu berada dalam bentuk normal ketiga. Untuk melihatnya, amati bahwa kunci adalah **{Kota, NmJalan} dan {KodePos, NmJalan}.** Oleh karena itu bukanlah atribut dengan informasi yang kuat, dan disana tidak ada apa pun untuk melanggar bentuk normal ketiga (Janner Simarmata; 2007: 74-84).

II.4.3. ERD (*Entity Relational Database*)

Suatu model data adalah suatu penyajian konseptual dari suatu data yang diperlukan oleh basis data. Struktur data meliputi objek data, asosiasi antarobjek data, dan aturan yang memerintah operasi pada objek. Seperti yanmg tersirat pada namanya, model data berfokus pada data apa yang diperlukan dan bagaimana data tersebut harus diorganisasikan, alih-alih pada apa yang dilakukan pada data tersebut. Sebagai analogi, model data setara dengan gambar perencanaan yang dibuat oleh seorang arsitek.

Suatu model data todak terkait pada perangkat keras atau perangkat lunak. Dari pada mencoba menyajikan data sebagai basis data yang akan terlihat, model data berfokus untuk mewakili data yang dilihat pengguna di "dunia nyata". Model data bertindak sebagai jembatan antara konsep yang menyusun dunia nyata dan proses serta tampilan fisik dari konsep tersebut di dalam suatu basis data (Janner Simarmata; 2007: 94).

II.4.3.1. Langkah- Langkah Perancangan ER

1. Memilih kelompok atribut yang sama untuk dijadikan sebuah entitas dan menentukan kunci utama dengan syarat unik serta bisa mewakili entitas.
2. Menggambarkan kardinalitas (*cardinality*) dan diagram ER berdasarkan relasi yang diperlukan. Relasi yang terjadi adalah hubungan satu- kesatu, satu- kebanyak, dan banyak- kebanyak.
3. Membentuk Skema database atau LRS (*Logical Record Structure*) berdasarkan diagram ER.
    a. Jika relasinya satu- kesatu, maka *foreign key* diletakkan pada salah satu dari 2 entitas yang ada atau menyatukan kedua entitas tersebut.
    b. Jika relasinya satu- kebanyak, maka *foreign key* diletakkan pada entitas Many.
    c. Jika relasinya banyak- kebanyak, maka dibuat "file konektor" pada berisi 2 *foreign key* yang berasal dari kedua entitas.
4. Membentuk beberapa tabel berdasarkan primary key yang terpilih dengan syarat sudah mencapai aturan normalisasi sekurang-kurangnya 3 NF dari skema DB/LRS yang ada (Janner Simarmata; 2007: 115).

II.5. *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 II.1. sebagai berikut (Wahana Komputer; 2008 : 40).

Gambar II.1. *SQL Server 2008*

Sumber : Wahana Komputer (2008 : 40)

II.5.1. *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 *server*-nya 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

server-nya 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.5.2. 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 perinth 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 menngerti 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).

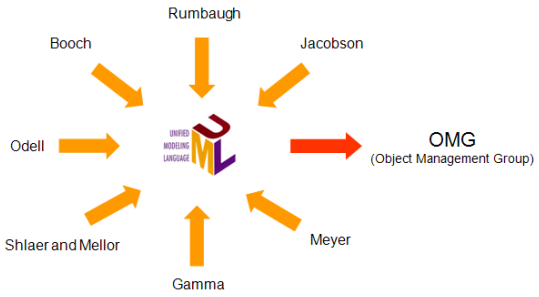
II.6. UML *(Unified Modelling Language)*

UML (*Unified Modelling Language*) adalah salah satu alat bantu yang sangat handal didalam dunia pengembangan sistem yang berorientasi obyek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

UML merupakan kesatuan dari bahasa pemodelan yang dikembangan oleh Booch, *Object Modelling Techniqu* (*OMT*) dan *Object Oriented Software Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadi proses analisis dan desain ke dalam empat tahapan iterative, yaitu : identifikasi kelas-kelas dan

obyek-obyek, identifikasi semantic dari hubungan obyek dan kelas tersebut, perincian interface dan implementasi. Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan *entity-relationship.* Tahapan utama dalam metodologi ini adalah analisis, design sistem , design obyek dan implementasi. (Munawar, 2005 ; 17 - 19)

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen yang tidak praktis ditambahkan dengan elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspesif dan seragam daripada metode lainnya. Gambar berikut adalah unsur-unsur yang membentuk UML.



Gambar II.2. UML Sebagai Bahasa Standar Pemodelan Aplikasi OOP
Sumber : (Sri Dharwiyanti: 2006 ; 3)

II.6.1. Konsepsi Dasar UML

Dari berbagai penjelasan rumit yang terdapat di dokumen dan buku-buku UML. Sebenarnya konsepsi dasar UML bisa kita rangkumkan dalam gambar dibawah.

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	static view	class diagram	class, association, generalization, dependency, realization, interface
	use case view	use case diagram	use case, actor, association, extend, include, use case generalization
	implementation view	component diagram	component, interface, dependency, realization
	deployment view	deployment diagram	node, component, dependency, location
dynamic	state machine view	statechart diagram	state, event, transition, action
	activity view	activity diagram	state, activity, completion transition, fork, join
	interaction view	sequence diagram	interaction, object, message, activation
		collaboration diagram	collaboration, interaction, collaboration role, message
model management	model management view	class diagram	package, subsystem, model
extensibility	all	all	constraint, stereotype, tagged values

Gambar.II.3. Konsep Dasar UML
(Sumber : Sri Dharwiyanti, 2006 ; 3)

Seperti juga tercantum pada gambar diatas UML mendefinisikan diagram-diagram sebagai berikut:

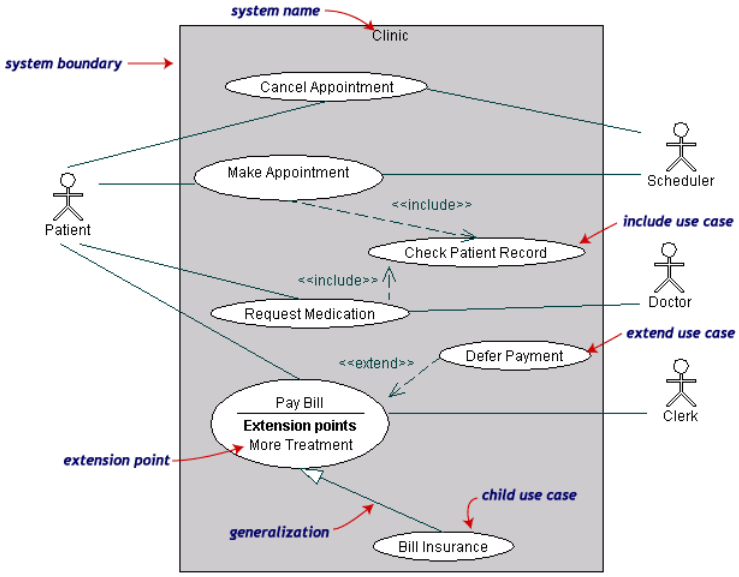
1. use case diagram
2. class diagram
3. statechart diagram
4. activity diagram
5. sequence diagram
6. collaboration diagram
7. component diagram
8. deployment diagram

Dalam pembuatan skripsi ini penulis menggunakan diagram *Use Case* yang terdapat di dalam UML. Adapun maksud dari *Use Case Diagram* diterangkan dibawah ini.

1. Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah "apa" yang diperbuat sistem, dan bukan "bagaimana". Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem. Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*. Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Contoh *use case diagram* :



Gambar.II.4.Use Diagram
(Sumber : Sri Dharwiyanti, 2006 ; 5)

2. Class Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

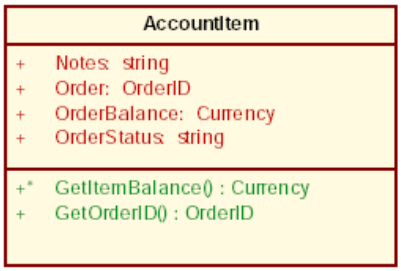
Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.

Class memiliki tiga area pokok :

1. Nama (dan *stereotype*)
2. Atribut
3. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
c. *Public*, dapat dipanggil oleh siapa saja



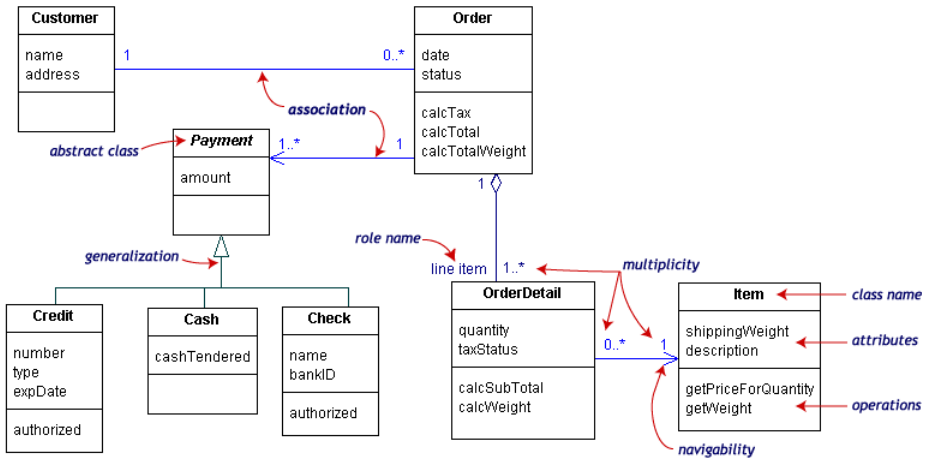
Gambar.II.5.Class Diagram
(Sumber : Sri Dharwiyanti, 2006 ; 5)

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class.* Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*. (Sri Dharwiyanti, 2005; 5)

a. Hubungan Antar *Class*

1. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. Panah *navigability* menunjukkan arah *query* antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).

3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian.



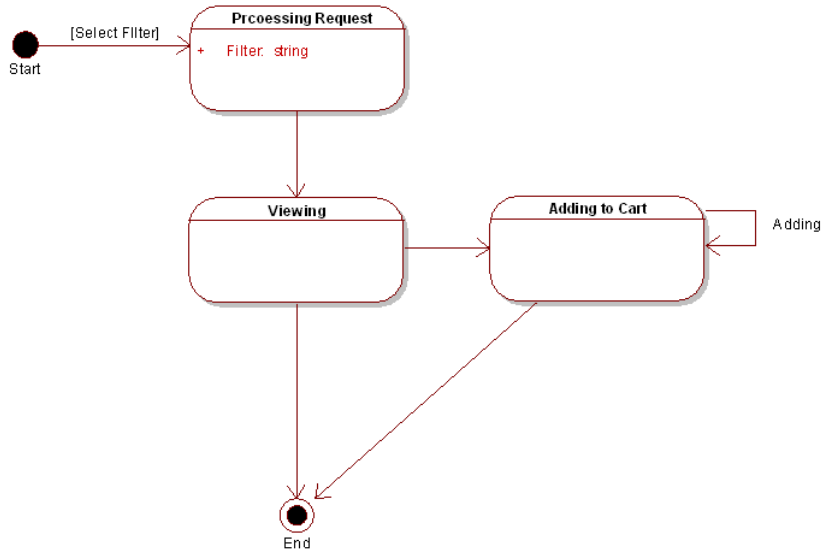
Gambar.II.6.Hubungan antar Class
(Sumber : Sri Dharwiyanti, 2006 ; 6)

3. Statechart Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.

Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.



Gambar.II.7. State Diagram
(Sumber : Sri Dharwiyanti, 2006 ; 7)

4. Activity Diagram

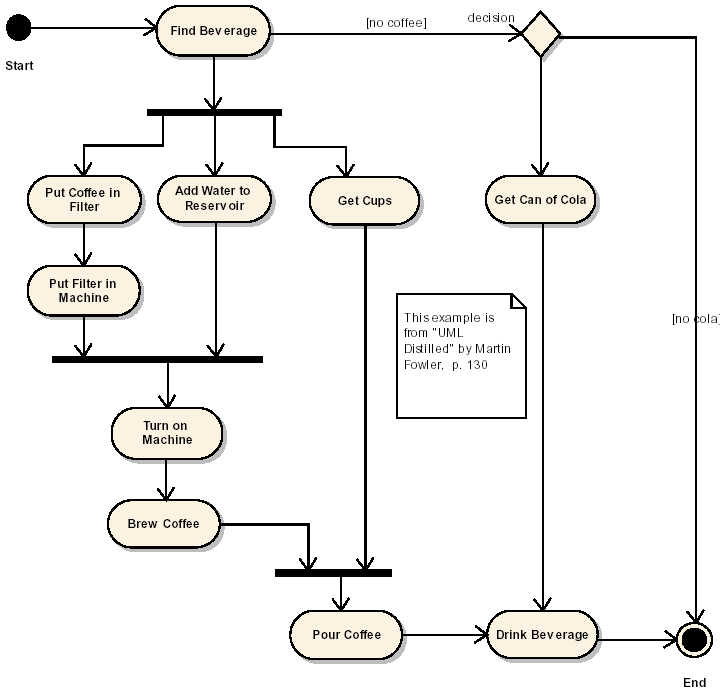
Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas.

Sama seperti *state*, standar UML menggunakan segiempat dengan sudut membulat untuk menggambarkan aktivitas. *Decision* digunakan untuk menggambarkan behaviour pada kondisi tertentu. Untuk mengilustrasikan proses-proses paralel (*fork* dan *join*) digunakan titik sinkronisasi yang dapat berupa titik, garis horizontal atau vertikal.

Activity diagram dapat dibagi menjadi beberapa *object swimlane* untuk menggambarkan objek mana yang bertanggung jawab untuk aktivitas tertentu.



Gambar.II.8. Contoh Activity Diagram State Diagram
(Sumber : Sri Dharwiyanti, 2006 ; 7)

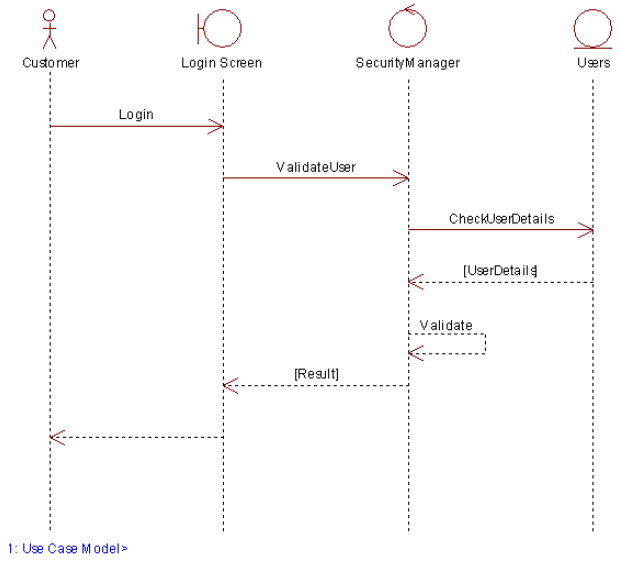
5. Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri atar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).

Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah *event* untuk menghasilkan *output* tertentu. Diawali dari apa yang men-*trigger* aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan *output* apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase

desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*.*Activation bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah message. Untuk objek-objek yang memiliki sifat khusus, standar UML mendefinisikan *icon* khusus untuk objek *boundary*, *controller* dan *persistent entity*.



Gambar.II.9. Contoh Activity Diagram State Diagram
(Sumber : Sri Dharwiyanti, 2006 ; 8)