BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

II.1.1. Sistem

Secara sederhana, suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau *variable* yang terorganisir, saling berinteraksi, saling tergantung satu sama lain dan terpadu. Selain itu suatu sistem tidak bisa lepas dari lingkungan sekitarnya maka umpan balik (*feed back*) dapat berasal dari lingkungan sistem yang dimaksud. (Tata Sutabri : 2012 : 10).

Suatu sistem mempunyai karakteristik atau sifat-sifat tertentu, yaitu:

1. Komponen Sistem (*Components*)

Suatu sistem terdiri dari sejumlah komponen-komponen yang saling berinteraksi, artinya saling bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem.

2. Batas Sistem (*Boudary*)

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

3. Lingkungan Luar Sistem (*Environtment*)

Bentuk apapun yang ada di luar ruang lingkup atau batasan sistem yang mempengaruhi operasi sitem tersebut disebut lingkungan luar sistem. Lingkungan

luar sistem ini dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem (*Interface*)

Media yang menghubungkan sistem dengan subsistem lain disebut penghubung sistem atau *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lain.

5. Masukan Sistem (*Input*)

Energi yang dimasukkan ke dalam sistem disebut masukan sistem, yang dapat berupa pemeliharaan (*maintenance input*) dan sinyal (*signal input).*

6. Keluaran Sistem (*Output*)

Hasil energi yang diolah dan diklasifikasikan menjadi pengeluaran yang berguna. Keluaran ini merupakan masukan bagi subsistem yang lain.

7. Pengolahan Sistem (*Proses*)

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

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 : 21).

II.1.2. Informasi

Informasi adalah data yang telah diklasifikasikan atau diolah atau diinterpretasikan untuk digunakan dalam proses pengambilan keputusan. Informasi dapat mengenai data mentah, data tersusun, kapasitas sebuah saluran komunikasi dan lain sebagainya. (Tata Sutabri : 2012 : 29).

II.1.3. Sistem Informasi

Sistem informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar dengan laporan-laporan yang diperlukan (Tata Sutabri : 2012 : 46).

II.2. Sistem Informasi Akuntansi

II.2.1. Akuntansi

Akuntansi adalah sebuah sistem informasi yang menghasilkan informasi keuangan kepada phak-pihak yang berkepentingan mengenai aktivitas ekonomi dan kondisi suatu perusahaan. Hasil dari suatu proses akuntansi disebut dengan laporan keuangan. Informasi yang dihasilkan dari prose akuntansi tersebut harus dapat menjawab kebutuhan umum para pemakainya. Karena itu, laporan keuangan suatu badan usaha harus memiliki kualitas yang diperlukan oleh berbagai pihak yang memerlukan informasi keuangan tersebut. (Rudianto : 2009 : 4).

II.2.2. Sistem Informasi Akuntansi

Sistem informasi akuntansi adalah sebuah sistem informasi yang menangani segala sesuatu yang berkenaan dengan akuntansi. Sistem Informasi Akuntansi (SIA) merupakan suatu kerangkan pengkordinasian sumber daya (data, *materials, equipment, suppliers, personal and funds*) untuk mengkonversi *input* berupa data ekonomik menjadi keluaran berupa informasi keuangan yang digunakan untuk melaksanakan kegiatan suatu entitas dan menyediakan informasi akuntansi bagi pihak-pihak yang berkepentingan. (Tata Sutabri : 2012 : 83).

II.2.3. Tujuan Sistem Informasi Akuntansi

Lingkup sistem informasi akuntansi dapat dijelaskan dari manfaat yang didapat dari informasi akuntansi. Manfaat atau tujuan sistem informasi akuntansi tersebut adalah sebagai berikut :

1. Mengamankan harta/kekayaan perusahaan.
2. Menghasilkan beragam informasi untuk mengambil keputusan.
3. Menghasilkan informasi untuk pihak *eksternal*.
4. Menghasilkan informasi untuk penilaian kinerja karyawan atau divisi.
5. Menyediakan data masa lalu untuk kepentingan audit (pemeriksaan).
6. Menghasilkan informasi untuk penyusunan dan evaluasi anggaran perusahaan.
7. Menghasilkan informasi yang diperlukan dalam kegiatan perencanaan dan pengendalian.

II.3. Bagian Akuntansi

II.3.1. Penjualan

Jurnal khusus penjualan adalah buku harian yang hanya digunakan untuk mencatat transaksi penjualan produk perusahaan secara kredit. Penjualan produk perusahaan secara tunai tidak dicatat dibuku harian ini. Demikian pula aktivitas penjualan aktiva perusahaan selain produk perusahaan tidak dapat ditampung dibuku harian ini. (Rudianto : 2009 : 136).

II.3.2. Penerimaan Kas

Perusahaan dagang dapat memperoleh penerimaan dari beberapa sumber yang terkait dengan operasi perusahaan Sumber penerimaan yang paling sering muncul adalah dari penjualan tunai dan dari penerimaan piutang . Selain dari kedua sumber tersebut memang masih terdapat sumber penerimaan dari transaksi lain seperti penjualan aktiva tetap, pembagian *deviden* dari investasi jangka panjang, penjualan surat berharga dan lain-lain. (Rudianto : 2009 : 137).

II.3.3. Pembelian

Perusahaan dagang dapat melakukan penjualan jika perusahaan tersebut telah membeli produk tersebut dari produsen atau dari *supplier* lain. Pembelian barang dagangan dapat dilakukan secara tunai maupun secara kredit. Pembelian yang dilakukan secara kredit mengharuskan akuntan perusahaan mencatat transaksi tersebut di sisi debet akun pembelian dan di sisi kredit akun utang usaha.

Buku harian pembelian adalah buku harian yang digunakan hanya untuk mencatat transaksi pembelian barang dagangan secara kredit. Buku harian ini tidak digunakan untuk mencatat aktivitas pembelian perlengkapan kantor, peralatan kantor, aktiva tetap, surat berharga dan lainnya.(Rudianto : 2009 : 138).

II.3.4. Pengeluaran Kas

Salah satu aktivitas yang harus dan selalu dilakukan perusahaan dagang adalah aktivitas pengeluaran kas untuk berbagai alasan. Pengeluaran kas tersebut dapat digunakan untuk membayar utang, melakukan pembelian secara tunai, membayar berbagai macam beban operasi, ataupun untuk berbagai keperluan lain. Apapun alasan pengeluaran kas tersebut, akun kas akan di kredit sebesar nilai transaksi dan di sisi debetnya disesuaikan dengan aktivitas transaksi.

Buku harian khusus pengeluaran kas adalah buku harian yang digunakan khusus untuk mencatat transaksi pengeluaran kas untuk berbagai keperluan. Baik pengeluaran kas untuk membayar utang, pembelian barang dagangan secara tunai, membayar berbagai macam beban operasi maupun untuk berbagai keperluan yang ada (Rudianto : 2009 : 138). Poin penting dalam siklus pengeluaran dapat dispesifikasikan ke dalam bagan sebagai berikut :

Tabel II.1 Siklus Pengeluaran Kas

AKTIVITAS PENGENDALIAN	SISTEM PEMROSESAN PEMBELIAN	SSISTEM PENGELUARAN KAS
Otorisasi Transaksi	Pengendalian persediaan	Bagian utang usaha mengotorisasi pembayaran
Pemisahan Pekerjaan	Pengendalian persediaan dipisahkan dari bagian pembelian dan penyimpanan persediaa. Buku besar utang usaha terpisah dari buku besar	Pisahkan bagian buku besar pembantu utang usaha, pengeluaran kas, dan buku besar.
Supervisi	Bagian penerimaan	
Catatan Akuntansi	Buku pembantu utang usaha, buku besar, file permintaan pemmbelian, file pesanan pembelian, file laporan penerimaan.	File voucher utang, buku pembantu utang usaha, jurnal pengeluaran kas, akun kas di buku besar.
Akses	Keamanan fisik aktiva. Batasi akses hanya ke catatan akuntansi di atas	Keamanan yang memadai atas kas. Batasi akses ke berbagai catatan akuntansi diatas.
Verifikasi independen	Bagian utang dengan merekonsiliasi berbagai dokumen sumber sebelum mencatat kewajiban. Bagian buku besar merekonsiliasi akurasi umum proses tersebut.	Peninjauan akhir oleh bagian pengeluaran kas. Rekonsiliasi keseluruhan oleh bagian buku besar. Rekonsiliasi bank secara berkala oleh kontroler.

(Sumber : Marshall B.Romney : 2010:10)

II.4. PHP

PHP singkatan dari *PHP Hypertext Preprocessor* yang digunakan sebagai bahasa *script server-side* dalam pengembangan *web* yang disisipkan pada dokumen *HTML*.

Penggunaan *PHP* memungkinkan *web* dapat dibuat dinamis sehingga *maintenance* situs *web* tersebut menjadi lebih mudah dan efisien. *PHP* merupakan

software Open-Source yang disebarkan dan dilisensikan secara gratis serta dapat di download secara bebas dari situs resminya.

PHP diciptakan pertama kali oleh Rasmus Lerdorf pada tahun 1994. Awalnya, *PHP* digunakan untuk mencatat jumlah serta untuk mengetahui siapa saja pengunjung pada *homepage*-nya. Rasmus Lerdorf adalah salah satu pendukung *open source*. Oleh karena itu, ia mengeluarkan *Personal Home Page Tools* versi 1.0 secara gratis, kemudian menambah kemampuan *PHP 1.0* dan meluncurkan *PHP 2.0*.

Pada tahun 1996, *PHP* telah banyak digunakan dalam *website* di dunia. Sebuah kelompok pengembang *software* yang terdiri dari Rasmus, Zeew Suraski, Andi Gutman, Stig Bakken, Shane Caraveo, dan Jim Winstead bekerja sama untuk menyempurnakan *PHP 2.0*. Akhirnya, pada tahun 1998, *PHP 3.0* diluncurkan. Penyempurnaan terus dilakukan sehingga pada tahun 2000 dikeluarkan *PHP 4.0*.

Bagi pemula belajar *PHP*, istilah ini sangat penting untuk mendukung dalam pembuatan program, tetapi bagi yang telah belajer bahasa pemrograman sperti bahasa C, *Pascal* dan *Java*, istilah ini sudah sering didengar. Kita Perlu mengingat perbedaan tipe data, variabel, konstantaa dan operator yang terdapat dalam *PHP*. Tipe data merupakan jenis data, setiap data pasti memiliki jenis data tertentu. Jenis data berguna untuk menghindari ekspresi yang tipe datanya bercampur. Sebagai contoh ’*PHP*” + 4.0 (jenis data *string PHP* mau ditambah dengan tipe data *numerik* 4.0). PHP mengenal 3 kelompok jenis data yaitu :

1. Integer (bilangan bulat)

   Tipe data integer merupakan tipe data yang direpresentasikan berbentuk angka dan tidak memakai tanda titik desimal. Contoh : $angka=2-;

2. Tipe data Double (bilangan pecahan)

   Tipe data double merupakan tipe data yang direpresentasikan memakai tanda titik desimal. Contoh : $angka=20.50;

3. Tipe data String (teks)

   Tipe data string merupakan tipe data yang direpresentasikan diapit tanda petik tunggal (’ ’) atau petik ganda (” ”). Contoh : $kantor=”CV.Noornet”; (Ir.Yuniar Supardi :2010 : 17-18)

II.4.1. Keunggulan PHP

PHP memiliki banyak kelebihan yang tidak dimiliki oleh bahasa *script* sejenis *PHP* difokuskan pada pembuatan *script server-side* yang bisa melakukan apa saja yang dapat dilakukan oleh *CGI*, seperti mengumpulkan data dari *form*, menghasilkan isi halaman *web* dinamis, dan kemampuan mengirim serta menerima *cookies*, bahkan lebih daripada kemampuan *CGI*.

Berikut ini adalah beberapa keunggulan dari bahasa pemrograman PHP yaitu :

1. Bahasa Pemrograman *PHP* mendukung komunikasi dengan layanan seperti *protocol IMAP, SNMP, NNTP, POP3* bahkan *HTTP*.
2. *Securiry:* Tingkat keamanan yang cukup tinggi dan Stabil.
3. *Access:* Akses ke sistem *Database* yang lebih *fleksibel*, seperti *MySQL*.

4. *Easy & Faster*: Dalam sisi pemahaman, *PHP* adalah bahasa *scripting* yang paling mudah karena memiliki *referensi* yang banyak dan berkecepatan tinggi.
5. *Cross platform* yaitu *PHP* dapat berjalan lintas *platform*, yaitu dapat berjalan dalam sistem operasi seperti *Windows, Linuz, MacOS* dan *OS* lainnya dan *web server* apapun.
6. *Free*: Dapat digunakan secara gratis.
7. Termasuk bahasa yang *embedded***,** yakni dapat diletakkan dalam *tag HTML*.
8. Termasuk Jenis *server side programming*, sehingga kode asli/*source code PHP* tidak dapat dlihat di *browser* pengguna, yang terlihat hanya kode dalam format *HTML*.
9. Dapat memanfaatkan sumber-sumber aplikasi yang dimiliki oleh *server* misalnya untuk keperluan *Database connection*.
10. *PHP* dapat melakukan semua aplikasi program CGI, seperti mengambil nilai *form*, menghasilkan halaman *web* yang dinamis, mengirimkan dan menerima *cookies*.
11. *On The Fly***:** *PHP* sudah mendukung *on the fly*, artinya dengan *php* anda dapat membuat *document text, Word, Excel, PDF*, menciptakan *image* dan *flash*, juga menciptakan *file-file* seperti *zip, XML*, dan banyak lagi.
12. Dalam sisi pengembangan lebih mudah, karena banyaknya *milis - milis* dan *developer* yang siap membantu dalam pengembangan.

Salah satu fitur yang dapat diandalkan oleh *PHP* adalah dukungannya terhadap banyak *database*, seperti *Adabas D, dBase, Direct MS-SQL, Empress,*

FilePro, FrontBase, Hyperware, IBM DB2, Informix, MSQL, MySql dan masih banyak lagi (Dadan Sutisna : 2008 : 40).

II.4.2. Kekurangan PHP

PHP juga tidak luput dari kekurangan sehingga sesungguhnya PHP sendiri masih perlu banyak perbaikan untuk pengembangannya. Berikut adalah beberapa kekurangan dari PHP yaitu:

1. Tidak detail untuk pengembangan skala besar
2. Tidak memiliki system pemrogaman berorientasi objek yang sesungguhnya.
3. Tidak bisa memisahkan antara tampilan dengan logic dengan baik.
4. PHP memiliki kelemahan security tertentu apabila programmer tidak jeli dalam melakukan pemrogaman dan kurang memperhatikan isu konfigurasi PHP.
5. Kode PHP dapat dibaca orang, dan kompilasi hanya dapat dilakukan dengan tool yang mahal dari Zend.

II.5. MySQL

MySQL adalah salah satu jenis *database server* yang sangat terkenal. Kepopulerannya desebabkan *MySQL* menggunakan *SQL* sebagai bahasa dasar untuk mengakses *database*nya. Selain itu, ia bersifat *Open Source* (Anda tidak perlu membayar untuk menggunakannya. Database merupakan kumpulan data yang didalamnya terdapat tabel-tabel. Jika kita berbicara mengenai database, sebenarnya mengacu pada bentuk data relational yang terdiri dari baris

(row/record) dan kolom (colomn/field). Perangkat lunak PHP mendukung perangkat lunak database konvensional hingga database modern. Database MYSQL tergolong database server, PHP sangat serasi denga server web apache dan database MYSQL. (Ir.Yuniar Supardi :2010:154).

Database digunakan untuk penyimpanan data, demikian pula dengan *MYSQL*. Kita akan memanggil data pada *MYSQL* melalui *PHP*, kemudian hasilnya dikirim kekomputer *klien* untuk ditampilkan pada *browser*. Data pada *MYSQL* dapat dipanggil, dihapus atau di tambah melalui *query* .(Dadan Sutisna : 2008 : 46)

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

Unified Modelling Language (UML) merupakan alat komunikasi yang konsisten dalam mensupport para pengembang sistem saat ini. Sebagai perancang sistem, mau tidak mau pasti akan menjumpai *UML*, baik kita sendiri yang membuat atau sekedar membaca diagram *UML* buatan orang lain. (Prabowo Pudjo Widodo ; 2011 : 7)

II.6.1. Diagram – Diagram Pada Metode UML

1. Use Case Diagram

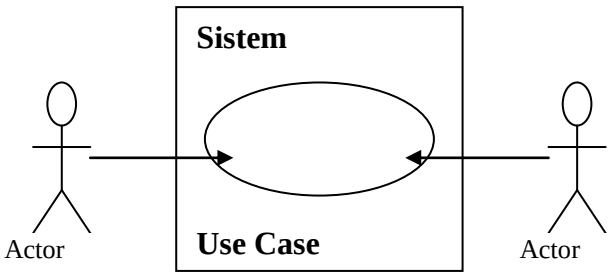
Use case adalah deskripsi fungsi dari sebuah sistem dari *perspektif* pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan

antara pengguna dan sistem yang disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras dan urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh pengguna tujuan umum pengguna.

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan *actor*. *Actor* adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan sistem.

Model *use case* adalah bagai dari model *requirement*. Termasuk disini adalah problem domain object dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari *persfectif user*.

Notasi *use case* menunjukkan 3 aspek dari sistem yaitu *actor use case* dan *system / sub system boundary*. *Actor* mewakili peran orang, *system* yang lain atau alat ketika berkomunikasi dengan *use case*. Ilustrasi *actor*, *use case* dan *system* ditunjukkan pada Gambar II.1 berikut ini:



Gambar II.1 : *Use Case* Diagram

(Sumber : Prabowo Pudjo Widodo ; 2011 : 17)

Untuk mengidentifikasi *actor*, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. *Actor* adalah *abstraction* dari orang dan sistem yang lain mengaktifkan fungsi dari target sistem. Orang atau sistem bila muncul dalam beberapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*.

Use case adalah abstraksi dari interaksi antara sistem dan *actor*. Oleh kerena itu sangat penting untuk memilih abstraksi yang cocok. *Use case* dibuat berdasarkan keperluan *actor*. *Use case* harus merupakan ’apa’ yang dikerjakan *software* aplikasi, bukan ’bagaimana’ *software* aplikasi mengerjakannya. Setiap use case harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Namun *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama.

2. Activity diagram

Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku *paralel* sedangkan *flowchart* tidak bisa.

3. Class Diagram

Diagram kelas atau class diagram 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 Pudjo Widodo :2011 : 37).

Kelas memiliki apa yang disebut *Atribut* dan metode atau operasi :

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Susunan kelas suatu sistem yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

a. Kelas main, kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
b. Kelas yang menangani tampilan sistem, kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
c. Kelas yang diambil dari pendefinisian *use case*, kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.
d. Kelas yang diambil dari pendefinisian data, kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

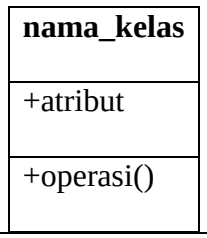
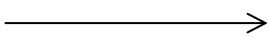
Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohension* dan *coupling*. *Cohension* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama

lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan yang lain dalam sebuah kelas.

Berikut adalah simbol-simbol yang ada pada diagram kelas, dapat dilihat pada Tabel II.2 berikut ini:

Tabel II.2 Simbol-Simbol *Class Diagram*

Simbol	Deskripsi
Kelas 	Kelas Pada struktur system
Antarmuka / *interface*  nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *association* 	Relasi antar kelas dengan maksna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *directed association* 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai dangan *multiplicity*
generalisasi 	Relasi antar kelas dengan makna

	generalisasi – spesialisasi (umum khusus)
Keberuntungan /*dependency* .....>	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi /*aggregation*	Relasi antar kelas dengan makna

(Sumber : Rosa A.S dan M. Shalahuddin : 2011 : 123)

4. *Squence Diagram*

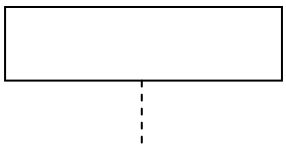
Squence Diagram digunakan untuk menggambarkan perilaku pada sebuah sekenario. Diagram ini menunjukkan sebuah contoh objek dan pesan yang diletakkan diantara objek-objek ini didalam *use case*.

Komponen utama *Squence diagram* terdiri dari atas objek yang dituliskan dengan kotak segiempat bernama. *Messege* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical.*

a. Objek / *participant*

Objek diletakkan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. Mereka diatur dalam urutan guna menyederhanakan diagram. Setiap *participant* dihubungkan garis titik-titik yang disebut *lifeline.* Sepanjang *lifeline* ada kotak yang disebut *activation*. *Activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation*. *Activation* mewakili sebuah eksekusi operasi dari

participant. Panjang kotak ini berbanding lurus dengan durasi *activation.* Bentuk *participant* dapat dilihat pada Gambar II.2 berikut ini:



Gambar II.2 : Bentuk *Participant*

(Sumber : Prabowo Pudjo Widodo ; 2011 : 175)

b. *Messege*

Sebuah *messege* bergerak dari suatu *participant* ke *participant* yang lain dan dari *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri.

Sebuah *message* bisa jadi *simple*, *synchronous* atau *asynchoronous*. *Message* yang *simple* adalah sebuah perpindahan (transfer), contoh dari satu *participant* ke *participant* yang lainnya. Jika suatu *participant* mengirimkan sebuah *message* tersebut akan ditunggu sebelum di proses dengan urusannya. Namun jika *message asynchoronous* yang dikirimkan, maka jawabannya atas *message* tersebut tidak perlu ditunggu. Simbol *message* pada *squnence diagram* dapat dilihat pada Gambar II.3. berikut ini:

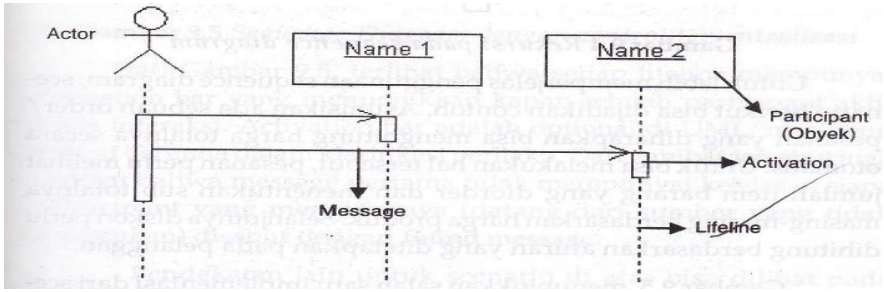
Gambar II.3 : Bentuk *Messege*

(Sumber : Prabowo Pudjo Widodo ; 2011 : 179)

c. *Time*

Time adalah diagram yang mewakili waktu pada arah vertikal. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibanding *message* yang lebih dekat kebawah.

Terdapat dua dimensi pada *squence diagram* yaitu dimensi dari kiri ke kanan menunjukkan tata letak participant dan dimensi dari atas ke bawah menunjukkan lintasan waktu. Simbol-simbol yang ada pada *squence diagram* ditunjukkan pada Gambar II.4 berikut ini:



Gambar II.4 : Bentuk *Time*

II.7. Konsep Sistem Database

Database adalah kumpulan data terhubung *(interrelated data)* yang disimpan secara bersama-sama pada suatu media, tanpa mengatap satu sama lain atau tidak perlu suatu kerangkapan data *(controlled redundancy)* dengan cara tertentu sehingga mudah digunakan atau ditampilkan kembali; dapat digunakan oleh satu atau lebih program aplikasi secara optimal; data disimpan tanpa mengalami ketergantungan pada program yang akan menggunakannya; data disimpan sedemikian rupa sehingga penambahan, pengambilan, dan modifikasi dapat dilakukan dengan mudah dan terkontrol. Sementara itu, sistem *database*

adalah sekumpulan database yang dapat dipakai secara bersama-sama, *personal-personal* yang merancang dan mengelola *database,* teknik-teknik untuk merancang dan mengelola *database,* serta *computer* untuk mendukungnya.

II.8. Normalisasi

Normalisasi merupakan proses pengelompokan elemen data menjadi tabel yang menunjukkan entitas sekaligus relasinya. (Ema Utami dan Anggit Dwi Hartanto ; 2012 : 40).

Tahap normalisasi terdiri dari beberapa bentuk :

1. 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 sesuai dengan kedatangannya.

2. Bentuk Normal Kesatu (1NF/ *First Normal Form*)

Bentuk normal kesatu mempunya ciri : setiap data dibentuk dalam *file file* (*file* datar/rata), data dibentuk dalam satu *record* demi *record* dan nilai dari *field* berupa ”atomic value”. Tidak ada set atribut yang berulang atau atribut bernilai ganda (*multivalue*). Tiap *field* hanya satu pengertian, bukan merupakan kumpulan kata yang mempunyai arti mendua, hanya satu arti saja dan juga bukan pecahan kata sehingga artinya lain. Atom adalah zat terkecil yang masih memiliki sifat induknya, bila dipecah lagi, maka ia tidak memiliki sifat induknya.

3. Bentuk Normal Kedua (2NF/ *Second Normal Form*)

Bentuk normal kedua memiliki syarat : bentuk data telah memenuhi kriteria bentuk normal kesatu. Atribut bukan kunci haruslah bergantung fungsi pada kunci utama/*primary key* sehingga untuk membentuk normal kedua haruslah sudah ditentukan kunci *field*. Kunci *field* haruslah unik dan dapat mewakili atribut lain yang menjadi anggotanya. Tahap normalisasi terdiri dari beberapa bentuk.

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

Untuk menjadi bentuk normal ketiga, relasi haruslah dalam bentuk normal kedua dan semua atriut dalam primer tidak punya hubungan yang transif. Dengan kata lain, setiap atribut bukan kunci haruslah bergantung hanya pada *primary key* dan pada *primary key* secara menyeluruh. Contoh pada bentuk kedua di atas termasuk juga bentuk normal ketiga seluruh atribut yang ada disitu bergantung penuh pada kunci primernya.

5. Bentuk Normal *Boyce Codd* (BCNF/ *Boyce Codd Normal Form*)

Boyce Codd Normal *Form* mempunyai paksaan yang lebih kuat dari pada bentuk normal ketiga. Untuk menjadi BCNF, relasi harus dalam bentuk normal kesatu dan setiap atribut harus bergantung fungsi pada atribut *superkey*.

II.9. *Entity Relationship Diagram* (ERD)

ERD merupakan notasi grafis dalam pemodelan data *konseptual* yang mendeskripsikan hubungan antar penyimpanan. *ERD* juga merupakan gambaran yang menghubungkan antara objek satu dengan objek yang lainnya dalam dunia nyata. (Ema Utami dan Anggit Dwi Hartanto ; 2012 : 18).

ERD menggunakan sejumlah notasi dan simbol untuk menggambarkan struktur dan hubungan antar dua data. Pada dasarnya ada 3 macam simbol yang digunakan, yaitu :

1. Entity

Entity adalah suatu objek yang dapat diidentifikasikan dalam lingkungan pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat. Sebagai contoh adalah barang, pemasok, pekerja dan lain-lain.

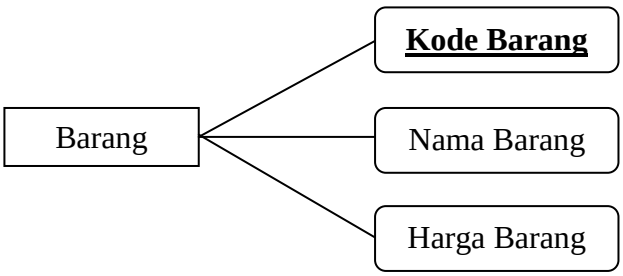
Seandainya adalah A maka barang A adalah isi dari barang, sedangkan jika B adalah seorang pelanggan maka B adalah isi dari pelanggan. Karena itu harus dibedakan antara entitas sebagai bentuk umum dari deskripsi tertentu dan isi entitas seperti A dan B dalam contoh diatas. Entitas dapat digambarkan dalam bentuk persegi empat. Dapat dilihat pada Gambar II.5 berikut ini:

Gambar II.5 : Entitas

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 19)

2. *Atribut*

Entitas mempunyai elemen yang disebut *atribut* dan berfungsi mendeskripsikan karakter *entitas*, misalnya atribut nama barang dari *entitas* barang. Setiap *ERD* bisa berisi lebih dari satu *atribut*. *Entitas* digambarkan dalam bentuk elips. Yang dapat dilihat pada Gambar II.6 berikut ini:

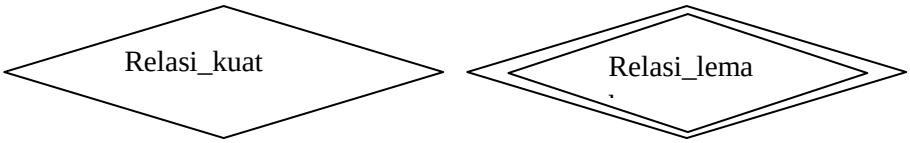


Gambar II.6 : Atribut

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 20)

3. Hubungan / *relationship*

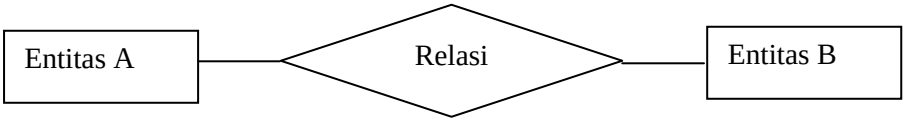
Belah ketupat merupakan penggambaran hubungan (relasi) antarentitas atau sering disebut kerelasian. Ada dua macam penggambaran relasi yaitu relasi kuat dan relasi lemah. Relasi kuat biasanya menghubungkan antarentitas kuat, sedangkan relasi lemah untuk menghubungkan antara entitas kuat dengan entitas lemah. *Relationship* digambarkan pada Gambar II.7 berikut ini:



Gambar II.7: *Relationship*

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 24)

Untuk menghubungkan entitas-kerelasian-entitas digunakan garis lurus seperti pada Gambar II.8 berikut ini :



Gambar II.8 : Kerelasian Antarentitas

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 24)

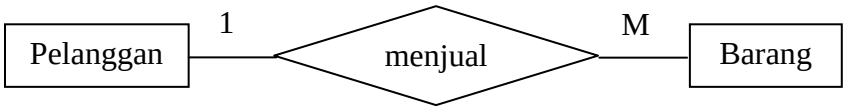
Jenis-jenis hubungan:

a. Satu ke satu, misalnya suatu perusahaan mempunyai aturan satu supir hanya boleh menangani satu kendaraan karena alasan tertentu. Seperti pda Gambar II.9 berikut ini:

Gambar II.9 : *Relational* 1 to 1

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 24)

b. Satu ke banyak atau banyak ke satu, misalya suatu perusahaan selalu berasumsi bahwa satu pelanggan dapat membeli banyak barang. Dapat dilihat pada Gambar II.10 berikut ini:



Gambar II.10 : *Relational* 1 to *Many*

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 25)