BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. Hal ini menjelaskan bahwa sistem bekerja dalam suatu prosedur yang saling berhubungan satu sama lain untuk menyelesaikan tujuan dan sasaran yang dimaksud. Sistem juga diartikan sebagai sekelompok elemen-elemen yang saling berinteraksi dengan maksud dan tujuan yang sama untuk melaksanakan sasaran yang telah ditentukan (Antonio, Safriadi, 2012:12). Selain itu, dapat dilihat bahwa sistem berusaha mencapai tujuan. Pencapaian tujuan ini menyebabkan timbulnya dinamika, perubahan yang terus menerus perlu dikembangkan dan dikendalikan.

Sistem Informasi adalah suatu sistem dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk menyediakan kepada pihak luar tertentu dengan informasi yang diperlukan untuk pengambilan keputusan (Antonio dan Safriadi, 2012:12).Sistem informasi mencakup sejumlah komponen (manusia, komputer, teknologi informasi, dan prosedur kerja), ada sesuatu yang diproses (data menjadi informasi), dan dimaksudkan untuk mencapai suatu sasaran atau tujuan.

Mengingat sistem informasi merupakan kumpulan dari beberapa elemen yang saling berhubungan untuk mencapai tujuan, maka sistem informasi terdiri dari beberapa komponen. Adapun komponen-komponen sistem informasi adalah sebagai berikut :

1. Perangkat keras (*hardware*), mencakup peranti-peranti fisik seperti komputer dan printer.
2. Perangkat lunak (*software*) atau program, yaitu sekumpulan instrukti yang memungkinkan perangkat keras untuk dapat 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 yangberkaitan dengan penyampaian data.
6. Jaringan komputer dan komunikasi data, yaitu sistem penghubung yang memungkinkansumber (*resources*) dipakai secara bersama atau diakses oleh sejumlah pemakai.

II.1.1. Siklus Sistem Informasi

Pada suatu sistem informasi terdapat siklus yang disebut dengan siklus makro.Siklus sistem informasi ini merupakan tahapan-tahapan dalan merancang dan mengembangkan dari sistem informasi itu sendiri. Dalam jurnalnya, Joefrie dan Kalatiku (2012:191) menjelaskan bahwa siklus makro meliputi beberapa tahapan, yaitu :

1. *Feasibility analysis*

   Tahap ini berhubungan dengan analisis area aplikasi potensial, mengidentifikasi sisi ekonomi dari pengambilan dan disseminasi, membentuk studi keuntungan awal, menentukan kompleksitas data dan proses mengatur prioritas aplikasi.

2. *Requirement collection and analysis*

   Kebutuhan detail deikumpulkan dengan iteraksi dengan pemakai potensial dan kelompok pemakai untuk mengidentifikasi permasalahan dan kebutuhan khusus. Ketergantungan aplikasi, komunikasi dan prosedur pelaporan diindetifikasi.

3. *System design*

   Tahap ini mempunyai dua aspek yaitu mendesain sistem basis data dan mendesain sistem aplikasi (program) yang menggunakan dan memproses basis data.

4. *Implementation*

   Sistem informasi yang baru dbuat diimplementasi, basis data diberntauk dan transaksi basis data diimplementasikan dan diujicobakan.

5. *Validation and acceptance*

   Tingkat akses dari sistem dalam memenuhi kebutuhan pemakai dan kriteria performansi divalidasi. Sistem diujicoba dengan kriteria performansi dan sesifikasi kelakukan.

6. *Deployment, operataion and maintenance*

   Pada tahap ini dilakukan konversi pemakai dari sistem lama ke sistem baru melalui training. Tahap operasional mulai jika semua fungsi sistem dioperasikan dan divalidasi. Jika kebutuhan baru atau aplikasi bertambah, maka harus melalui semua tahap sebelumnya sampai semua divalidasi dan berhubungan dengan sistem. Monitoring performansi sistem dan pemeliharaan sistem merupakan aktifitas yang penting selama tahap operasi.

II.2. Sistem Basis Data

Basis data adalah sekumpulan file. Definisi umum dari basis data adalah bahwa basis data merupakan kumpulan dari seluruh data berbasis komputer sebuah organisasi/perusahaan.Definisi basis data yang lebih sempit adalah bahwa basis data merupakan kumpulan data yang berada di bawah kendali peranti perangkat lunak sistem menajemen basis data (Raymond dan George, 2008:158). Menurut Joefrie dan Kalatiku (2012:191), basis data adalah bagian dari sistem informasi, di dalamnya termasuk sumber daya yang dilibatkan dalam koleksi, manajemen, penggunaan dan disseminasi sumber daya informasi dari organisasi. Sistem basis data *(database system)* adalah suatu sistem informasi yang mengintegrasikan kumpulan dari data yang saling berhubungan satu dengan yang lainnya dan membuatnya tersedia untuk beberapa aplikasi yang bermacam – macam di dalam suatu organisasi.

Sistem basis data terdiri dari beberapa komponen yang saling berhungan satu sama lain. Adapun komponen-komponen dasar pada sistem basis data adalah sebagai berikut:

1. Data

   Data di dalam sebuah basis data dapat disimpan secara terintegrasi dan dapat pula dipakai secara bersama-sama. Terdapat tiga jenis data, yaitu :

   a. Data Operasional
   b. Data Masukkan
   c. Data Keluaran

2. *Hardware* (Perangkat Keras)

   Terdiri dari semua peralatan komputer yang digunakan untuk pengelolaan sistem basis data yang berupa: peralatan untuk penyimpanan, peralatan masukkan dan keluaran, dan peralatan komunikasi data.

3. *Software*(Perangkat Lunak)

   Berfungsi sebagai perantara antara pemakai dengan data fisik pada basis data.Perangkat lunak dalam basis data berupa *Database Management System* (DBMS) atau program aplikasi prosedur.

4. *User* atau Pemakai

   Pemakai basis data dibagi atas tiga klasifikasi, yaitu :

   a. *Database Administrator* (DBA)

      Database Administartor (DBA) adalah orang atau tim yang bertugas untuk mengelola sistem basis data secara keseluruhan. Adapun tugas dari DBA adalah sebagai berikut :

      1) Mengontrol DBMS dan *software-software*.
      2) Memonitor siapa saja yang mengakses basis data.
      3) Mengatur pemakaian basis data.

4) Memeriksa keamanan, *integrity, recovery* dan *concurrency*.

b. *Programmer*

Programmer adalah orang atau tim yang bertugas membuat program aplikasi, misalnya untuk perbankan, administrasi dan lain-lain.

c. *End User*

End User adalah orang yang mengakses basis data melalui terminal dengan menggunakan *query language* atau program aplikasi yang telah dibuat oleh *programmer*.

II.2.1. *Entity Relationship Diagram*

Entity Relationship Diagram merupakan jaringan yang menggunakan susunan data yang disimpan dari sistem secara abstrak (Widyaestoeti, 2011:3). Tujuan dari *entity relationship diagram* ini adalah untuk menunjukkan objek data dan *relationship* yang ada pada objek tersebut. Ada beberapa alasan diperlukannya model *entity relationship*, yaitu :

1. Dapat menggambarkan hubungan antar entitas dengan jelas.
2. Dapat menggambarkan batasan jumlah entitas dan partisipasi antar entitas.
3. Mudah dimengerti oleh pemakai.
4. Mudah disajikan oleh perancang basis data.

Model *entity relationship* terbentuk karena didukung oleh beberapa komponen yang saling berhubungan satu sama lain, antara lain :

1. *Entity*

Entity adalah suatu yang dapat dibedakan dalam dunia nyata dimana informasi yang berkaitan dengannya dikumpulkan.Simbol *entity* adalah persegi panjang.

2. *Relationship*

   Merupakan hubungan yang terjadi antara satu atau lebih *entity*.

3. *Attribute*

   Attribute merupakan karakteristik dari *entity* atau *relationship* yang menyediakan penjelasan detail tentang hal tersebut. Nilai *attribute* adalah suatu data yang aktual.

4. Indikator Tipe

   Indikator tipe ada dua, yakni : indikator tipe *asscociative object* dan indikator tipe supertipe.

5. *Cardinality Rasio*

   Menjelaskan hubungan batasan jumlah keterhubungan satu *entity* dengan *entity* lainnya atau banyaknya *entity* yang bersesuaian dengan *entity* yang lain melalui *relationship*.

6. Derajat *Relationship*

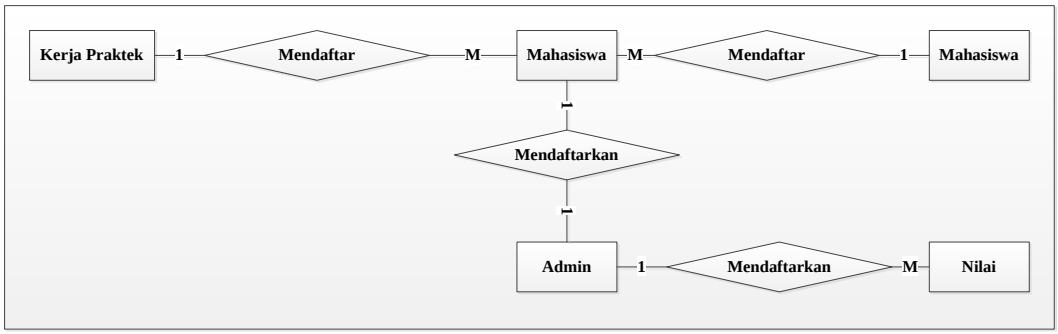
   Derajat *Relationship* menyatakan jumlah entity yang berpatisipasi di dalam suatu *relationship*.

7. *Participantion Contraint*

   Participantion Contraint, menjelaskan apakah keberadaan suatu *entity* tergantung pada hubungannya dengan *entity* yang lain.

8. Representasi dari *Entity Set*

   Entity set dipresentasikan dalam bentuk tabel dan nama yang *unique*. Setiap tabel terdiri dari sejumlah kolom dan diberi nama yang *unique*.



Gambar II.1. Contoh Entity Relationship Diagram

(Sumber : Antonio dan Safrida, 2012:13)

II.2.2. Kamus Data

Kamus data (data dictionary) mencakup definisi-definisi dari data yang disimpan di dalam basis data dan dikendalikan oleh sistem manajemen basis data (Raymond dan George, 2008:171). Struktur basis data yang dimuat dalam kamus data adalah kumpulan dari seluruh definisi *field*, definisi tabel, relasi tabel dan hal-hal lainnya.

Kamus data dibuat dan digunakan baik pada tahap analisis maupun pada tahap perancangan sistem. Pada tahap analisis kamus data digunakan sebagai alat komunikasi antara sistem analis dengan *user* tentang data yang mengalir pada sistem tersebut serta informasi yang dibutuhkan oleh pemakai sistem. Karena kamus data merupakan suatu *file* yang terpisah dalam suatu basis data, sehingga kamus data menyimpan informasi seperti :

1. Nama setiap *item*/jenis atau kolom data.
2. Struktur data untuk setiap *item*.
3. Program yang menggunakan tiap *item*.
4. Ukuran *item* pada setiap tipe data.

Kamus data berguna khusus bagi perlindungan timbulnya kelebihan data. Tanpa kamus data, pemakai data lain bagian mungkin menyimpan versi identik dari *item* data yang sama pada beberapa lokasi, dimana masing-masing *item* data mempunyai nama yang berbeda.

II.2.3. Normalisasi

Normalisasi merupakan proses pengelompokan elemen data menjadi tabel-tabel yang menunjukkan entitas dan relasinya. Teori normalisasi secara umum merupakan satu *set* peraturan yang membenarkan perkara pangkalan data menganai perkumpulan data yang tidak memuaskan dan menentukan hubungan yang boleh ditukar menjadi bentuk yang lebih baik. Untuk menggunakan normalisasi yang baik, pangkalan data mestilah mengetahui maksud data-data.

Menurut Maanari el at. (2013:4), proses normalisasi adalah proses pengelompokan elemen data menjadi tabel-tabel yang menunjukkan entitas dan relasinya.Terdapat enam bentuk normal. Namun demikian, dalam pemaparan berikut hanya akan dijelaskan bentuk normal pertama, kedua dan ketiga.

1. Bentuk Tidak Normal (*Unnormalized Form*)

   Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan untuk mengikuti suatu format tertentu. Dapat saja data tidak lengkap atau terduplikasi. Data tersebut dikumpulkan apa adanya.

2. Bentuk Normal Pertama (1NF)

   Suatu relasi 1NF jika dan hanya jika sifat dan setiap relasi atributnya bersifat atomik. Atomik bermaksud tidak berkepunyaan untuk berada dalam keadaan satu bagian. Ciri-ciri bentuk normal pertama, yaitu :

a. Setiap data dibentuk dalam *flat file*.

b. Tidak ada *set* atribut yang berulang atau bernilai ganda.

c. Tiap *field* hanya satu pengertian.

3. Bentuk Normal Kedua (2NF)

   Bentuk normal kedua mempunyai syarat yaitu bentuk data telah memenuhi kriteria bentuk normal pertama. Atribut bukan kunci haruslah bergantung secara fungsi pada *primary key*. Peraturan ini menentukan kebergantungan sepenuhnya. Beberapa sumber teks menjelaskan sebagai kebergantungan secara fungsi dan transitif.

4. Bentuk Normal Ketiga (3NF)

   Satu hubungan dikatakan dalam bentuk 3NF jika dan hanya jika ia dalam bentuk 2NF dan setiap atribut tanpa kekunci pula bergantungan secara tidak transitif dengan kunci primer.

II.3. *Unfied Modeling Languange* (UML)

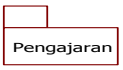
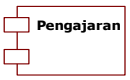
Unified Modeling Language (UML) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek (Adi Nugroho, 2010:6). Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

UML juga merupakan salah satu alat yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML

menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

Ada beberapa pengklasifikasi (*Classifer*) dan notasi dalam yang ditunjukkan pada Tabel II.1.dibawah ini.

Tabel II.1. Klaisfikasi dan Notasi UML

Pengklasifikasi	Kegunaan	Notasi
Actor	Menggambarkan semua objek diluar sistem yang berinteraksi dengan sistem yang dikembangkan.	
Use Case	Menggambarkan fungsionalistis yang dimiliki sistem.	
Kelas (*Class*)	Menggambarkan konsep dasar pemodelan sistem.	
Subsistem (*Subsystem*)	Menggambarkan paket spesifikasi serta implementasi.	
Komponen (*Componen*t)	Menggambarkan bagian-bagian fisik sistem/perangkat lunak yang dikembangkan.	
Antarmuka (*Interface*)	Menggambarkan antarmuka pengiriman pesan (*message*) antar pengklasifikasi.	
Simpul (*Node*)	Menggambarkan sumber daya komputasional yang digunakan oleh sistem.	

(Sumber : Adi Nugroho, 2010:16)

Relasi-relasi antar pengkalsifikasi yang dikenali UML adalah asosiasi, generalisasi, aliran, dan berbagai jenis kebergantungan termasuk didalamnya realisasi dan penggunaan.Untuk lebih jelasnya dapat dilihat pada Tabel II.2.berikut ini :

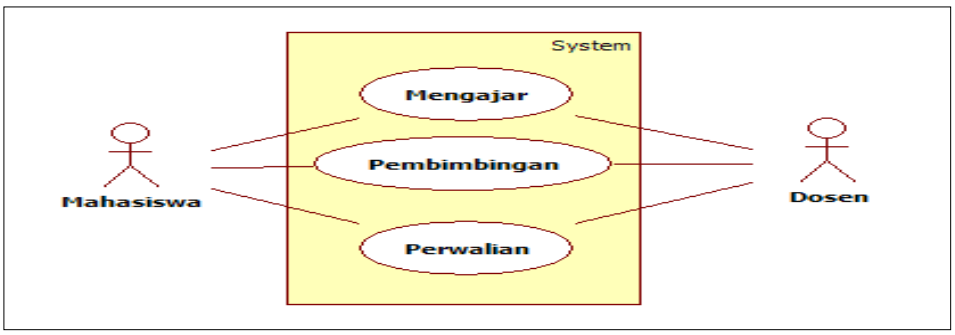
Tabel II.2. Relasi-relasi dalam UML

Relasi	Fungsi	Notasi
Asosiasi (*Association*)	Mendeskripsikan hubugan antar instance suatu kelas.	———
Kerbergantungan (*Dependency*)	Relasi antar dua elemen model.	- - - - - ->
Aliran (*Flow*)	Relasi antar dua versi suatu model.	- - - - - ->
Generalisasi (*Generalization*)	Relasi antar pengkalisifikasi yang memiliki deskripsi yang bersifat lebih umum dengan berbagai pengklasifikasi yang lebih spesifik, digunakan dalam struktur pewarisan.	——→
Realisasi (*Realization*)	Relasi antara spesifikasi dan implementasinya.	- - - - - -→
Penggunaan (*Usage*)	Situasi di mana salah satu elemen membutuhkan elemen yang lainnya agar dapat berfungsi dengan baik.	- - - - - ->

(Sumber : Adi Nugroho, 2010:23)

II.3.1. *Use Case Diagram*

Use Case Diagram adalah fungsionalitas atau persyaratan -persyaratan sistem yang harus dipenuhi oleh sistem yang akan dikembangkan tersebut menurut pandangan pemakai sistem (Sholiq, 2010:21). *Use case* diagram juga dapat diartikan 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*.

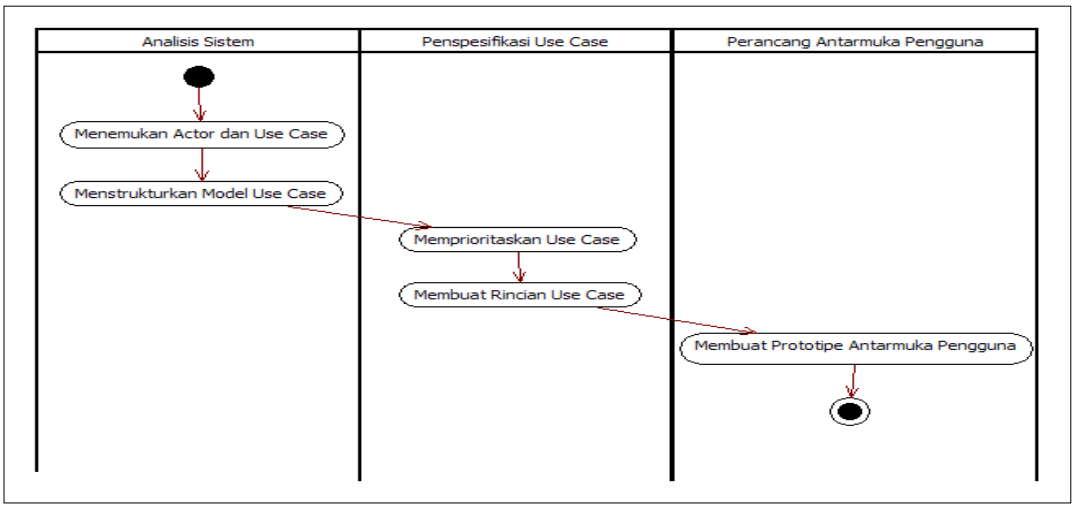


Gambar II.2. Contoh *Use Case Diagram*

(Sumber : Adi Nugroho, 2010:34)

II.3.2. *Activity Diagram*

Acitivity diagram memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses (Sulistyorini, 2009:27). Diagram ini sangat mirip dengan sebuah *flowchart*karena dapat dimodelkan sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas ke dalam keadaan sesaat (*state*). Seringkali bermanfaat bila dibuat sebuah *activity* terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan. *Activity diagram* juga sangat berguna ketika ingin menggambarkan perilaku paralel atau menjelaskan bagaimana perilaku dalam berbagai *use case* berinteraksi.



Gambar II.3. Contoh *Activity Diagram*

(Sumber : Adi Nugroho, 2010:141)

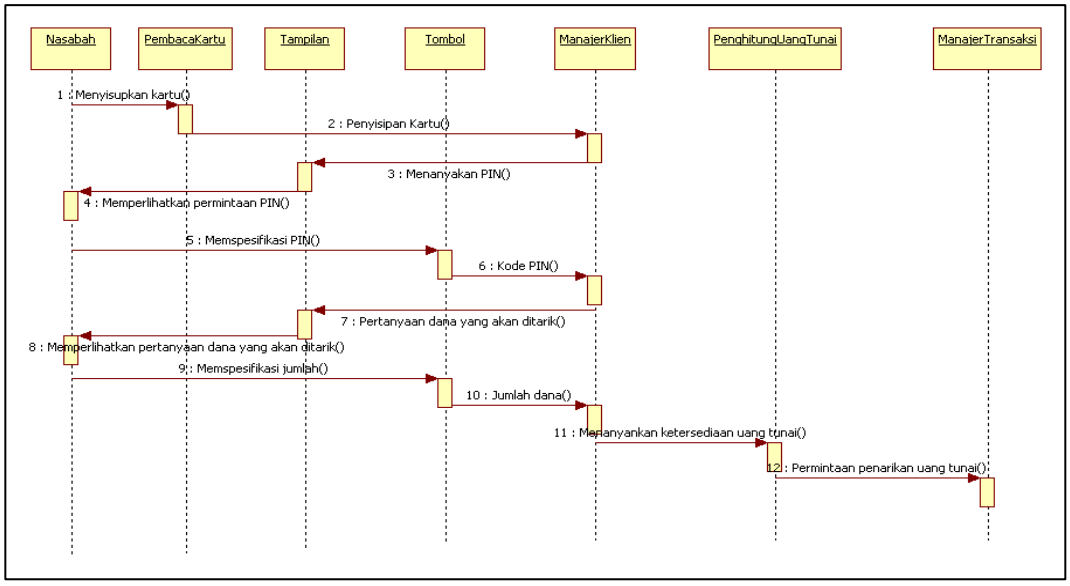
Dapat digunakan *statechart* diagram untuk memodelkan perilaku dinamis satu kelas atau objek. *Statechart* diagram memperlihatkan urutan keadaan sesaat (*state*) yang dilalui sebuah objek, kejadian yang menyebabkan sebuah transisi dari satu state atau aktivitas ke *state* atau aktivitas lainnya, dan aksi yang

menyebabkan perubahan satu state lainnya, dan aksi yang menyebabkan perubahan satu state atau aktivitas. Diagram aktivitas paling cocok digunakan untuk memodelkan aktivitas dalam suatu proses.

II.3.3. *Sequence Diagram*

Sequence Diagram memeperlihatkan interaksi sebagai diagram dua mantra (dimensi). Mantra vertical adalah sumbu waktu; waktu bertambah dari atas ke bawah sedangkan mantra horizontal memperlihatkan pengklasifikasi yang mempresentasikan objek-objek mandiri yang terlibat dalam kolaborasi.

Diagramsequence merupakan diagram interaksi yang menekankan pada pengiriman pesan (*message*) dalam suatu waktu tertentu (Sulistyorini, 2009:24). *Sequence Diagram* digunakan untuk menggambarkan perilaku pada sebuah skenario.Diagram ini menunjukkan sejumlah contoh objek dan *message* (pesan) yang diletakkan diantara objek-objek ini dalam *use case*.



Gambar II. 4. Contoh *Sequence Diagram*

(Sumber : Adi Nugroho, 2010:109)

Ada beberapa komponen yang terdapat pada *sequence diagram*, yaitu :

1. Objek*/Participant*

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

2. *Message*

   Sebuah *message* bergerak dari satu *participan*t ke *participant* yang lain dan dari satu *lifeline* ke *lifeline*. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri. Jika sebuah *participant* mengirimkan sebuah *message synchronous*, maka jawaban atas *messege* tersebut akan ditunggu sebelum diproses dengan urusannya. Namun jika *message synchronous*yang dikirimkan, maka jawaban atas *message* tersebut tidak perlu ditunggu.

3. *Time*

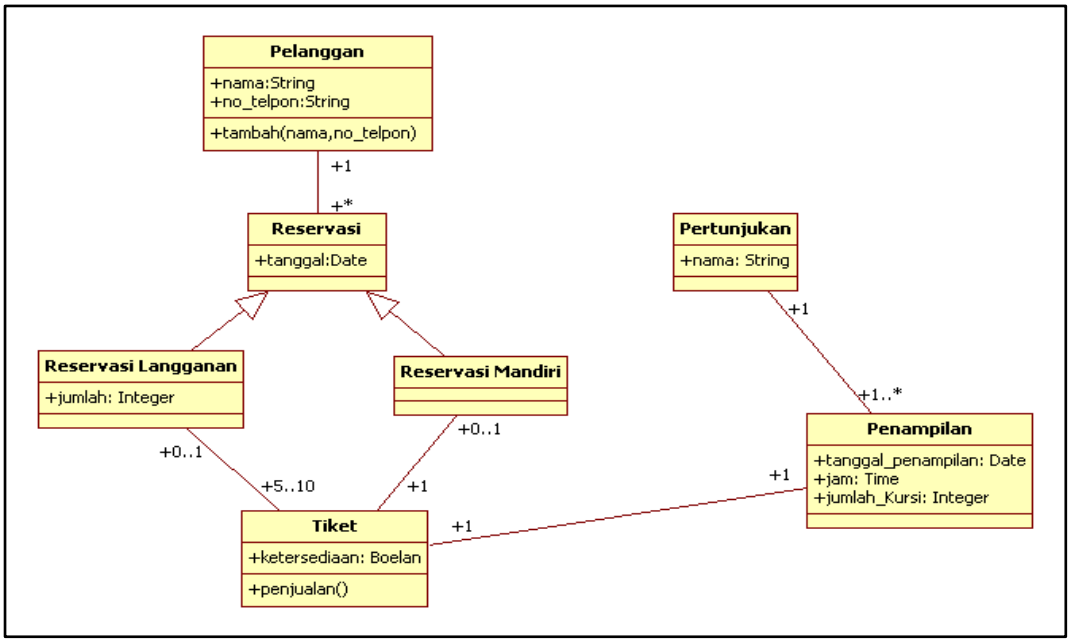
   Time adalah diagram yang mewakili waktu pada arah *vertical*. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibanding *message* yang lebih dekat ke bawah.

II.3.4. *Class Diagram*

Class Diagram menunjukkan interaksi antar kelas-kelas dalam sistem. Sebuah kelas mengandung informasi dan tingkah laku (*behavior*) yang berkaitan dengan informasi tersebut.*Class diagram* sesungguhnya merupakan deskripsi dari konsep yang datang dari arah aplikasi atau solusi aplikasi (Adi Nugroho, 2010

:13). Oleh karena itu pengertian kelas sangat penting sebelum merancang diagram kelas. Kelas diagram sebagai satu *set* objek yang memiliki atribut dan prilaku yang sama. *Class diagram* membantu dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak.

*Class diagram*memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain (dalam *logical view*) dari suatu sistem. Selama proses analisi, *class diagram* memperlihatkan aturan -aturan dan tanggung jawab entitas yang menentukan perilaku sistem.



Gambar II.5. *Contoh Class Diagram*

(Sumber : Adi Nugroho, 2010:12)

II.4. *MySQL Server*

MySQL adalah database server relasional yang gratis di bawah lesensi GNU (*General Public License*). Dengan sifatnya *Open Source*, memungkinkan juga user untuk melakukan modifikasi pada *source code*-nya untuk memenuhi

kebutuhan spesifik. *MySQL* merupakan database server *multi-user* dan *multi-threaded* yang tangguh (*robust*). Dengan memiliki banyak fitur *MySQL* bisa bersaing dengan *database* komersil lainnya. *MySQL* dikembangkan, disebarluaskan, dan didukung oleh *MySQL* AB yang merupakan perusahaan komersial yang didirikan oleh para pengembang *MySQL* (Wahana, 2010:26).

MySQL merupakan sebuah perangkat lunak sistem manajemen basis data*SQL*(*database management system*) atau *DBMS* yang memiliki konsep *multithread* dan *multi-user*serta menggunakan perintah *SQL* (*Structured Query Language*). *MySQL* merupakan sebuah program *database server* yang mampu menerima dan mengirimkan datanya dengan sangat cepat kedalam sebuah sistem aplikasi (*Client*).

II.4.1. Tipe Data *MySQL Server*

MySQL Server mendukung banyak tipe data yang dapat disimpan pada sebuah kolom. Terdapat tiga kategori tipe data yang didukung oleh *MySQL Server*, yaitu :

1. Tipe Data Numerik

   Data numeric adalah salah satu bentuk data berupa angka, baik berupa bilangan bulat maupun bilangan *real*. Bilangan bulat dapat berupa tipe data *integer/int*, *tinyiny*, *smallint*, dan lainnya. Sebaliknya bilangan *real* dapat menyimpan data berupa angka pecahan. Jenis tipe data bilangan bulat dapat dilihat pada Tabel II.3. dan jenis tipe data bilangan *real* dapat dilihat pada Tabel II.4. berikut ini.

Tabel II.3. Tipe Data Numerik Bilangan Bulat

Tipe Data	*Byte*	Nilai Minimal	Nilai Maksimal
Tinyint	1	-128	127
		0	255
Smallint	2	-32768	32767
		0	65535
Mediumint	3	-8388608	8388607
		0	16777215
Int/Integer	4	-2147483648	21455483648
		0	4294967295
Bigint	8	-9223372036854775808	9223372036854775807
		0	184467440373709551615

(Sumber : Wahana, 2010:31)

Tabel II.4. Tipe Data Numerik Bilangan *Real*

Tipe Data	*Byte*	Keterangan
Float (p)	4 jika0<=p<=24	P mempresentasikan presisi bit.
Float	4	Angka *floatingpoint* kecil
Double	8	Ukuran normal angka *floating point*
Decimal(M,D), *Numeric*(M,D)	Variasi	M adalah jumlah angka digit *decimal* dan D adalah angka dibelakang *decimal*
Bit (M)	(M+7)/8	M adalah banyaknya bit setiap nilai.

(Sumber : Wahana, 2010:31)

2. Tipe Data String

Tipe data string dapat menyimpan semua data, baik berupa karakter, angka, waktu maupun tanggal. Data dapat pula merupakan kombinasi karakter dan angka.

Tabel II.5. Tipe Data String

Tipe Data	*Byte*	Keterangan
Varchar	255	Tipe *varchar* menyimpan data sebanyak karakter yang diinputkan.
Char	255	Tipe *char* sama dengan *varchar*, hanya saja tempat penyimpanan selalu tetap.
Binary	255	*Binary* mirip dengan *char* hanya yang disimpan adalah nilai biner dari data yang disimpan.
Varbinary	255	*Varbinary* sama dengan *binary*, tetapi keduanya berbeda sebagaimana perbedaaan *char* dengan *varchar*.

Enum	N	Tipe data ini disebut juga dengan tipe data validasi. Pada tipe ini data inputan telah dideklarasikan terlebih dahulu.
Set	N	*Set* memeiliki fungsi yang sama dengan *enum*.

(Sumber : Wahana, 2010:33)

3. Tipe Data Penanggalan dan Waktu

   Dalam menangani data tanggal dan waktu (jam), *MySQL* memiliki tipe data tersendiri.

Tabel II.6. Tipe Data Tanggal dan Waktu

Tipe Data	*Byte*	Keterangan
Datetime	8	Bentuk ini merupakan tipe data yang menyimpan dua tipe, yaitu tanggal dan jam.
Date	3	Tipe ini hanya menyimpan data tanggal dengan format "0000-00-00".
Timestamp	4	Tipe ini ditulis berjajar tanpa ada pembatas. Tipe ini dapat menyimpan tanggal dan jam.
Time	3	Tipe ini hanya dapat menyimpan data jam.
Year	1	Tipe ini hanya menyimpan data tahun saja.

(Sumber : Wahana, 2010:33-34)

II.4.2. *Internal* dan *Protabilitas MySQL Server*

Salah satu fitur yang terdapat pada *MySQL* adalah fitur *internal* dan *portabilitas*. Menurut Wahana Komputer (2010:27-28), ada beberapa fitur *internal* dan portabilitas sebagai salah satu fitur utama pada *MySQL*, antara lain :

1. *MySQL Server* ditulis dengan bahasa C dan C++.
2. *MySQL Server* diuji secara luas menggunakan *compiler* yang berbeda.
3. *MySQL Server* mampu bekerja pada banyak *platform* berbeda.
4. *MySQL Server* menggunakan GNU *Automake*, *Autoconf*, dan *Libtool* untuk *portabilitas.*

5. Tersedia API untuk C, C++, Eifel, Java, Perl, PHP, Python, Ruby, dan Tcl untuk mengakses *MySQL Server*.
6. *MySQL Server* secara penuh mendukung *multi-threaded* menggunakan *thread* kernel sehingga mudah menggunakan banyak CPU, jika tersedia.
7. *MySQL Server* menyediakan mesin penyimpanan transaksi dan non-transaksi.
8. *Thread MySQL Server* sangat cepat berdasarkan sistem alokasi memori. Penggabungan tabel *MySQL Server* dapat dilakukan sangat cepat menggunakan optimasi *one sweep multi-join*.

II.5. PHP

Hypertext preprocessor atau PHP adalah adalah salah satu jenis bahasa pemograman *web* yang open source, sehingga dapat digunakan oleh siapa saja secara cuma-cuma (Wiswakarma, 2009:12). PHP juga merupakan bahasa *server-sidescripting* yang bisa menyatu dengan tag-tag HTML. *Server-sidescripting* adalah sintaks dan perintah-perintah yang dijalankan pada server dan disertakan pada dokumen HTML. PHP berfungsi sebagai bahasa pemrograman yang menjalankan suatu perintah tertentu. PHP mampu mengelolah data pada berbagai *platform database*, namun yang paling ideal dan banyak digunakan adalah menggunakan database MySQL. PHP+MySQL menjadi standar bagi pembuatan web dinamis saat ini, hal ini dikarenakan keduanya *opensource*, sehingga bisa digunakan siapa saja dengan bebas.

Dalam penulisan *script* PHP, terdapat *tag* pembukan dan *tag* penutup. *Tag* ini merupakan sebagai identifer PHP itu sendiri. Sehingga ketika PHP digabungkan dengan *scripting* yang lain, seperti HTML dan *Javascript* maka

*tag*pembukan dan *tag* penutup inilah yang akan memisahkan PHP dengan bahasa pemograman yang lainnya. Selain *tag* tersebut, PHP juga mempunyai variabel. Variabel dalam PHP tidak seperti variabel dalam bahasa pemograman lain. Variabel PHP tidak membutuhkan deklarasi sebelum digunakan (Ramadhan dan Nugroho, 2009:314). Dengan begitu nilai dapat dimasukkan kapanpun untuk digunakan. Penulisan variabel PHP diawali dengan simbol ”**$**”.

Pemograman PHP tidak terlepas dari *control flow*. Hal ini dikarenakan PHP mampu bermain dengan logika. Pada Tabel II.7. akan menjelaskan jenis-jenis *control flow* pada PHP.

Tabel II.7. *Control Flow* Pada PHP

Statement Type	***Keywords***
Looping	*While, do-while, for.*
Decision making	*If-else, switch-case*
Exception handling	*Try-catch, finally, throw*
Branching	*Break, continue, label, return.*

(Sumber : Ramadhan dan Nugroho, 2009:314)