BAB II

TINJAUAN PUSTAKA

II.1. Konsep Dasar Sistem Informasi

II.1.1. Sistem

Istilah sistem bukanlah hal yang asing bagi kebanyakan orang. Dari segi Etimologi kata sistem sebenarnya berasal dari bahasa Yunani yaitu ”*Systema*” di dalam bahasa Inggris juga dikenal dengan nama “System” yang dapat diartikan yaitu sehimpunan bagian atau komponen lain yang saling berhubungan secara teratur dan merupakan satu keseluruhan yang tidak terpisahkan. Menurut filusuf Stoa, bahwa sistem adalah gabungan dari keseluruhan langit dan bumi yang bekerja bersama-sama, sehingga dapat kita lihat bahwa sistem terdiri dari unsur-unsur yang bekerja sama membentuk satu keseluruhan dan apabila salah satu dari keseluruhan tersebut hilang atau tidak berfungsi, maka gabungan keseluruhan tidak lagi dapat disebut suatu sistem. Berikut ini adalah definisi kata sistem menurut beberapa ahli yaitu:

1. Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu (Jogiyanto HM ; 2005 : 1).
2. Suatu sistem dapat dirumuskan sebagai setiap kumpulan komponen atau subsistem yang dirancang untuk mencapai suatu tujuan. Dengan pendekatan sistem kita berhubungan dengan komponen perseorangan dan

kita lebih menekankan perannya dalam sistem daripada perannya sebagai suatu keseluruhan individu. Dengan pendekatan sistem untuk menggambarkan kenyataan, dapat di berikan suatu keuntungan yang besar kepada pemakai. Keberhasilan komponen-komponen yang di pertimbangkan secara bersama sebagai suatu kemungkinan lebih besar daripada jumlah keberhasilan setiap komponen yang dipertimbangkan secara terpisah (Tata Sutabri ; 2005 : 8-9).

Dari pendapat kedua ahli diatas dapat diambil kesimpulan bahwa Sistem adalah sekumpulan komponen yang saling bekerjasama untuk mencapai tujuan atau sasaran tertentu. Dimana masing-masing komponen memiliki fungsi yang berbeda antara satu dengan yang lainnya, namun tetap dapat bekerjasama. Fungsi sistem yang utama adalah menerima masukan, mengolah atau memproses masukan dan menghasilkan keluaran.

II.1.2. Informasi

Berikut pengertian informasi menurut beberapa ahli:

1. Informasi adalah data yang sudah dibentuk kedalam format yang memiliki arti bagi manusia. (Kenneth C. Laudon dan Jane P. Laudon ; 2005 : 10)
2. Menurut Tata Sutabri (2005:23), informasi adalah data yang telah diklasifikasi atau diolah diinterprestasi untuk digunakan dalam proses pengambilan keputusan.

   Dari beberapa pendapat diatas dapat disimpulkan bahwa sistem adalah data yang sudah diolah atau diproses sehingga dapat dimengerti, mempunyai makna serta dapat digunakan untuk mengambil keputusan.

II.1.3. Sistem Informasi

Berikut pengertian sistem informasi menurut beberapa ahli :

1. Sistem informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan data yang mendukung fungsi operasional yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar tertentu dengan laporan-laporan yang diperlukan. (Tata Sutabri ; 2005 : 42)
2. Sistem informasi adalah suatu sistem dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian mendukung operasi bersifat manajerial dan kegiatan dari strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan-laporan yang diperlukan. (Jogiyanto Hartono ; 2006 : 11)
3. Sistem informasi dapat didefenisikan sebagai kumpulan elemen-elemen yang saling berhubungan satu sama lain yang membentuk satu kesatuan untuk mengintegrasikan data memproses, menyimpan serta mendistribusikan informasi. Sistem informasi merupakan kesatuan elemen-elemen yang saling berinteraksi secara sistematis dan teratur unuk menciptakan dan membentuk aliran informasi yang akan mendukung pembuatan keputusan dan melakukan kontrol terhadap jalannya perusahaan. (Budi Sutedjo Dharma Oetomo ; 2006 : 11)

Dari beberapa pendapat diatas dapat disimpulkan bahwa sistem informasi adalah sekumpulan komponen yang bekerjasama secara sistematis dan terpadu

dalam pengolahan data untuk memperoleh informasi dengan maksud dan tujuan yang terpenting sebagai bahan masukan dalam mengambil keputusan.

II.2. Sistem Informasi Geografis (SIG)

Sistem Informasi Geografis (SIG) adalah sebuah sistem atau teknologi berbasis komputer yang dibangun dengan tujuan untuk mengumpulkan, menyimpan, mengolah, dan menganalisa, serta menyajikan data-data dan informasi dari suatu objek atau fenomena yang berkaitan dengan letak atau keberadaannya di permukaan bumi. Pada dasarnya SIG dapat dirinci menjadi beberapa sub sistem yang saling berkaitan mencakup input data, manajemen data, pemrosesan atau analisis data, pelaporan (*output),* dan hasil analisa (Andree Ekadinata ; 2008 : 13).

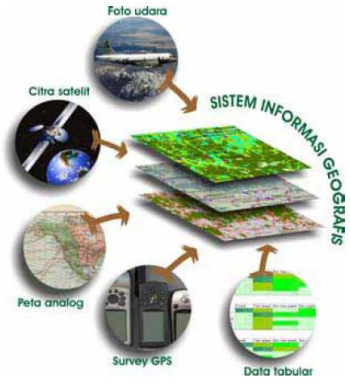
Komponen-komponen yang membangun SIG adalah perangkat lunak, perangkat keras, data, pengguna, dan aplikasi. SIG dalam pengelolaan sumber daya alam di lingkungan pemerintah lokal, sebagai contoh, memerlukan sistem yang mendukung tersedianya kelima komponen tersebut, sebagaimana di ilustrasikan oleh Gambar II.1.

Gambar II.1 Komponen Sistem Informasi Geografis
Sumber : Andree Ekadinata (2008:13)

II.2.1. Jenis dan Sumber Data Geografis

Data geografis pada dasarnya tersusun oleh dua komponen penting yaitu data spasial dan data atribut. Data spasial merepresentasikan posisi atau lokasi geografis dari suatu objek di permukaan bumi, sedangkan data atribut memberikan deskripsi atau penjelasan suatu objek. Data atribur dapat berupa informasi numerik, foto, narasi, dan lain sebagainya, yang diperoleh dari data statistik, pengukuran lapangan dan sensus, dan lain-lain.

Data spasial dapat diperoleh dari berbagai sumber dalam berbagai format. Sumber data spasial antara lain mencakup data grafis peta analog, foto udara, citra satelit, survey lapangan, pengukuran theodolit, pengukuran dengan menggunakan *Global Positioning Systems (GPS)*. Adapun format data spasial, secara umum dapat dikategorikan dalam format digital dan format analog. Sebagaimana di ilustrasikan oleh Gambar II.2.



Gambar II.2 Sumber data dalam SIG

II.2.2. Pengelolaan Basis Data Dalam SIG

Sebuah sistem informasi geografis yang sudah berjalan dan dikelola dengan baik, pada umumnya merupakan kumpulan data yang cukup banyak

jumlahnya dan beragam jenis an sumbernya. Data-data ini sangat bervariasi ditinjau dari tema, sumber, skala dan resolusi, tingkat pengerjaan dan lain sebagainya. Dalam SIG, seluruh data tersebut terintegrasi dalam sebuah sistem yang disebut basis data spasial (*spatial database).* Pengelolaan basis data spasial adalah inti dari sistem informasi geografis itu sendiri. Semakin baik pengelolaan basis data spasial maka akan semakin mudah SIG diaplikasikan. Lebih jauh lagi, kukalitas keluaran dari analisa SIG akan sangat tergantung pada kualitas pengelolaan data spasial.

Satu komponen penting dalam pengelolaan basis data spasial adalah metadata. Metadata adalah informasi sebuah data. Metadata memuat informasi-informasi penting yang sangat membantu untuk menjawab pertanyaaan-pertanyaan sebagai berikut:

1. Kapan sebuah data dihasilkan?
2. Siapa yang membuat?
3. Apa tujuan data tersebut dibuat?
4. Berapa tingkat akurasinya?

Metadata akan sangat penting nilainya terutama jika sebuah basis data spasial merupakan kumpulan data yang sangat banyak. Dengan adanya metadata, berbagai kesalahan seperti duplikasi data dapat dihindarkan. Berikut adalah contoh metadata sederhana Geonetwork yang bisa didapatkan secara gratis pada situs http://geonetwork-opensource.org/

II.2.3. Komponen Sistem Informasi Geografis

Menurut Eddy Prahasta (2009:120) Sistem informasi geografis terdiri dari beberapa komponen sebagai berikut:

1. Perangkat keras

   Perangkat keras yang sering digunakan antara adalah komputer (PC), mouse, monitor (plus VGA-card grafik) yang beresolusi tinggi, *digitizer, printer, plotter, receiver* GPS dan *scanner*.

2. Perangkat lunak

   (Arc View, Idrisi, ARC/INFO, ILWIS, MapInfo dan lain-lain)

3. Data dan informasi geografis

   SIG dapat mengumpulkan dan menyimpan data atau informasi yang diperlukan baik secara tidak langsung (dengan cara meng-*import*-nya dari format-format perangkat lunak SIG yang lain) maupun secara langsung dengan cara melakukan dijitasi data spasialnya dari peta analog dan kemudian memasukan data atributnya dari tabel-tabel atau laporan dengan menggunakan *keyboard*.

II.2.4. Pengguna SIG

Pengguna (*brainware*) adalah komponen terpenting dalam SIG. Pengguna menentukan jenis analisa yang dilakukan. Menjalankan analisa tersebut, mengintrepetasi hasil analisa, dan menyajikan hasil analisa dalam bentuk yang komunikatif. Pengguna dapat dikategorikan dalam dua kelompok besar: pelaku analisa (*analyst/operator*) dan pengguna informasi (*user*).

Analyst/operator SIG sebaiknya memiliki pengetahuan yang cukup dalam bidang ilmu, wlaupun bidang ilmu yang berkaitan dengan SIG antara lain:

geografi, matematika, statistik, dan lain-lain. Hal ini dikarenakan banyak dimensi dalam aplikasi SIG yang menyangkut berbagai bidang keilmuan. *Analyst/operator* juga sebaiknya memiliki pengetahuan dan keterampilan yang cukup dalam pengoperasian komputer.

Pada sisi *user*, pandangan bahwa menggunakan SIG sulit dan membutuhkan keahlian tinggi, adalah tidak benar. Pada kenyataannya beberapa fungsi SIG dapat digunakan secara mudah, bahkan bagi pengguna yang awam. Hal ini dibuktikan pada penggunaan Google Earth yang semakin hari semakin banyak jumlahnya. Pada akhirnya tujuan utama SIG adalah menyediakan informasi bagi semua orang, bukan hanya ahli geografi saja. Aplikasi SIG yang baik mampu menyediakan informasi yang akurat, tepat guna, dan informatif.

II.2.5. MapInfo

MapInfo merupakan salah satu perangkat lunak pemetaan (SIG) *desktop* yang dikembangkan dan kemudian dipasarkan untuk memenuhi (sebagian besar) kebutuhan-kebutuhan di lingkungan bisnis. Perangkat SIG yang versi 7.5 profesionalnya di-*release* tahun 2003 ini memungkinkan para penggunanya untuk memvisualisasikan dan menganalisa data-data yang menjadi masukannya secara geografis lebih cepat dan menyediakan informasi yang diperlukan di dalam proses pengambilan keputusan

MapInfo profesional versi 7.5 memilki beberapa fungsionalitas yang telah dimiliki oleh versi-versi sebelumnya dan beberapa *features* baru seperti

1. Dilengkapi dengan *interface* yang memungkinkan para pengguna untuk menggunakan peta-peta dijital yang tersebar dalam jaringan komputer lokal bahkan internet.
2. Menggeser dan merotasikan objek-objek grafis
3. Menyimpan objek-objek ke dalam sebuah table.
4. Menentukan batas (table bounds) untuk table RDBMS ketika pengguna membuat sebuah table pada saat live acces
5. Merubah skala di dalam *window layout* dengan cara merubah ukuran frame atau merubah nilai perbesaran (*zoom*).
6. Memiliki fasilitas *scale pattern* yang dapat digunakan untuk menghasilkan *print-out* peta dijital berwarna kualitasnya sedekat mungkin dengan tampilan softcopy-nya di layar monitor (Eddy Prahasta ; 2005:3).

II.2.6. Data Spasial MapInfo

Data spasial utama yang di gunakan oleh MaInfo yang menggunakan model vektor diimplementasikan sebagai sebuah *table*. Data spasial yang diimplementasikan sebagai *table* ini terdiri dari beberapa komponen *file* seperti berikut :

1. ***.DAT:** *file* yang digunakan untuk menyimpan data atribut atau tabel milik sebuah *table.*
2. ***.TAB**: *file* utama yang berisi informasi struktur tabel, urutan, nama field, dan tipe *field* yang terdapat di dalamnya.

3. *.**MAP**: *file* yang berisi informasi geografis yang mendeskripsikan objek-objek peta.
4. *.**IND**: *file* yang berisi indeks data yang terdapat di dalam tabel atribut terkait (*DAT).
5. *.**ID**: *file* yang berisi indeks data yang terdapat di dalam objek geografis terkait (*.MAP).

II.3. UML (*Unified Modelling Language*)

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

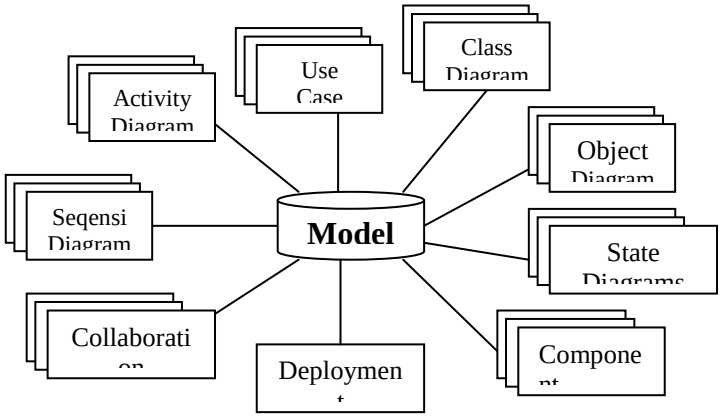
UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, Ob*ject Modeling Technique* (OMT) dan *Object Oriented Software Engineering* (OOSE). Metode Booch dari Grady Booch sangat terkenal dengan nama metode *Design Object Oriented*. Metode ini menjadikan proses analisis dan desain kedalam empat tahapan interatif, yaitu: identifikasi kelas-kelas dan objek-objek, identifikasi *semantic* dari hubungan objek dan kelas tersebut, perincian interface dan implementasi. Keunggulan metode Booch adalah pada detil dan kayanya dengan notasi dan elemen. Pemodelan OMT dikembangkan oleh

Rumbaugh didasarkan pada analisis struktur dan pemodelan *entity relationship*. Tahapan utama dalam metodologi ini adalah analisis, desain sistem, desain objek, dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua objek OO. Metode OOSE dari Jacobson lebih memberikan penekanan pada *use case*. OOSE memiliki tiga tahapan yaitu membuat model *requirement*, analisis, desain, implementasi dan model pengujian. Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak.

Dengan UML, metode Booch, OMT dan OOSE digabungkan dengan membuang elemen-elemen dari metode lain yang lebih efektif dan elemen-elemen baru yang belum ada pada metode terdahulu sehingga UML lebih ekspresif dan seragam dari metode lainnya.(Munawar ; 2005 : 17-18)

II.3.1. Klasifikasi Diagram UML

Diagram berbentuk grafik yang menunjukkan simbol elemen model yang disusun untuk mengilustrasikan bagian atau aspek tertentu dari sebuah sistem. Adapun klasifikasi diagram UML dapat dilihat pada gambar II.3.



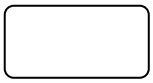
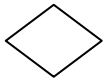
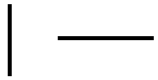
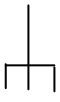
Gambar II.3. Klasifikasi Diagram UML
Sumber (Munawar ; 2005 :19)

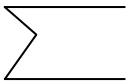
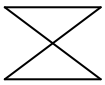
Keterangan:

1. *Activity Diagram,* menggambarkan berbagai alir aktivitas dalam sebuah sistem yang dirancang, bagaimana masing-masing alir berawal, *decision* yang terjadi dan bagaimana alir berakhir. *Activity diagram* juga dapat menggambarkan proses pararel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Berikut adalah simbol-simbol yang digunakan pada *Activity Diagram*.

   Adapun Notasi Activity Diagram dapat dilihat pada Tabel II.1.

Tabel II.1. Notasi *Activity Diagram*

Simbol	Keterangan
	*Initial*, merupakan titik awal untuk memulai suatu aktivitas.
	*Final*, merupakan titik akhir untuk mengakhiri aktivitas.
	*Aktivity*, menandakan sebuah aktivitas.
	*Decission*/*Marge*, pilihan untuk mengambil keputusan
	*Folk* (*Join*), Digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Rake*, menunjukkan adanya suatu dekomposisi.

	*Flow Final*, untuk mengkhiri suatu aliran
	*Receive*, merupakan tanda penerimaan.
	*Time*, merupakan tanda waktu
	*Send Siqnal Action*, merupakan tanda pengiriman

Sumber (Munawar ; 2005 : 109)

2. *Use-Case Diagram*, menjelaskan manfaat dari aplikasi jika dilihat dari sudut pandang orang yang berada diluar sistem (*actor*). Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem berinteraksi dengan dunia luar. *Use-case diagram* dapat digunakan selama proses analisa untuk menangkap *requirements* atau permintaan terhadap sistem dan untuk memahami bagaimana sistem tersebut harus berkerja. Selama tahap desain, *use-case diagram* menetapkan perilaku dari aplikasi saat implementasi. Dalam sebuah model memungkinkan terdapat satu atu beberapa *use-case diagram*.

Adapun Notasi Use-Case Diagram dapat dilihat pada Tabel II.2.

Tabel II.2. Notasi *Use-Case Diagram*

Notasi	Keterangan
	*Actor*, menggambarkan segala pengguna software aplikasi (*user*). *Actor* memberikan suatu gambaran jelas tentang apa yang harus dikerjakan software aplikasi.

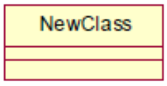
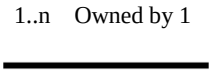
	*Use case*, menjelaskan urutan kegiatan yang dilakukan *actor* dan sistem untuk mencapai suatu tujuan tertentu. Walaupun menjelaskan kegiatan, namun *use case* hanya menjelaskan apa yang dilakukan oleh *actor* dan sistem bukan bagaimana *actor* dan sistem melakukan kegiatan tersebut.
	Asosiation digunakan untuk menghubungkan *actor* dengan *use case*.

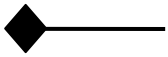
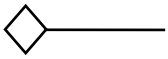
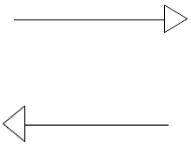
Sumber (Munawar ; 2005 : 111)

3. *Class Diagram,* dapat membantu dalam memvisualisasikan struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class diagram* banyak memperhatikan hubungan antar kelas dan penjelasan detail tiap kelas dalam pemodelan desain (dalam *logical view*) dari suatu sistem. Selama proses analisa, *class diagram* memperhatikan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

   Adapun Notasi Class Diagram dapat dilihat pada Tabel II.3.

Tabel II.3. Notasi *Class Diagram*

Notasi	Keterangan
	*Class*, merupakan pembentuk utama dari sistem berorientasi obyek, karena class menunjukkan kumpulan obyek yang memiliki atribut dan operasi yang sama. *Class* digunakan untuk mengimplementasikan *interface*, mengabstraksikan elemen-elemen dari sistem yang sedang dibangun. *Class* bisa merepresentasikan baik perangkat lunak maupun perangkat keras, baik konsep maupun benda nyata. Notasi *class* berbentuk persegi panjang berisi tiga bagian, yaitu: persegi panjang paling atas untuk nama *class*, persegi panjang paling bawah untuk operasi, dan persegi panjang ditengah untuk *atribut*.
	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 *class*, dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship* (*One-to-one*, *one-to-many*, *many-to-many*).

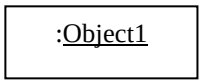
	*Composition*, Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.
	*Dependency*, merupakan relasi yang menunjukan bahwa perubahan pada salah satu elemen memberi pengaruh pada elemen lain. Elemen yang ada di bagian tanda panah adalah elemen yang tergantung pada elemen yang ada dibagian tanpa tanda panah. Terdapat 2 *stereo type* dari *dependency*, yaitu *include* dan *extend*.
	*Aggregation Relationship*, adalah bentuk khusus assosiasi yang memodelkan hubungan keanggotaan antara 2 kelas, yakni satu kelas disusun oleh kelas lainnya.
	*Generalization*, menunjukkan hubungan antara elemen yang lebih umum ke elemen yang lebih spesifik. Dengan *generalization*, *class* yang lebih spesifik (*subclass*) akan menurunkan atribut dan operasi dari *class* yang lebih umum (*superclass*) atau *subclass* is *superclass*. Dengan menggunakan notasi *generalization* ini, konsep *inheritance* dari prinsip hirarki dapat dimodelkan.

Sumber (Munawar ; 2005 : 112)

4. *Object Diagram,* merupakan gambaran objek-objek secara ringkas dalam sebuah sistem pada suatu waktu. Objek diagram sering disebut sebagai instance diagram karena menunjukkan instance-instance dari *class*. Objek diagram bisa digunakan untuk menunjukkan contoh konfigurasi dari objek-objek. Hal ini sangat berguna untuk menunjukkan hubungan yang mungkin ada diantara objek-objek yang sangat kompleks.

   Adapun Notasi *Object Diagram* dapat dilihat pada Tabel II.4.

Tabel II.4. Notasi *Object Diagram*

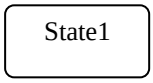
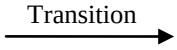
Notasi	Keterangan
	*Object,* Obyek-obyek diidentifikasikan dengan cara meletakkan nama instance-nya kemudian diikuti oleh tanda titik dua didepan nama class-nya. Nilai property/atribut dituliskan berpasangan seperti "nama_atribut=nilai". Sedangkan notasi sebuah obyek digambarkan segi empat yang terbagi atas 2 bagian.
	Association, *Object* diagram juga dapat mengandung asosiasi. Biasanya constraint, detil relationship, multiplisitas yang ada di class diagram tidak disertakan dalam object diagram sebagai upaya memfokuskan perhatian hanya terhadap obyek dan property/atributnya. Asosiasi antar 2 obyek biasanya dinotasikan dengan sebuah garis yang menghubungkan kedua obyek.

Sumber (Munawar ; 2005 : 115)

5. *Statechart Diagram*, digunakan 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 suatu state atau aktivitas kepada yang lainnya. *Statechart diagram* khusus digunakan untuk memodelkan tahap-tahap diskrit dari sebuah siklus hidup objek.

Adapun Notasi Statechat Diagram dapat dilihat pada Tabel II.5.

Tabel II.5. Notasi *Statechat Diagram*

Notasi	Keterangan
	State, menggambarkan kondisi sebuah entitas, dan digambarkan dengan segiempat yang pinggirnya tumpul dengan nama state didalamnya.
	Transition, menggambarkan sebuah perubahan kondisi objek yang disebabkan oleh sebuah event. Transition digambarkan dengan sebuah anak panah dengan nama event yang ditulis diatasnya, dibawahnya atau sepanjang anak panah tersebut.
	Initial State adalah sebuah kondisi awal sebuah object sebelum ada perubahan keadaan. Initial State digambarkan dengan sebuah lingkaran solid. Hanya satu Initial State yang diizinkan dalam sebuah diagram.

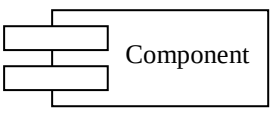
	Final State menggambarkan ketika objek berhenti memberi respon terhadap sebuah event. Final State digambarkan dengan lingkaran solid didalam sebuah lingkaran kosong.

Sumber (Munawar ; 2005 : 117)

6. *Component Diagram,* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi kode, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

Adapun Notasi *Component Diagram* dapat dilihat pada Tabel II.6.

Tabel II.6. Notasi *Component Diagram*

Notasi	Keterangan
	Sebuah komponen melambangkan sebuah entitas software dalam sebuah sistem. Sebuah komponen dinotasikan sebagai sebuah kotak segiempat dengan dua kotak kecil tambahan yang menempel disebelah kirinya.

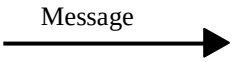
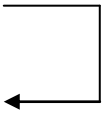
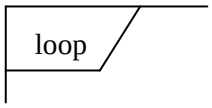
	Dependency digunakan untuk menotasikan relasi antara dua komponen. Notasinya adalah tanda panah putus-putus yang diarahkan kepada komponen tempat sebuah komponen itu bergantung.

Sumber (Munawar ; 2005 : 119)

7. *Deployment*/*physical Diagram,* menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, *server* atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi *server*, dan hal-hal lain yang bersifat fisikal.
8. *Collaboration Diagram,* menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki *prefiks* yang sama.
9. *Sequence Diagram,* menjelaskan interaksi objek yang disusun dalam suatu urutan waktu. Diagram inisecara khusus berasosiasi dengan *use-case*. *Sequence* diagram memperlihatkan tahap demi tahap apa yang arus terjadi untuk menghasilkan suatu didalam *use-case* diagram. Tipe diagram yang digunakan sebaiknya digunakan diawal tahap desain atau analisis karena kesederhanaannya dan mudah untuk di mengerti.

   Adapun Notasi *Sequence Diagram* dapat dilihat pada Tabel II.7.

Tabel II.7. Notasi *Sequence Diagram*

Notasi	Keterangan
	Lifeline mengindikasikan keberadaan sebuah object dalam basis waktu.
	*Activation* dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah *lifeline*. *Activation* mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.
	*Message*, digambarkan dengan anak panah horizontal antara *Activation*. *Message* mengindikasikan komunikasi antara objek-objek.
	*Self-message* atau panggilan mandiri mengindikasikan komunikasi kembali kedalam sebuah objek itu sendiri.
	Operator *loop* adalah *fragmen* yang dapat mengeksekusi berulang kali dan *message* penjaga menunjukkan dasar iterasi.

II.4. Web Server

Web server adalah server yang melayani permintaan klien terdapat halaman web seperti *apache,* IIS *(Internet Information Server)* dan berkomunikasi dengan *Middleware* untuk menterjemahkan kode-kode tertentu, menjalankan kode-kode tersebut dan memungkinkan berinteraksi dengan basis data, PHP atau ASP. Adapun arsitektur aplikasi server adalah sebagai berikut :

a. *Browser* atau *client* berinteraksi dengan *web server*.

b. Secara internal *web server* berinteraksi dengan *middleware***.**

c. *Middleware* yang berhubungan dengan database. Adapun teknologi yang berjalan di server antara lain : CGI *(Common Gateway Interface),* ASP *(Aktive Server Page*), JSP *(Java Server Page)* dan *PHP*. (Arief Ramadhan ; 2006: 32).

II.4.1. *Web Browser*

Web browser adalah perangkat lunak (*software)* disisi klien yang digunakan untuk mengakses informasi web, memformat teks dan menempatkan grafik pada layer. Ada beberapa jenis *web browser* yang dipakai sebagai sumber tampilan antara lain : *Internet Explorer, Netscape Navigator, Mozila, Opera*. Prinsip kerja pengaksesan sebuah halaman *web* yang berbasisi HTML adalah sebagai berikut :

a. *Browser* meminta sebuah halaman kesuatu situs *web* melalui protokol http.

b. *Web server* menerima permintaan

c. *Web server* segera mengirimkan *dokumen* HTML yang diminta ke klien.

d. *Browser* pada klien akan membaca dan mengartikan dokumen yang diterima berdasarkan kode-kode pemformatan yang terdapat pada dokumen HTML, lalu menampilkan dalam versinya masing-masing. (Arief Ramadhan ; 2006: 32)

II.5. Sejarah Singkat PHP (*Hypertext Preprocessing*)

Rumus Lerdoft, merasa kurang puas dengan sistem yang ada pada saat itu sehingga dia menciptakan suatu model interface (antarmuka) yang dapat digunakan untuk menampung informasi tentang para pengunjung situsnya. Pertama kali, Rasmus membuat interface dengan mengunakan PERL dan selajutnya dia mengembangklan dejngan bahasa C untuk memberikan floeksibel pada interface/parser tersebut.

Pada mulawanya, interface tersebut diberi nama *Personal Home Page* yang memiliki kemampuan utnuk mencatat seluruh informasi dari pengunjung situs online-nya. Kemudian interface atau parser tersebut dimodifikasi dengan mendukung data base mSQL atau mini struktur Query Language dengan menggunakan parser SQL, pengembangan ini diberi nama FI (Form Interpreter). Kemudian PHP/FI 5.x.x. selanjutnya php ini dikembangkan oleh tim untuk memberikan kemampuan yang seimbang dengan aplikasi lainnya.

Saat ini Zend menjadi pengembang utama dan telah mendistribusikan ZEND engine-nya untuk perkembangan PHP. Sampai sekarang, pengguna PHP sudah sangat banyak arena kemudahan dan keadaannya didalam proses pemrograman. (Stendy B. Sakur ; 2010 :4)

II.5.1. Pengertian PHP

PHP adalah sebuah bahasa pemrograman berbasis web yang mempunyai banyak keunggulan dibandingkan dengan bahasa pemrograman berbasis web lain. PHP merupakan bahasa pemrograman yang bersumber dari Perl. Sedangkan Perl merupakan pengembang dari bahasa C. Oleh karenanya, struktur pemrograman

yang ada di PHP sama dengan yang ada di bahasa C. Melihat bahwa PHP merupakan pengembangan dari bahasa C secara tidak langsung, maka PHP mempunyai banyak sekali fitur-fitur yang dapat digunakan. Misalnya, PHP dapat mengakses shell di linux, mempunyai fungsi yang lengkap berhubungan dengan *networking* (Andi Pramono,M.Syafii ; 2005:2).

PHP pertama kali dibuat leh Rasmus Lerdof pada tahun 1995. pada waktu itu PHP masih bernama FI (*Form Interpreted)* yang wujudnya berupa sekumpulan *script* yang digunakan untuk mengolah data *form* dari *web.*

Selanjutnya Rasmus merilis kode sumber tersebut untuk umum dan menamakannya PHP/FI, kependekatan dari *Hypertext Preprocessing/Form Interpreter*. Dengan perilisan kode sumber ini menjadi *open source*, maka banyak *programmer* yang tertarik untuk ikut mengembangkan PHP. Dan selanjutnya berkembang terus hingga PHP 5.0 yang yang dirilis pada Juni 2004 oleh Zend dengan memasukkan model pemograman berorientasi objek ke dalam PHP untuk menjawab perkembangan bahasa pemograman ke arah paradigma berorientasi objek.

Hypertex Preprocessor (PHP) adalah skrip yang berjalan dalam *server side* yang di tambahkan dalam HTML. PHP itu sendiri merupakan singkatan dari *Personal Home Page Tools*. Skrip ini akan membuat suatu aplikasi dapat di integrasikan kedalam HTML sehingga suatu halaman HTML tidak lagi bersifat statis, namun menjadi besifat dinamis. Sifat *server side* ini membuat pengerjaaan skrip tersebut dikerjakan di *server* sedangkan yang dikirimkan kepada *browser* adalah hasil proses dari skrip tersebut yang sudah berbentuk HTML.

Keunggulan dari sifat server side tersebut adalah:

a. Tidak di perlukan adanya kompatibilitas *browser* atau harus menggunakan *browser* tertentu, karena serverlah yang akan mengerjakan skrip tersebut. Hasil yang di kirimkan kembali ke *browser* biasanya dalam bentuk teks ataupun gambar sehingga dapat dikenali oleh *browser* apa pun.
b. Dapat memanfaatkan sumber-sumber aplikasi yang dimiliki oleh *server*, contoh: hubungan kedalam *database*.
c. Skrip asli tidak dapat dilihat sehingga keamanan lebih terjamin.

PHP dibuat pada tahun 1994 oleh Rasmus Lerdorf. Tetapi kemudian dikembangkan oleh orang lain dan setelah melalui tiga kali karya penulisan akhirnya PHP menjadi bahasa pemrograman web. PHP adalah sebuah produk yang bersifat *open source*, sehingga *source code-code* dari PHP dapat digunakan, diganti atau diedit tanpa harus membayar atau dikenai biaya.

Adapun teknik penulisan *script PHP* dapat dilakukan dengan berbagai cara, yaitu :

1. <? *Script* PHP di sini ?> atau
2. <?php *Script* PHP di sini ?> atau
3. <% *Script* PHP di sini %>
4. <SCRIPT languange=”PHP”> *Script* PHP di sini </SCRIPT>

II.6. MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia. MySQL AB membuat

MySQL tersedia sebagai perangkat lunak gratis dibawah lisensi GNU General Public License (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL.

Tidak sama dengan proyek-proyek seperti Apache, dimana perangkat lunak dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, MySQL dimiliki dan disponsori oleh sebuah perusahaan komersial Swedia MySQL AB, dimana memegang hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan MySQL AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

MySQL adalah *Relational Database Management System* (RDBMS) yang didistribusikan secara gratis dibawah lisensi GPL *(General Public License).* Dimana setiap orang bebas untuk menggunakan MySQL, namun tidak boleh dijadikan produk turunan yang bersifat *closed source* atau komersial. MySQL sebenarnya merupakan turunan salah satu konsep utama dalam database sejak lama, yaitu SQL *(Structured Query Language).* SQL adalah sebuah konsep pengoperasian database, terutama untuk pemilihan atau seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Keandalan suatu sistem database (DBMS) dapat diketahui dari cara kerja optimizer-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh user maupun program-program aplikasinya. Sebagai *database server*, MySQL dapat dikatakan lebih unggul dibandingkan *database server* lainnya dalam *query*

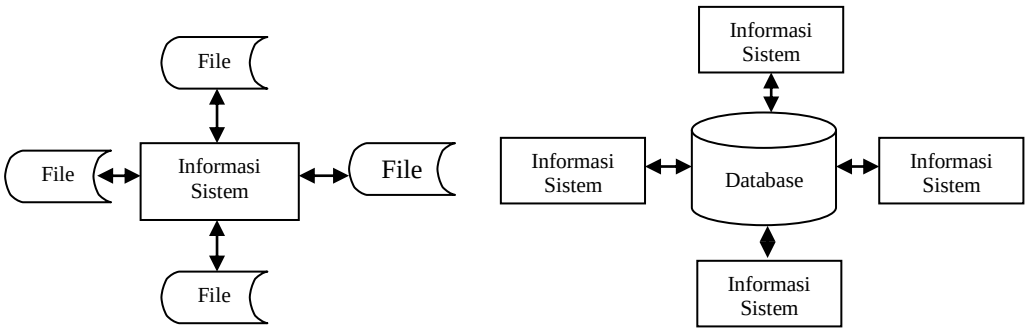
data. Hal ini terbukti untuk query yang dilakukan oleh *single user*, kecepatan *query* MySQL bisa sepuluh kali lebih cepat dari PostgreSQL dan lima kali lebih cepat dibandingkan Interbase.

II.7. Database

Semua sistem informasi membuat, membaca, memperbaharui, dan menghapus data. Data disimpan di dalam file dan database. File adalah sebuah kumpulan record yang serupa. Contohnya mencakup file customer, file order, dan produk file (Jeffry L. Whitten ; 2006 : 518).

Database adalah kumpulan file yang saling terkait.. Database tidak hanya merupakan kumpulan file. Record pada setiap file harus memperbolehkan hubungan-hubungan anggaplah sebagai pointer untuk menyimpan file-file lain. Contohnya database sales mungkin terdiri dari record order yang terhubung ke record customer dan record produk yang terkait (Jeffry L. Whitten ; 2006 : 518).

Pada lingkungan file, data storage dibangun disekitar aplikasi yang akan menggunakan file-file. Pada lingkungan database, aplikasi dibangun disekitar database yang sudah diintegrasikan. Pada akhirnya database tidak begitu tergantung pada aplikasi-aplikasi yang akan menggunakannya. Dengan kata lain, pada sebuah database yang akan ditentukan, dapat dibuat aplikasi-aplikasi baru untuk berbagi database tersebut. Setiap lingkungan memiliki keuntungan dan kelemahannya. Membandingkan file dan database dapat dilihat pada Gambar II.4



Gambar II.4 File Konvensional Versus Database
Sumber : Jeffry L. Whitten, 2006