BAB II

TINJAUAN PUSTAKA

2.1 Tinjauan Instansi

CV Putra Sentosa merupakan perusahaan yang bergerak dalam bidang jasa pelayanan dalam perjalanan atau *traveling* baik di daerah pegunungan maupun di laut. Penelitian juga sering dilakukan terhadap pelestarian habitat darat dan laut demi menunjang kelangsungan kehidupan biota. CV Putra sentosa ikut serta dalam pelestarian lingkungan untuk ikut mensukseskan program pemerintah dalam pelestarian lingkungan.

CV Putra Sentosa melalui program Pariwisata dan Pendidikan berusaha memadukan aktivitas wisata air dengan kegiatan konservasi kelautan, dengan tujuan:

1. Memberi peluang kepada penyelam rekreasi untuk mengetahui lebih jauh tentang kelautan yang mereka cintai dan mengajak mereka terlibat langsung dalam konservasi.
2. Mendorong terciptanya pariwisata berkelanjutan, yang menjamin kelestarian biota dan pemanfaatannya bagi perekonomian masyarakat.

2.2 Landasan Teori

Pada landasan teori ini akan diuraikan sejumlah teori-teori yang membantu dan memecahkan permasalahan yang ada. Beberapa landasan teori tersebut meliputi pengertian informasi, pengertian sistem informasi geografis, konsep model data spasial sig, pengenalan php, konsep kerja php, sisntak dasar php, pengenalan mapserver, komponen pembentul mapserver, kompenen untuk akses data spasial, komponen untuk akses data peta, komponen pendukung, struktur file map, pemrograman dengan php/mapscript dan database mysql.

2.2.1 Pengertian Informasi

Informasi dapat didefinisikan sebagai hasil dari pengolahan data dalam suatu bentuk yang lebih berguna dan lebih berarti bagi penerimanya yang menggambarkan suatu kejadian-kejadian (event) yang nyata (fact) yang digunakan untuk mengambil keputusan[3].

Sumber dari informsi adalah data. Data adalah kenyataan yang menggambarkan suatu kejadian-kejadian dan kesatuan nyata. Kejadian-kejadian adalah sesuatu terjadi pada saat yang tertentu.[4]

2.3 Sistem Informasi Geografis (SIG)

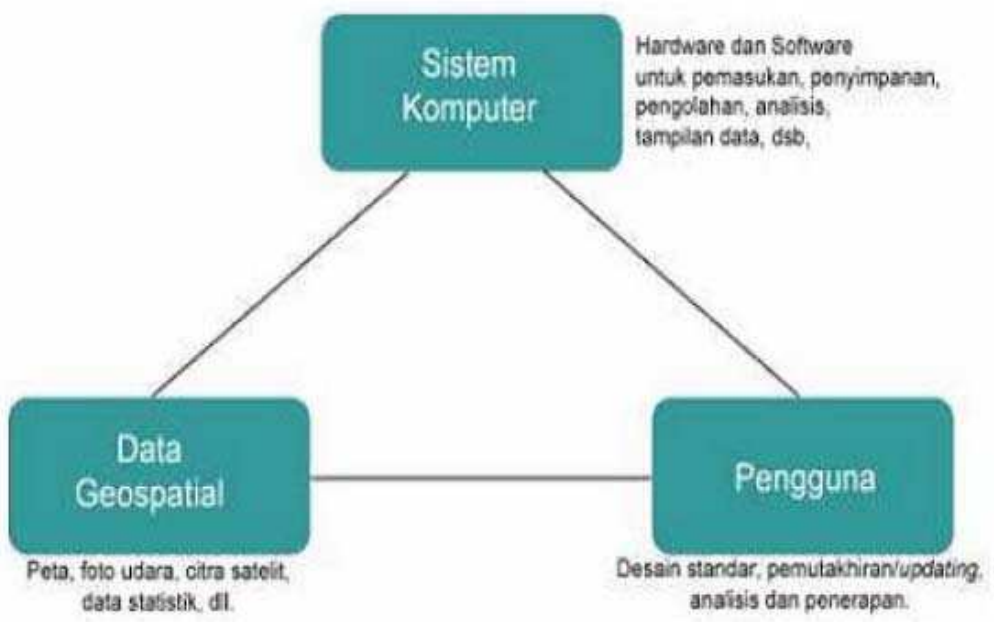
2.3.1 Pengenalan SIG

Sistem Informasi Geografi (SIG) adalah sebuah alat bantu manajemen berupa informasi dengan media komputer yang berkaitan erat dengan sistem pemetaan dan analisis terhadap segala sesuatu serta peristiwa – peristiwa yang terjadi di muka bumi. Teknologi SIG mengintegrasikan operasi pengolahan data berbasis database yang terintegrasi, seperti pengambilan data berdasarkan kebutuhan, serta analisis statistik dengan menggunakan visualisasi serta berbagai keuntungan yang mampu ditawarkan melalui analisis geografis melalui gambar-gambar petanya.

Definisi SIG selalu berkembang, bertambah dan bervariasi. Hal ini telihat dari banyaknya definisi SIG yang telah beredar. Selain itu, SIG juga merupakan suatu kajian ilmu dan teknologi yang relatif baru, digunakan oleh berbagai bidang disiplin ilmu, dan berkembang dengan cepat.

Dari definisi yang ada, diambil satu buah definisi yang dapat mewakili SIG secara umum yaitu sistem informasi yang digunakan untuk memasukkan, menyimpan, memanggil kembali, mengolah, menganalisa dan menghasilkan data bereferensi geografi atau data geospatial, untuk mendukung pengambilan keputusan dalam perencanaan dan pengolahan.

Komponen SIG adalah sistem komputer, data geospatial dan pengguna, seperti pada Gambar 2.1



Gambar 2.1 Komponen SIG

Data yang diolah pada SIG ada 2 macam yaitu data geospasial (data spasial dan data non-spasial). Jika pada gambar diatas data non-spasial tidak digambarkan karena memang dalam SIG yang dipentingkan adalah tampilan data secara spasial.

Data spasial adalah data yang berhubungan dengan kondisi geografi misalnya sungai, wilayah administrasi, gedung, jalan raya dan sebagainya. Seperti yang telah diterangkan pada gambar diatas, data spasial didapatkan dari peta, foto udara, citra satelit, data statistik dan lain-lain.

Hingga saat ini secara umum persepsi manusia mengenai bentuk representasi entity spasial adalah konsep raster dan vector. Sedangkan *data non-spasial* adalah selain data spasial yaitu data yang berupa text atau angka. Biasanya disebut dengan atribut.

Data non-spasial ini akan menerangkan data spasial atau sebagai dasar untuk menggambarkan data spasial. Dari data non-spasial ini nantinya dapat dibentuk data spasial.

2.3.2 Konsep Model Data Spasial pada SIG

Data spasial merupakan data yang paling penting dalam SIG. Data spasial ada 2 macam yaitu data raster dan data vektor.:

1. Data Raster

   Model data raster menampilkan, menempatkan dan menyimpan spasial dengan menggunakan struktur matriks atau pixel-pixel yang membentuk grid. Akurasi model data ini sangat bergantung pada resolusi atau ukuran pixelnya (sel grid) di permukaan bumi. Contoh data raster adalah citra satelit misalnya Spot, Landsat, dll.

   Konsep model data ini adalah dengan memberikan nilai yang berbeda untuk tiap-tiap pixel atau grid dari kondisi yang berbeda

2. Data Vektor

   Model data vektor yang menampilkan, menempatkan dan menyimpan data spasial dengan menggunakan titik-titik, garis-garis, atau kurva atau poligon beserta atribut-atributnya. Bentuk dasar representasi data spasial didalam sistem model data vektor, didefinisikan oleh sistem koordinat kartesian dua dimensi (x,y).

2.4 Pengenalan PHP

PHP merupakan bahasa berbentuk script yang disertakan dalam dokumen HTML, bekerja di sisi server sehingga script-nya tak tampak di sisi client. PHP dirancang untuk dapat bekerja sama dengan database server dan dibuat sedemikian rupa sehingga pembuatan dokumen HTML yang dapat mengakses database menjadi begitu mudah atau secara umum dokumen yang dihasilkan adalah dokumen WEB Dinamis.

Pada saat ini PHP cukup popular sebagai piranti pemrograman WEB di lingkungan Linux. Walaupun demikian PHP sebenarnya juga dapat berfungsi pada server-server yang berbasis UNIX, Windows dan Macintosh. Pada awalnya PHP dirancang untuk berintegrasi dengan Web Server Apache, tetapi sekarang ini PHP juga bekerja pada Web Server lainnya seperti IIS dan PWS.

PHP bersifat freeware, artinya bebas untuk dipakai tanpa harus membayar lisensi. Anda dapat mendownloadnya pada situs www.php.net.

2.4.1 Konsep Kerja PHP

Model kerja HTML diawali dengan permintaan suatu halaman web oleh browser. Berdasarkan URL atau dikenal dengan sebutan alamat internet, browser mendapatkan alamat dari web server, mengidentifikasi halaman yang dikehendaki dan menyampaikan segala informasi yang dibutuhkan oleh web server. Informasi yang disampaikan ke web server antara lain adalah nama browser, versinya dan sistem operasinya. Selanjutnya web server akan mencarikan berkas yang diminta dan

memberikan isinya ke browser. Browser yang mendapatkan isinya segera melakukan proses penterjemahan kode HTML dan menampilkan ke layar pemakai.

Bagaimana halnya kalau yang diminta adalah sebuah halaman PHP Prinsipnya serupa dengan kode HTML, hanya saja ketika berkas PHP dan mesin inilah yang memproses dan memberikan hasilnya (Berupa kode HTML) ke web Server untuk selanjutnya disampaikan ke client yang request.

2.4.2 Sintaks Dasar PHP

Sebagai penanda yang memberitahukan parser PHP bahwa sebuah skrip menggunakan bahasa PHP dapat dilakukan dengan cara berikut :

a) Menggunakan tanda <? .. ?>
b) Menggunakan tanda <?php ... ?>
c) Menggunakan perintah seperti pada Javascript
d) Contoh :
e) <script language=”php”> ... </script>
f) Menggunakan tanda <% ... %>

Dari seluruh cara tersebut dapat dilihat bahwa tanda penutup harus sama dengan tanda pembukanya. Misalnya jika tanda pembukanya (<?) maka tanda pentutpnya adalah (?>). Fungsi dari tanda penutup ini adalah untuk memberikan tanda bahwa sebuah statement atau skrip PHP telah selesai dilengkapi.

Komentar Komentar pada PHP juga mirip dengan komentar pada C atau C++ maupun shell Unix. Untuk membuat komentar satu baris dapat digunakan tanda // atau # seperti pada shell Unix. Sedangkan untuk komentar yang terdiri dari banyak baris dapat digunakan tanda /* ... */.

2.5 MapSever

2.5.1 Pengenalan MapSever

MapServer merupakan aplikasi *freeware* dan *open source* yang memungkinkan kita menampilkan data spasial (peta) di web. Aplikasi ini pertama kali dikembangkan di Universitas Minesotta, Amerika Serikat untuk proyek *ForNet* (sebuah proyek untuk menajemen sumber daya alam) yang disponsori NASA (*Nasional Aeronautics and Space Administration*). Dukungan NASA dilanjutkan dengan dikembangkan proyek *TerraSIP* untuk menajemen data lahan. Saat ini, karena sifatnya yang terbuka (*open source*), pengembangan MapServer dilakukan oleh pengembang dari berbagai negara.

Pengembangan MapServer menggunakan berbagai aplikasi *open source* atau *freeware* seperti Shapelib untuk baca/tulis format data *Shapefile*, FreeType untuk merender karakter, GDAL/OGR untuk baca/tulis berbagai format data vektor maupun raster, dan Proj.4 untuk menangani beragam proyeksi peta.

Pada bentuk paling dasar, MapServer berupa sebuah program CGI (*Common Gateway Interface*). Program tersebut akan dieksekusi di *web server* dan berdasarkan beberapa parameter tertentu (terutama konfigurasi dalam bentuk file *.MAP) akan

menghasilkan data yang kemudian akan dikirim ke web browser, baik dalam bentuk gambar peta atau bentuk lain.

MapServer mempunyai fitur-fitur berikut :

a) Menampilkan data spasial dalam format vektor seperti : *Shapefile* (ESRI), ArcSDE (ESRI), PostGIS dan berbagai format data vektor lain dengan menggunakan *library* OGR
b) Menampilkan data spasial dalam format raster seperti : TIFF/GeoTIFF, EPPL7 dan berbagai format data raster lain dengan menggunakan *library* GDAL
c) Menggunakan *quadtree* dalam *indexing* data spasial, sehingga operasi-operas spasial dapat dilakukan dengan cepat.
d) Dapat dikembangkan (*customizable*), dengan tampilan keluaran yang dapat diatur menggunakan file-file *template*
e) Dapat melakukan seleksi objek berdasar nilai, berdasar titik, area, atau berdasar sebuah objek spasial tertentu
f) Mendukung *rendering* karakter berupa *font* TrueType
g) Mendukung penggunaan data raster maupun vektor yang di-*tiled* (dibagi-bagi menjadi sub bagian yang lebih kecil sehingga proses untuk mengambil dan menampilkan gambar dapat dipercepat)
h) Dapat menggambarkan elemen peta secara otomatis : skala grafis, peta indeks dan legenda peta

i) Dapat menggambarkan peta tematik yang dibangun menggunakan ekspresi logik maupun ekspresi reguler.

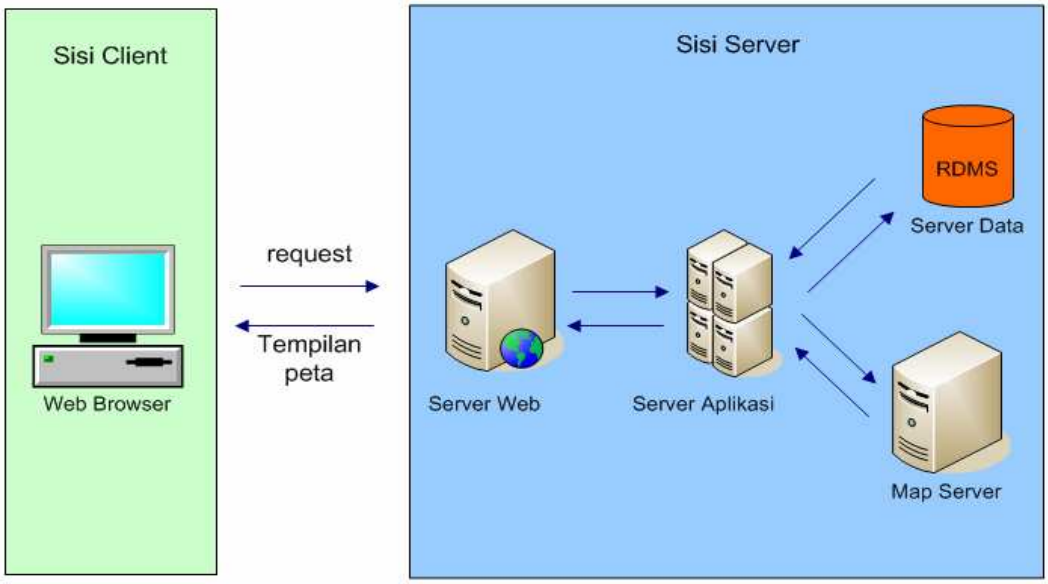
j) Dapat menampilkan label dari objek spasial, dengan label dapat diatur sedemikian rupa sehingga tidak saling tumpang tindih

k) Konfigurasi dapat diatur secara *on the fly* melalui parameter yang ditentukan pada URL.

l) Dapat menangani beragam system proyeksi secara *on the fly*

Saat ini, selain dapat mengakses MapServer sebagai program CGI, kita dapat mengakses MspServer sebagai modul MapScript, melalui berbagai bahasa skrip : PHP, Perl, Phyton atau Java. Akses fungsi-fungsi MapServer melalui skrip akan lebih memudahkan pengembangan aplikasi. Pengembang dapat memilih bahasa yang paling familiar.

Arsitektur Umum Aplikasi Pemetaan di Web Bentuk umum arsitektur aplikasi berbasis peta di web dapat dilihat pada Gambar 2.3.



Gambar 2.2 Arsitektur Umum Aplikasi Pemetaan Berbasis Web

Pada gambar di atas, interaksi antara klien dengan server berdasar skenario *request* dan *respon*. Web browser di sisi kilen mengirim *request* ke server web. Karena server web tidak memiliki kemampuan pemrosesan peta, maka *request* berkaitan dengan pemrosesan peta akan diteruskan oleh server web ke server aplikasi dan MapServer. Hasil pemrosesan akan dikembalikan lagi melalui server web, terbungkus dalam bentuk file HTML atau *applet*.

Arsitektur aplikasi pemetaan di web dibagi menjadi dua pendekatan sebagai berikut :

Pendekatan *Thin Client*

Pendekatan ini menfokuskan diri pada sisi server. Hampir semua proses dan analisis data dilakukan berdasarkan request di sisi server. Data hasil pemrosesan kemudian dikirimkan ke klien dalam format standard HTML, yang di dalamnya terdapat file gambar dalam format standard (misalnya GIF, PNG atau JPG) sehingga

dapat dilihat menggunakan sembarang *web browser*. Kelemahan utama pendekatan ini menyangkut keterbatasan opsi interaksi dengan user yan kurang fleksibel.

Pendekatan *Thick Client* Pada pendekatan ini, pemrosesan data dilakuakn di sisi klien menggunakan beberapa teknologi seperti kontrol ActiveX atau applet. Kontrol ActiveX atau applet akan dijalankan di klien untuk memungkinkan web browser dengan format data yang tidak dapat ditangani oleh web browser dengan kemampuan standard. Dengan adanya pemrosesan di klien, maka transfer data antara klien dengan web server akan berkurang.

MapServer menggunakan pendekatan *thin client*. Semua pemrosesan dilakukan di sisi sever. Informasi peta dikirinkan ke web browser di sisi klien dalam bentuk file gambar (JPG, PNG, GIF atau TIFF). Untungnya, saat ini kelemahan pendekatan *thin client* dalam hal interaksi dengan user sudah jauh berkurang dengan adanya franework aplikasi seperti *Chameleon* atau *CartoWeb*.

2.5.2 Komponen Pembentuk MapSever

Pengembangan MapSever sebagai sebuah aplikasi open source, banyak memanfaatkan aplikasi lain yang juga bersifat open source. Sedapat mungkin menggunakan aplikasi yang sudah tersedia jika memang memenuhi kebutuhan untuk menghemat sumber daya dan waktu pengembangan.

2.5.1.1 Komponen Untuk Akses Data Spasial

Komponen pada kelompok ini bertugas untuk menangani baca/tulis data spasial, baik yang tersimpan sebagai file maupun tersimpan pada DBMS

a) Shapelib

Shapefile merupakan library yang ditulis dalam bahasa C, untuk keperluan baca/tulis format data *Shapefile* (*.SHP) yang didefinisikan ESRI (*Enviromental System Research Institute*). Format *Shapefile* umum digunakan oleh berbagai aplikasi Sistem Informasi Geografik untuk menyimpan data vector simple (tanpa topologi) dengan atribut. Pada MapSever, format data *Shapefile* merupakan format data *default*.

b) GDAL/OGR

GDAL (*Geographic Data Abstraction LibraryI*) merupakan *library* yang berfungsi sebagai penerjemah untuk berbagai format data raster. *Library* ini memungkinkan abstraksi untuk semua format data yang didukung, sehingga beragam format data tadi akan terlihat sebagai sebuah data model abstrak. Keberadaan data model abstrak tunggal akan memudahkan pengembang aplikasi karena dapat menggunkan antarmuka yang seragam untuk semua format data.. OGR merupakan *library* dengan fungsionalitas yang identik, untuk beragam format data vektor. Kode OGR sekarang ini digabung dalam kode *library* GDAL.

2.5.1.2 Komponen Untuk Akses Data Peta

MapServer akan mengirimkan tampilan peta berupa gambar. Kita dapat memilih apa format data gambar yang akan digunakan. Beberapa komponen di bawah ini berperan dalam membentuk gambar peta yang dihasilkan oleh MapSever.

a) Libpng

Libpng merupakan *library* yang digunakan untuk baca/tulis gambar dalam format PNG

b) Libjpeg

Libjpeg merupakan *library* yang digunakan untuk baca/tulis gambar dalam format JPG/JPEG

c) GD

Library GD digunakan MapServer untuk menggambar objek geografis seperti garis, poligon atau bentuk geometris lain. GD juga dapat dapat digunakan untuk menghasilkan gambar dalam format PNG, JPEG, selain menggunakan *libpng* atau *libjpeg* secara langsung.

d) FreeType

FreeType merupakan library yang digunakan MapServer untuk menampilkan tulisan menggunakan *font TrueType.*

2.5.3 Struktur File MAP

MapServer menggunakan file *.MAP (file dengan akhiran .map, misalnya *jawa*.map) sebagai file konfigurasi peta. File ini akan berisi komponen tampilan peta seperti definisi layer, definisi proyeksi peta, pengaturan legenda, skala dan sebagainya. Secara umum, file *.map memiliki beberapa karakteristik sebagai berikut :

a) Berupa file teks
b) Tidak *case sensitif* (tidak membedakan antara karakter yang ditulis dengan huruf besar atau huruf kecil), sebagai contoh : kata ”LAYER”, ”layer” maupun ”Layer” memiliki arti yang sama pada file *.map. Hal ini tidak berlaku bagi penamaan atribut, misalnya nama field pada sebuah *Shapefile* (file *.shp). Nama filed harus dituliskan persisi seperti yang tertulis pada sumbernya. Meskipun tidak case sensitif, sebaiknya kita menentukan aturan penggunaan huruf besar atau kecil untuk menjaga konsistensi. Pada umumnya digunakan huruf besar untuk menuliskan isi file *.map
c) Teks yang mengandung karakter bukan alfanumerik (huruf dan angka), harus berada di dalam tanda petik, misalnya : ”/opt/webgis/map” (karena karakter ’/’ bukan karakter alfanumerik). Meskipun keharusan ini hanya berlaku untuk teks yang mengandung karakter bukan alfanumerik, sebaiknya kita secara konsisten menggunakan tanda petik untuk setiap variabel teks.
d) Path yang menunjuk ke sebuah file, harus dituliskan dalam bentuk path absolut, misalnya */opt/webgis/data/batimeri.tif,* atau relatif terhadap lokasi file *.map (misalnya *./data/batimeri.tif*).

e) Pada kondisi normal, jumlah definisi layer pada sebuah file *.map maksimum sebanyak 50 buah, kecuali kita melakukan kompilasi program MapServer sendiri dan secara eksplisit mengubah definisi ini.

f) Komentar pada MapServer dimulai dengan karakter '#'. Teks yang berada setelah karakter tersebut akan diabaikan, kecuali jika karakter '#' berada di dalam tanda petik dan menjadi bagian dari variabel teks.

2.6 Pemrograman Dengan PHP/Mapscript

MapScript adalah antarmuka pemrograman MapServer. Saat ini MapScriptd tersedia dalam beberapa bahasa pemrograman : PHP, Perl, Phyton dan Ruby. Antarmuka MapScript menggunakan bahasa pemrograman PHP disebut dengan PHP/MapScript. PHP/MapScript memungkinkan kita melakukan akses terhadap MapScript API (*Application Programing Interface*) dari lingkungan PHP, dengan menggunakan berbagai kelas PHP.

PHP/MapScript tersedia sebagai sebuah modul PHP, dalam bentuk file DLL (*Dynamic Linked Library*) pada platform Windows, atau dalam bentuk ahared object pada platform Linux.

Seperti juga bahasa pemrograman PHP itu sendiri, modul PHP/MapScript disusun menggunakan pendekatan pemrograman berorientasi objek (*Object Oriented Programing*, atau biasa disingkat OOP). Jadi ketika kita bekerja dengan PHP/MapScript, kita akan bekerja dengan berbagai kelas, disamping beberapa fungsi dan variable khusus.

Berikut deskripsi singkat kelas-kelas pada PHP/Mapscript :

a) ClassObj

   Mengatur kelas-kelas pada sebuah layer.

b) ColorObj

   Mengatur tampilan warna.

c) ErrorObj

   Manajemen kesalahan (*error*). MapSever akan secara otomatis menginisiasi *ErrorObj* ketika terjadi kesalahn

d) ImageObj

   Mengatur penggambaran peta pada file.

e) LabelCacheObj

   Digunakan untuk keperluan membersihkan memori yang dialokasikan untuk *cache* label.

f) LabelObj

   Mengatur kenampakan label.

g) LayerObj

   Mengatur operasi-operasi yang berhubungan dengan layer: pembuatan, penggambaran, query.

h) LegendObj

   Mengatur tampilan legenda.

i) LineObj

   Mengatur objek peta berupa garis.

j) MapObj

Mengatur karakteristik peta yang akan ditampilkan.

k) OutputFormatObj

Mengatur format gambar hasil keluaran MapServer.

l) PointObj

Mengatur objek berupa titik.

m) ProjectionObj

Mengatur penggunaan system proyeksi peta.

n) RectObj

Mengatur objek peta berupa persegi panjang (*rectangle*).

o) ReferenceMapObj

Mengatur karakteristik peta indeks.

p) ResultCacheMemberObj

Mengatur objek-objek hasil query.

q) ScalebarObj

Mengatur karakteristik skala grafis.

r) ShapefileObj

Kelas untuk bekerja dengan data dalam format *Shapefile.*

s) ShapeObj

Kelas untuk bekerja dengan objek peta, baik berupa titik, garis maupun *polygon.*

t) StyleObj

Mengatur karakteristik tampilan symbol.

u) SymbolObj

Mengatur karakteristik symbol.

v) WebObj

Mengatur opsi-opsi berhubungan dengan web, misalnya *logging* aktivitas.

2.7 Database MySQL

2.7.1 Pengenalan MySQL

MySQL merupakan software resmi yng dikembangkan oleh perusahaan Swedia bernama MySQL AB,yang waktu itu bernama TcX Data Konsult AB.Pada awalnya MySQL memakai nama mSQL atai “mini SQL” sebagai antarmuka yang digunakan, ternyata dengan menggunakan mSQL itu mengalami banyak hambatan, yaitu sangat lambat dan tidak fleksibel. Oleh karena itu, Michael Widenius (“Monty”), panggilan akrabnya, berusaha mengembangkan interface yang tersebut hingga ditemukan MySQL. Kala itu, MySQL didistribusikan secara khusus, yakni untuk keperluan nonkomersial bersifat gratis, sedangkan untuk kebutuhan komersial diharuskan membayar lisensi. Barulah sejak versi 3.23.19, MySQL dikategorikan software berlisensi GPL, yakni dapat dipakai tanpa biaya untuk kebutuhan apapun.
Hingga kini, MySQL dapat dijalankan di berbagai sistem operasi misalnya Linux, Unix, Windows. Kelebihan MySQL adalah pada kecepatan akses, biaya, konfigurasi, tersedia source code karena MySQL dibawah Open Source License dan menjadi

database open source yang sangat popular. MySQL merupakan database relational (RDBMS) yang mempunyai kemampuan yang sangat cepat untuk menjalankan SQL dengan multithread dan multiuser. Oleh karena itu, dengan dengan melihat begitu tingginya potensi MySQL untuk dijadikan sebagai database yang andal, segala feature pendukung terus dikembangkan agar penggunaan MySQL dapat lebih optimal lagi.

SQL *(Structured Query Language)* merupakan bahasa query yang digunakan untuk mengakses database relasional. SQL sekarang sudah menjadi bahasa database standard dan hampir semua sistem database memahaminya. SQL terdiri dari berbagai jenis statemen. Semuanya didesain agar dia memungkinkan untuk dapat secara interaktif berhubungan dengan database.

Penggunaan SQL pada DBMS (Database Management System) sudah cukup luas. SQL dapat dipakai oleh berbagai kalangan, misalnya DBA (Database Administrator), progammer ataupun pengguna. Hal ini disebabkan karena :

a. SQL sebagai bahasa administrasi database

   Dalam hal ini SQL dipakai oleh DBA untuk menciptakan serta mengendalikan pengaksesan database.

b. SQL sebagai bahasa query interaktif Pengguna dapat memberikan perintah-perintah untuk mengakses database yang sesuai dengan kebutuhannya.

c. SQL sebagai bahasa pemrograman database Pemrogram dapat menggunakan perintah-perintah SQL dalam program aplikasi yang dibuat.

d. SQL sebagai bahasa klien/server

SQL juga dipakai sebagai untuk mengimplementasikan sistem klien/server. Sebuah klien dapat menjalankan suatu aplikasi yang mengakses database. Dalam hal ini sistem operasi antara server dan klien bisa berbeda.

Di samping hal tersebut diatas SQL juga diterapkan pada internet atau intranet untuk mengakses database melalui halaman-halaman web untuk mendukung konsep web dinamis.

2.7.2 Kelompok Pernyataan SQL

Pernyataan SQL dapat dikelompokkan menjadi 5 kelompok DDL, DML, DCL, pengendali transaksi dan pengendali programatik.

1. D**DL (Data Definition Language)**

DDL merupakan kelompok perintah yang berfungsi untuk mendefinisikan atribut-atribut database, table, atribut (kolom), batasan-batasan terhadap suatu atribut serta hubungan antar table. Yang termasuk kelompok DDL ini adalah :

CREATE untuk menciptakan table atau indeks

ALTER untuk mengubah struktur table

DROP untuk menghapus table atau indeks

2. DML (Data Manipulation Language)

Adalah kelompok perintah yang berfungsi untuk memanipulasi data, misalnya untuk pengambilan, penyisipan pengubahan dan penghapusan data. Yang termasuk DML adalah :

SELECT untuk memilih data

INSERT untuk menambah data

DELETE untuk menghapus data

UPDATE untuk mengubah data

3. DCL (Data Control Language)

Berisi perintah-perintah untuk mngendalikan pengaksesan data. Yang termasuk DCL adalah :

GRANT untuk memberikan kendali pada pengaksesan data

REVOKE untuk mencabut kemampuan pengaksesan data

LOCK TABLE untuk mengunci table

2.8 Pemrograman Berorientasi Objek (OOP)

Pemrograman diperuntukkan bagi implementasi untuk memenuhi kebutuhan-kebutuhan yang dispesifikasikan dengan rancangan rinci. Meskipun rancangan rinci seharusnya telah mencukupi sebagai dokumen untuk pemrograman, namun umumnya tim pemrogram memeriksa semua dokumen yang sebelumnya pada waktu yang sama

(arsitektur dan dokumen spesifikasi kebutuhan) untuk membantu pemeriksaan inkosistensi di antara dokumen-dokumen itu.

Booch menyatakan :

Pemrograman berorientasi objek adalah metode implementasi di mana program diorganisasikan sebagai kumpulan objek yang bekerja sama, masing-masing objek merepresentasikan instant dari kelas, dan kelas-kelas itu anggota suatu hirarki kelas-kelas yang disatukan lewat keterhubungn pewarisan.

Tiga aspek penting dalam pemrograman berorientasi objek :

1. Menggunakan objek-objek bukan algoritma-algoritma sebagai blok-blok bangunan logik dasar.
2. Masing-masing objek adalah instan suatu kelas.
3. Kelas-kelas saling berhubungan lewat keterhubungan pewarisan.

2.8.1 Hubungan OOA, OOD dan OOP

Hasil analisis berorientasi objek dapat menjadi model dimana kita akan memenuhi perancangan berorientasi objek. Hasil perancangan berorientasi objek dapat digunakan sebagai blue print untuk implementasi sistem dengan sepenuhnya menggunakan metode bahasa berorientasi objek.

Manfaat utama pendekatan berorientasi objek :

1. Dapat menangani domain masalah yang lebih besar. OOA memberi pemahaman domain masalah yang bagus, OOD dan OOP masih menjaga semantiks domain masalah itu.
2. Meningkatkan interaksi antar pakar masalah, analisis, perancang dan pemrogram.
3. Meningkatkan konsistensi internal antara analisis, perancangan dan pemrograman. OOD mereduksi jarak antara aktivitas-aktivitas berbeda itu dengan antarmuka di kelas.
4. Membangun sistem yang tahan terhadap perubahan.
5. Penggunaan ulang hasil-hasil OOA, OOD dan OOP.
6. Menyediakan representasi yang konsisten di OOA,OOD dan OOP.

2.8.2 Bahasa Pemrograman Berorientasi Objek

Bahasa-bahasa yang dikembangkan saat ini hampir seluruhnya mempedulikan pendekatan / konsep berorientasi objek.

Bahasa baru seperti PHP telah mengakomodasikan beberapa konsep berorientasi objek.

Berikut ini akan dijelaskan karakteristik *Object Oriented Programming* (OOP) pada PHP :

1. Abstraksi dan Enkapsulasi

   Abstraksi dalam informasi dicapai melalui definisi kelas yang memiliki data (berupa atribut) dan service (berupa method). Namun data hiding belum didukung

sehingga kelas, atribut dan method dapat diakses oleh kelas manapun. Selain itu belum terdapat konsep package yang dapat membungkus beberapa kelas dalam 1 unit.

2. Inheritance

   Suatu kelas dapat mewarisi atribut dan method dari kelas lain dengan cara melakukan extend pada kelas parent.

3. Instansiasi

   Kelas di intansi menjadi objek denga keyword new. Namun PHP secara default mengembalikan copy objek dan bukan reference ke objek tersebut.

4. Konstruktor

   Konstruktor pada PHP berupa method yang memiliki nama yang sama dengan nama kelasnya dan otomatis dipanggil pada saat instansiasi kelas tersebut.

5. Polymorphisme

   Variabel pada PHP bersifat juggling, artinya penentuan terhadap variable ditentukan pada saat run time dan bergantung pada konteks penggunaan variable tersebut karena variable dapat diberikan tipe dasar maupun objek.

2.9 UML

UML (*Unified Modelling Language*) sebagai bahasa standar pemodelan visual (visual modelling) dalam rekayasa software, memberikan cara standar untuk menggambar cetak biru software yang dibangun. Penggunaan UML berdampak pada peningkatan produktivitas dan kualitas, serta pengurangan biaya dan waktu.

Kerumitan dalam segi arsitektural sistem software yang dibangun juga dapat diatasi dengan menggambarkan cetak biru sistem tersebut.

UML adalah bahasa grafis untuk mendokumentasi, menspesifikasikan, dan membangun sistem perangkat lunak atau bahasa pemodelan untuk membangun dan mendokumentasikan artifak-artifak dari sistem.

UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks. *Artifact* adalah sepotong informasi yang digunakan atau dihasilkan dalam suatu proses rekayasa software. *Artifact* dapat berupa model, deskripsi, atau software.

Tujuan utama UML diantaranya untuk :

a. Memberikan model yang siap pakai, bahasa pemodelan visual yang ekspresif untuk mengembangkan dan saling menukar model dengan mudah dan dimengerti secara umum.
b. Memberikan bahasa pemodelan yang bebas dari berbagai bahasa pemrograman dan proses rekayasa.
c. Menyatukan praktek-praktek terbaik yang terdapat dalam pemodelan

Untuk membuat suatu model, UML memiliki diagram grafis sebagai berikut :

- Diagram kelas (*Class diagram*)
- Diagram objek (*Object diagram*)
- Diagram komponen (*Component diagram*)
- Diagram *deployment* (*Deployment diagram*)

- Diagram *use case* (*Use case diagram*)
- Diagram sekuen (*Sequence diagram*)
- Diagram kolaborasi (*Collaboration diagram*)
- Diagram *state*chart (*Statechart diagram*)
- Diagram aktivitas (*Activity diagram*)

Berikut ini penjelasan secara singkat mengenai artifact dari UML di atas :

Use Case Diagram

Use case Diagram menjelaskan manfaat sitem jika dilihat menurut pandangan orang yang berada diluar sistem (*actor*). Diagram ini menunjukkan fungsionalitas suatu sistem atau kelas dan bagaimana sistem berinteraksi dengan dunia luar

Use case Diagram digunakan selama proses analisis untuk menangkap *requirements* sistem dan untuk memahami bagaimana sistem seharusnya bekerja. Selama tahap desain, *use case diagram* menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa *use case diagram*.

Beberapa simbol yang digunakan dalam *use case diagram* :

a. Actor

   Actor adalah peran dari seseorang atau sistem lain yang bertukar informasi dengan sistem lainnya.

b. Use Case

Use case adalah urutan dari sebuah transaksi dalam sistem yang berfungsi untuk mendapat *measurable value* (nilai yang terukur) dari *actor* sistem. *Use case* mewakili suatu proses bisnis dan pada umumnya *use case* diberi nama menurut sudut pandang *actor* yang bersangkutan dengan *use case* tersebut dan harus berbentuk kata kerja.

c. Association

Association adalah hubungan antara *actor* dengan *use case* untuk memperlihatkan *instance* yang berinteraksi atau dengan kata lain memperlihatkan komunikasi antara *actor dengan use case*.

d. Extend Relationship

Extend Relationship memperlihatkan hubungan antara *use case,* dimana *use case* yang satu merupakan warisan (*inheritance*) dari use case yang lainnya.

e. Uses / Include Relationship

Uses / Include Relationship memperlihatkan hubungan antar *use case,* dimana *use case* yang satu menggunakan / digunakan oleh *use case* yang lainnya.

Class Diagram

Class diagram membantu kita dalam visualisasi struktur kelas-kelas dari suatu sistem dan merupakan tipe diagram yang paling banyak dipakai. *Class diagram* memperlihatkan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain (dalam *logical view*) dari suatu sistem.

Selama proses analisis, *class diagram* memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan prilaku sistem. Selama tahap desain, *class diagram* berperan dalam menangkap struktur dari semua kelas yang membentuk arsitektur sistem yang dibuat.

Class memiliki tiga area pokok :

a. Nama (*data stereotype*)
b. Atribut
c. Metoda

Atribut dan metoda dapat memiliki salah satu sifat berikut :

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

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung instansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat *run-time*.

Hubungan antar Class, antara lain :

a. Asosiasi

Asosiasi yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa *class* lain, atau *class* yang harus mengetahui eksistensi *class* lain. panah *navigability* menunjukkan arah *query* antar *class*.

b. Agregasi

Agregasi yaitu hubungan yang menyatakan bagian (“terdiri atas...”).

c. Pewarisan

Pewarisan yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang mewarisinya. Kebalikan dari pewarisan adalah generalisasi.

d. Hubungan dinamis

Hubungan dinamis yaitu rangkaian pesan (*message*) yang di-passing dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence diagram* yang akan dijelaskan kemudian

Activity Diagram

Activity diagram memodelkan alur kerja (*workflow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah *flowchart* karena kita dapat memodelkan sebuah alur kerja dari suatu aktifitas ke aktifitas lainnya atau dari suatu aktifitas ke dalam keadaan sesaat (*state*). Seringkali bermanfaat jika kita membuat *activity diagram* terlebih dahulu dalam memodelkan sebuah proses untuk membantu kita memahami proses secara keseluruhan. *Activity*

diagram juga sangat berguna ketika kita ingin menggambarkan perilaku paralel atau menjelaskan bagaimana perilaku dalam berbagai *use case* berinteraksi.

Beberapa simbol yang digunakan pada activity diagram :

a. Initial State

   Initial State digunakan untuk menunjukkan state awal dari objek sebelum sebuah peristiwa terjadi/dimasukkan.

b. State

   State digunakan untuk menunjukkan periode waktu selama seorang aktor memeprlihatkan perilaku tertentu.

c. Transition (Join)

   Trasition (Join) digunakan untuk menggabungkan beberapa *state* yang paralel sehingga menghasilkan outputan yang sama.

d. Transition (Fork)

   Transition (Fork) digunakan untuk menunjukkan bahwa sebuah *state* bisa menghasilkan lebih dari satu *state* yang paralel.

e. Decision

   Decision digunakan untuk menampilkan keputusan dari *state* yang bernilai boolean.

Object Diagram

Kegunaan diagram objek adalah mendeskripsikan bagaimana kumpulan objek tertentu saling berhubungan.

Diagram Objek adalah diagram instant (*instance diagram*) yang mendeskrpisikan instant-instan kelas. Diagram instant berguna untuk dokumentasi skenario, serta kasus pengujian (*test cases*) dan mendiskusikan contoh-contoh instanisasi diagram kelas.

Sequence Diagram

Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa *message* yang digambarkan terhadap waktu. *Sequence diagram* terdiri antar dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait). *Sequence diagram* biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai response dari sebuah event untuk menghasilkan autput tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan.

Masing-masing objek, termasuk aktor, memiliki *lifeline* vertikal. *Message* digambarkan sebagai garis berpanah dari satu objek ke objek lainnya. Pada fase desain berikutnya, *message* akan dipetakan menjadi operasi/metoda dari *class*. *Activition bar* menunjukkan lamanya eksekusi sebuah proses, biasanya diawali dengan diterimanya sebuah *message*.

Beberapa simbol yang digunakan pada *Sequence Diagram* :

a. Actor

   Actor adalah pesan dari seseorang atau sistem lain yang bertukar informasi dengan sistem lainnya.

b. Object Lifeline

   Object lifeline menunjukkan keberadaan dari sebuah objek terhadap waktu. Yaitu objek dibuat atau dihilangkan selama suatu periode waktu diagram ditampilkan, kemudian *lifeline* berhenti atau mulai pada titik yang tepat.

c. Activation

   Activation menampilkan periode waktu selama sebuah objek atau aktor melakukan aksi. Dalam *object lifeline*, *activation* berada diatas *lifeline* dalam bentuk kotak persegi panjang, bagian atas dari kotak merupakan inisialisasi waktu dimulainya suatu kegiatan dan yang dibawah merupakan akhir dari waktu.

d. Message

   Message adalah komunikasi antar objek yang membawa informasi dan hasil pada sebuah aksi. *Message* menyampaikan dari *lifeline* sebuah objek kepada *lifeline* yang lain, kecuali pada kasus sebuah *message* dari objek kepada objek itu sendiri, atau dengan kata lain *message* dimulai dan berakhir pada *lifeline* yang sama.

Collaboration Diagram

Diagram ini juga merupakan diagram interaksi. Diagram ini menekankan pada organisasi struktur dari objek-objek yang mengirim dan menerima pesan.

Diagram kolaborasi digunakan untuk

1. Pandangan dalam dari perilaku sistem, berfokus pada link-link di antara objek-objek
2. Ilustrasi dari suatu diagram use-case
3. Menyatakan objek-objek yang merealisasikan suatu layanan
4. Memeriksa jalur pengaksesan.

Kita membuat Diagram Kolaborasi dengan :

1. Lebih dulu tempatkan objek-objek yang berpartisipasi di interaksi sebagai simpul-simpul di graph.
2. Setelah itu, buat *link* yang menghubung objek-objek sebagai busur di *graph.*
3. Kemudian beri *link* ini dengan pesan yang dikirim dan diterima objek.

Statechart diagram

Kita dapat menggunakan *statechart diagram* untuk memodelkan perilaku dinamis satu kelas dari objek. *Statechart diagram* memperlihatkan urutan keadaan sesaat (*state*) yang dilalui sebuah objek. Kejadian yang menyebabkan transisi dari suatu *state* atau aktivitas kepada yang lainnya.

Statechart diagram khususnya digunakan untuk memodelkan tarap-tarap diskrit dari sebuah siklus hidup objek.

Component Diagram

Diagram komponen menunjukkan organisasi dan kebergantungan di antara sekumpulan komponen. Diagram ini memodelkan pandangan implementasi fisik dari sistem. Pada dasarnya, diagram ini merupakan diagram kelas yang berfokus pada komponen-komponen sistem.

Diagram komponen berisi :

1. Komponen
2. Antarmuka
3. Dependency, generalisasi, asosiasi, dan realisasi
4. Paket, untuk mengelompokan elemen-elemen model menjadi potongan-potongan besar.

Deployment Diagram

Merupakan suatu diagram yang menunjukkan bagian dari sistem yang bersifat fisik dan *Deployment/physical* diagram menggambarkan detail bagaimama komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server, atau piranti keras apa), bagaimana kemampuan jaringan lokasi tersebut, spesifikasi server, dan hal-hal yang bersifat fisikal.