BAB 2

LANDASAN TEORI

2.1 Pengertian Data

Berdasarkan pendapat Turban *et al*.(2003, p15), data adalah fakta-fakta mentah atau deskripsi dasar dari konsep-konsep, kejadian-kejadian, kegiatan-kegiatan, dan transaksi yang dapat ditangkap, direkam, disimpan, dan dikelompokkan, tetapi tidak terorganisasi dalam membawakan arti tertentu. Jadi data merupakan bentuk yang masih mentah yang belum dapat bercerita banyak sehingga masih perlu diolah lebih lanjut.

2.2 Pengertian Basis Data

Berdasarkan pendapat Connoly *et al*.(1997, p14), basis data adalah sebuah koleksi bersama data yang terkait secara logis, dan deskripsi dari data ini dirancang untuk memenuhi kebutuhan informasi dari sebuah organisasi.

2.3 Pengertian Data Spasial

Data spasial (Guting, 1994, p1**)** adalah :

1. Data yang berhubungan dengan ruang yang ditempati oleh benda.
2. Secara konseptual, titik, garis, persegi panjang, permukaan, volume, dan lain-lain.
3. Secara fisik, kota, sungai, jalan, menyatakan, tanaman cakupan, pegunungan, dan sebagainya.
4. Aplikasi termasuk pemantauan lingkungan, sistem informasi geografis, penelitian gempa bumi dll.

2.4 Basis Data Spasial

2.4.1 Pengertian Basis Data Spasial

Basis data spasial (Guting, 1994, p1**)** adalah basis data pada umumnya, tetapi menawarkan tipe data spasial (*spatial data type*) pada model data dan bahasa permintaan (*query*). Pada implementasinya, basis data telah mendukung tipe data spasial, menyediakan setidaknya pengindeksan spasial dan algoritma yang efisien untuk penggabungan spasial.

Basis data spasial dianggap menyediakan teknologi basis data yang mendasari Sistem Informasi Geografis (SIG) dan aplikasi lainnya. Aplikasi yang membawa perkembangan lebih lanjut untuk basis data spasial adalah Sistem Informasi Geografis (SIG).

Ada 2 yang perlu direpresentasikan dalam basis data ini yaitu objek di dalam ruang dan ruang itu sendiri.

1. Objek di dalam ruang, misalnya untuk menggambarkan kota, hutan, atau sungai.
2. Ruang itu sendiri, misalnya sebuah peta tematik yang menggambarkan penggunaan tanah atau bagian-bagian dari negara yang menjadi kabupaten.

Konsep untuk pemodelan kemudian dikembangkan menjadi 2, yaitu :

1. Objek tunggal

   Untuk memodelkan objek tunggal, dasar abstraksinya adalah titik, garis, dan bagian. Sebuah titik merepresentasikan sebuah lokasi objek hanya pada sebuah ruang, misalnya sebuah kota dapat dimodelkan sebagai sebuah titik dalam sebuah model yang menggambarkan wilayah geografis besar. Sebuah garis adalah

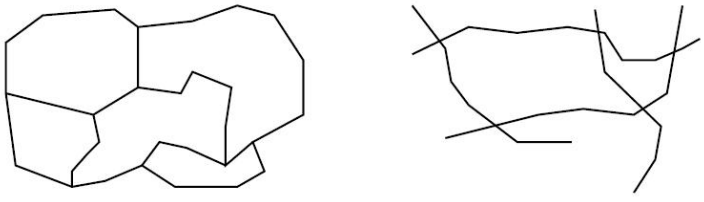
abstraksi dasar untuk menggambarkan koneksi di dalam ruang, misalnya jalan, sungai, kabel telepon atau jaringan listrik. Sebuah wilayah adalah abstraksi untuk sesuatu yang mempunyai luas pada area 2d, misalnya negara, danau, atau sebuah taman nasional. Sebuah wilayah boleh memiliki lubang dan boleh memiliki beberapa keping yang tidak tergabung.

Gambar 2.1 Objek tunggal

(Guting, 1994, p2)

2. Objek yang terhubung satu dengan yang lainnya.

   Sebuah partisi dapat dipandang sebagai satu set daerah objek yang dibutuhkan untuk dapat terbagi. Objek yang terhubung satu dengan yang lainnya dapat di gunakan untuk merepresentasikan peta tematik. Sebuah jaringan bisa di lihat seperti sebuah grafik yang di tanamkan dalam sebuah dataran, terdiri dari sebuah set titik objek, membentuk node, dan sebuah set objek garis yang mendeskripsikan geometri pada ujung-ujungnya. Jaringan ada dimana-mana dalam geografi, sebagai contoh jalan raya, sungai, alat transportasi publik, dan jaringan listrik.



Gambar 2.2 Objek yang terhubung satu dengan yang lainnya

(Guting, 1994, p3)

Jelas bahwa yang telah disebutkan di atas adalah abstraksi paling mendasar yang harus di dukung oleh basis data spasial (dalam kasus ini, digunakan untuk Sistem Informasi Geografis).

Keuntungan menggunakan basis data spasial :

1. Basis data spasial sangat handal dalam menangani data spasial.
2. Tidak memerlukan aplikasi SIG (contohnya *ArcView*).
3. Dapat digunakan oleh Sistem Informasi Geografis.
4. Memiliki indeks spasial, misalnya *R-Tree*.
5. Memiliki kemampuan penghitungan, misalnya jarak antar titik.
6. Memiliki fungsi spasial, misalnya fungsi irisan.
7. Memiliki fungsi pengamatan, misalnya dapat mengembalikan lokasi dari suatu titik.

2.4.2 Pengertian Data Geospasial (Geografis)

Data geospasial mempunyai komponen spasial dan komponen tematik. Secara konsep, data geografik bisa dibedakan menjadi 2 elemen yaitu *observation* / *entity* dan *attribute* / *variable*. Sistem Informasi Geografis dapat mengatur keduanya.

Observation mempunyai dua aspek dalam lokalisasinya yaitu lokalisasi berdasarkan sistem koordinat dan hubungan topologikal yang menunjuk ke *observation* lain. Contohnya : *The Department of Geomatics* berlokasi di posisi X dan Y tertentu, atau *The Department of Geomatics* terletak diantara *Grattan Street* dan *Old Engineering Building*. SIG berkemampuan mengatur kedua-duanya, sementara *computer assisted cartography* (komputer pembantu perpetaan) hanya dapat mengatur salah satunya.

Thematic component. Suatu variabel atau atribut bisa dipelajari mengenai aspek tematiknya (untuk ilmu statistika), aspek lokasinya (untuk analisis spasial) atau kedua-duanya (SIG). [http2]

2.5 Sistem Informasi Geografis (SIG)

Definisi Sistem Informasi Geografis yang dapat diterima secara luas diberikan oleh *National Centre of Geographic Information and Analysis*, yaitu sebuah sistem perangkat keras, perangkat lunak, dan prosedur untuk memfasilitasi pengelolaan, manipulasi, analisis, pemodelan, representasi, dan rujukan data geografis untuk memecahkan masalah-masalah yang kompleks tentang perencanaan dan pengelolaan sumber daya. [http2]

Peta terbagi menjadi 2 jenis, yaitu peta digital dan peta analog. Berikut diberikan perbedaan antara peta digital dan peta analog :

Tabel 2.1 Perbandingan perbedaan antara peta digital dan peta analog

Digital	Analog
Mudah diperbaharui	Harus membuat ulang semuanya
Mudah untuk dipindahkan (melalui internet)	Sulit untuk dipindahkan (melalui post)
Membutuhkan tempat penyimpanan yang relatif kecil (perangkat digital)	Membutuhkan tempat penyimpanan yang besar (perpustakaan peta)
Mudah untuk dijaga	Peta kertas mudah hancur seiring dengan waktu
Mudah untuk dianalisis	Sulit dan tidak akurat untuk dianalisis (misalnya jarak)

[http2]

Beberapa istilah di dalam Sistem informasi geografi berbasis vektor:

Vektor

Vektor adalah struktur data yang digunakan untuk *store* data spasial. Data vektor terdiri dari garis dan busur, didefinisikan oleh titik awal dan titik akhir, yang bertemu di simpul. Lokasi simpul tersebut dan struktur topologi biasanya disimpan secara eksplisit. Penyimpanan vektor hanya menyimpan titik-titik yang menentukan sebuah fitur.

Representasi data vektor

Pada model vektor, data geospasial direpresentasikan dalam bentuk koordinat. Dalam data vektor, unit dasar informasi spasial adalah titik-titik, garis (busur), dan poligon. Masing-masing unit ini terdiri dari rangkaian satu atau lebih titik koordinasi, sebagai contoh sebuah garis adalah kumpulan titik-titik yang terkait, dan poligon adalah kumpulan garis terkait.

1. Koordinat

   Sepasang angka mengungkapkan jarak horizontal sepanjang sumbu ortogonal, atau tiga pasang angka mengukur jarak horizontal dan vertikal, atau *n*-angka sepanjang sumbu *n*-mengungkapkan lokasi yang tepat dalam *n*-dimensi. koordinat umumnya mewakili lokasi di permukaan bumi relatif terhadap lokasi lain.

2. Titik

   Sebuah abstraksi dari obyek yang diwakili oleh satu koordinasi X dan satu koordinasi Y. Titik biasanya mewakili sebuah fitur geografis yang terlalu kecil untuk ditampilkan sebagai garis atau daerah, misalnya lokasi sebuah lokasi bangunan pada peta skala kecil.

3. Garis

   Satu set koordinat yang terurut yang mewakili bentuk geografis yang terlalu sempit untuk ditampilkan sebagai sebuah wilayah pada skala yang diberikan (jalan, sungai, dan lain-lain). Sebuah garis sinonim dengan busur.

4. Busur

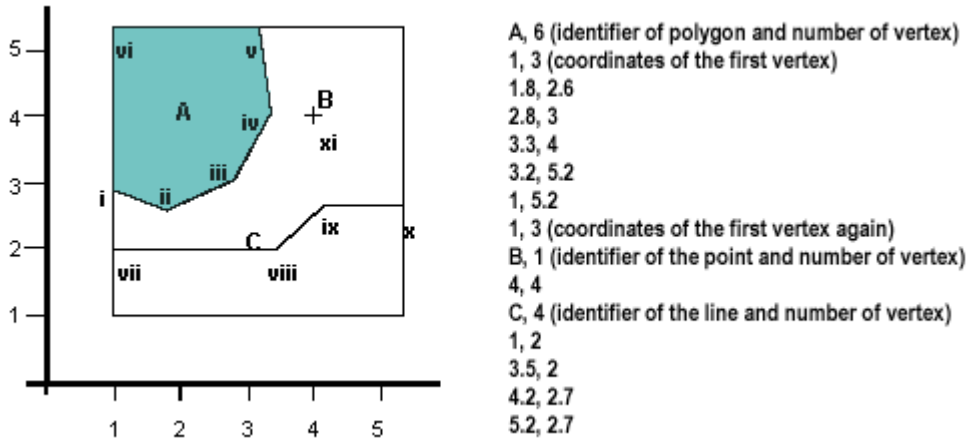
   Istilah yang memiliki arti yang sama dengan garis.

5. Poligon

   Sebuah objek yang digunakan untuk mewakili daerah. Suatu poligon didefinisikan oleh garis yang memiliki batas. Poligon memiliki atribut yang menjelaskan objek geografis yang mereka wakili.

Model-model Vektor

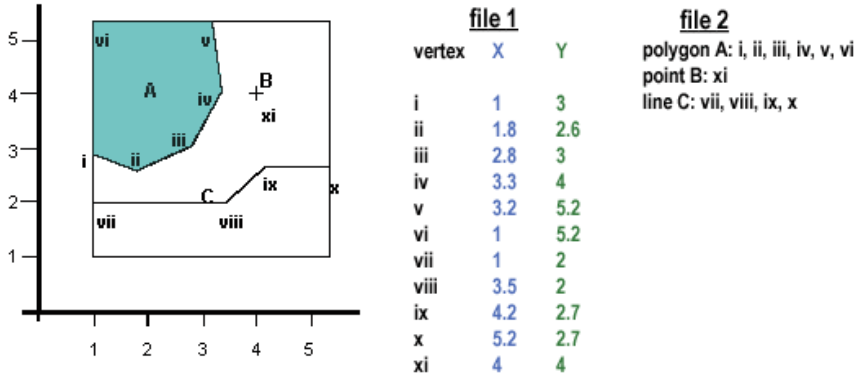
- *List of coordinates* "*spaghetti*"
  - Sederhana dan mudah dimengerti karena merupakan rangkaian koordinat yang menyatakan objek titik, garis, dan poligon.
  - Mudah di atur.
  - Tidak ada topologi.
  - Terjadi banyak duplikasi karena jika ada 2 poligon yang berbatasan, maka garis-garis yang menjadi batas-batas bersama di antara poligon-poligon yang bersebelahan akan disimpan dua kali, yang pertama untuk poligon pertama, dan yang kedua untuk poligon yang terletak berbatasan.



Gambar 2.3 *List of coordinates* "spaghetti"

[http2]

- *Vertex Dictionary*
  - Tidak ada duplikasi.
  - Tidak ada topologi.



file 1

vertex	X	Y
i	1	3
ii	1.8	2.6
iii	2.8	3
iv	3.3	4
v	3.2	5.2
vi	1	5.2
vii	1	2
viii	3.5	2
ix	4.2	2.7
x	5.2	2.7
xi	4	4

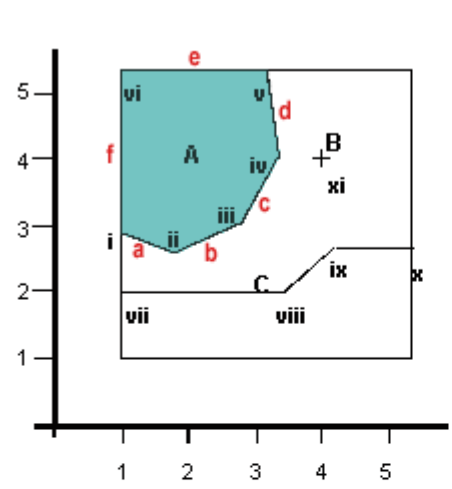
file 2

polygon A: i, ii, iii, iv, v, vi
point B: xi
line C: vii, viii, ix, x

Gambar 2.4 *Vertex dictionary*

[http2]

- *Dual Independent Map Encoding*
  - Dikembangkan oleh *US Bureau of the Census*.
  - Simpul diidentifikasikan dengan kode-kode.
  - Memberikan kode terarah dalam bentuk ‘dari simpul’ dan ‘ke simpul’



file 1

vertex	X	Y
i	1	3
ii	1.8	2.6
iii	2.8	3
iv	3.3	4
v	3.2	5.2
vi	1	5.2
vii	1	2
viii	3.5	2
ix	4.2	2.7
x	5.2	2.7
xi	4	4

segment	right polygon	left polygon	from vertex	to vertex
a	external	A	i	ii
b	external	A	ii	iii
c	external	A	iii	iv
d	external	A	iv	v
e	external	A	v	vi
f	external	A	vi	i
g	external	external	vii	viii
h	external	external	viii	ix
i	external	external	ix	x

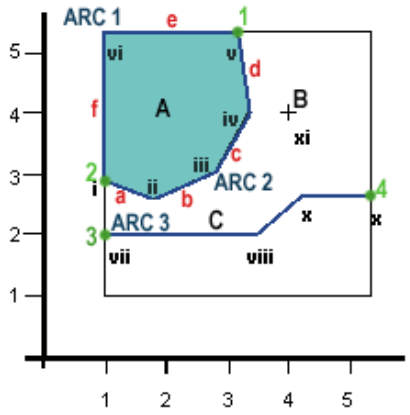
polygon	segments
A	a, b, c, d, e, f

Gambar 2.5 Bentuk *Dual Independent Map Encoding* (DIME)

[http2]

- *Arc* / *Node*

  Secara kartografis (Seni, ilmu, dan teknik dalam membuat peta) penggambaran garis yang membatasi dua poligon yang bersebelahan tidak akan dilakukan dua kali. Pada struktur *arc-node*, *node* direferensikan membentuk segmen garis tertentu dimana segmen-segmen garis membentuk poligon. Kemudian segmen garis diawali dan diakhiri oleh masing-masing satu *node*. Selain itu, diantara dua *node* ada titik-titik verteks.



Gambar 2.6 Model digital

[http2]

Tabel 2.2 ***Coordinates of nodes and vertex for all the arcs***

ARC	*F_node*	*Vertex*	*T_node*
1	3.2, 5.2	1, 5.2	1,3
2	1,3	1.8,2.6 2.8,3 3.3,4	3.2, 5.2
3	1,2	3.5,2 4.2,2.7	5.2,2.7

[http2]

Tabel 2.3 ***Arcs topology***

ARC	*F_node*	*T_node*	*R_poly*	*L_poly*
1	1	2	External	A
2	2	1	A	External
3	3	4	External	External

[http2]

Tabel 2.4 ***Polygons topology***

Polygon	*Arcs*
A	1, 2

[http2]

Tabel 2.5 ***Nodes topology***

Node	***Arcs***
1	1,2
2	1,2
3	3
4	4
5	5

[http2]

Raster

Raster adalah metode untuk penyimpanan, pengolahan, dan menampilkan data spasial. Setiap daerah terbagi menjadi baris dan kolom, yang membentuk struktur *grid* biasa. Setiap sel dalam matriks ini berisi koordinat lokasi serta nilai atribut. Lokasi spasial dari setiap sel secara implisit terkandung dalam susunan dari matriks, tidak seperti struktur vektor yang menyimpan topologi secara eksplisit. Bagaimanapun, struktur raster tidak dapat mengidentifikasi batas-batas bidang-bidang seperti poligon.

Data raster adalah sebuah abstraksi dari dunia nyata di mana data spasial dinyatakan sebagai matriks sel atau pixel, dengan posisi spasial yang tersirat dalam susunan pixel. Dengan model data raster, data spasial tidak sinambung, tapi terbagi menjadi unit yang berlainan. Hal ini membuat data raster sangat cocok untuk jenis operasi spasial tertentu, misalnya lapisan atau perhitungan area.

Perbandingan Vektor dan Raster

Tabel 2.6 Perbandingan vektor dan raster

	raster	**vector**
Presisi dalam grafik	×	√
Kartografi tradisional	×	√
Volume data	×	√
Topologi	×	√

	raster	vector
Perhitungan	√	×
Pembaruan	√	×
Ruang sinambung	√	×
Integrasi	√	×
Terputus-putus	×	√

[http2]

2.6 *Oracle* Spasial

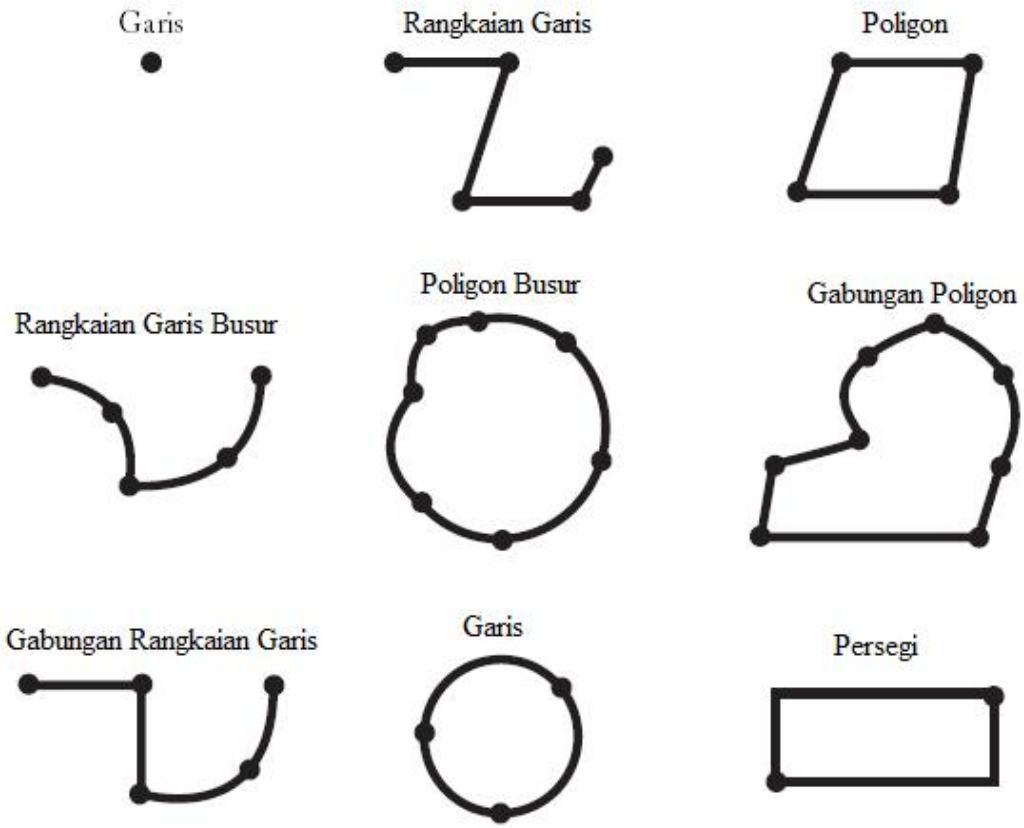
2.6.1 Geometri

Geometri adalah komponen di dalam *Oracle* Spasial. Sebuah geometri (Chuck, 2006, p3) adalah urutan teratur simpul yang terhubung dengan segmen-segmen garis lurus atau busur melingkar. *Oracle* spasial mendukung beberapa tipe primitif antara lain:

1. Titik dan kumpulan titik-titik (*Points and point clusters*)
2. Garis (*Line strings*)
3. Titik yang membentuk poligon (*n-point polygons*)
4. Garis busur atau lengkungan (*Arc line strings*)
5. Poligon busur atau lengkungan (*Arc polygons*)
6. Gabungan poligon (*Compound polygons*)
7. Gabungan garis (*Compound line strings*)
8. Lingkaran (*Circles*)
9. Persegi yang dioptimalkan (*Optimized rectangles*)

Titik Dua dimensi adalah elemen yang terdiri dari dua koordinat, X dan Y, sering berhubungan dengan bujur dan lintang. Rangkaian Garis terdiri dari satu atau lebih pasang titik yang mendefinisikan segmen garis. Poligon terdiri dari garis yang terhubung yang membentuk cincin tertutup, dan memiliki luas. Sebagai contoh, sebuah titik

mungkin mewakili suatu lokasi bangunan, satu garis mungkin mewakili suatu jalan atau jalur penerbangan, dan poligon bisa mewakili sebuah negara, kota, wilayah kabupaten, atau kota blok.



Gambar 2.7 Geometri

(Chuck, 2006, p4)

2.6.2 Model Data

Data model spasial (Chuck, 2006, p4) adalah struktur hirarkis yang terdiri dari unsur, geometri, dan lapisan. Lapisan terdiri dari geometri, yang pada gilirannya terdiri dari unsur-unsur. Penjelasan untuk setiap unsur :

- Unsur

  Unsur adalah blok bangunan dasar dari suatu geometri. Tipe unsur yang di dukung antara lain titik, rangkaian garis, dan poligon. Setiap koordinat pada unsur di simpan dalam sebuah x, dan y. data titik terdiri dari 1 koordinat. Data garis terdiri dari 2 koordinat.

- Geometri

  Geometri dimodelkan sebagai sebuah himpunan elemen-elemen primitif. Misalnya beberapa poligon dapat merepresentasikan sebuah kepulauan.

- Lapisan

  Lapisan adalah kumpulan dari geometri.

Sistem koordinat

Sebuah sistem koordinat merupakan suatu cara untuk menentukan koordinat lokasi dan untuk membangun hubungan dengan kumpulan koordinat. Ini memungkinkan penafsiran kumpulan koordinat sebagai representasi dari sebuah posisi dalam ruang dunia nyata. Setiap data spasial memiliki koordinat yang terkait dengan dunia nyata. Sistem koordinat dapat saja mengacu pada koordinat geografis sebenarnya (berkaitan dengan representasi tertentu dari bumi) atau tidak mengacu pada koordinat geografis sebenarnya (yaitu, Cartesian, dan tidak berkaitan dengan representasi tertentu dari bumi). Jika sistem koordinat mengacu pada koordinat geografis sebenarnya, maka koordinat memiliki unit pengukuran yang standar (seperti meter) yang terkait dengannya, namun Spasial dapat secara otomatis memberikan hasil dalam satuan yang ditentukan lain (seperti mil).

Data spasial dapat berhubungan dengan *Cartesian,* geodetik (Geografis), proyeksi, atau sistem koordinat lokal.

1. Koordinat *Cartesian* adalah koordinat yang mengukur posisi dari sebuah titik asal yang ditetapkan di sepanjang sumbu yang tegak lurus di representasikan pada dua dimensi atau tiga dimensi.
2. Koordinat *Geodetic* (kadang-kadang disebut koordinat geografis) adalah sudut koordinat (bujur dan lintang), terkait erat dengan koordinat polar bola, dan didefinisikan berhubungan dengan Bumi.
3. Proyeksi koordinat adalah koordinat Cartesian yang dihasilkan dari pemetaan matematika dari sebuah titik pada permukaan bumi ke pesawat.
4. Koordinat lokal adalah koordinat *cartesian* yang bukan merupakan koordinat bumi.

Tolerance / Toleransi

Tolerance digunakan untuk menghubungkan tingkat presisi dengan data spasial. *Tolerance* menganggap bahwa 2 titik yang terpisah masih dianggap sama. Nilai dari *tolerance* harus sebuah nilai positif yang lebih besar dari nol. Pentingnya nilai tergantung pada apakah data spasial dikaitkan dengan sistem koordinat geodetik atau tidak.

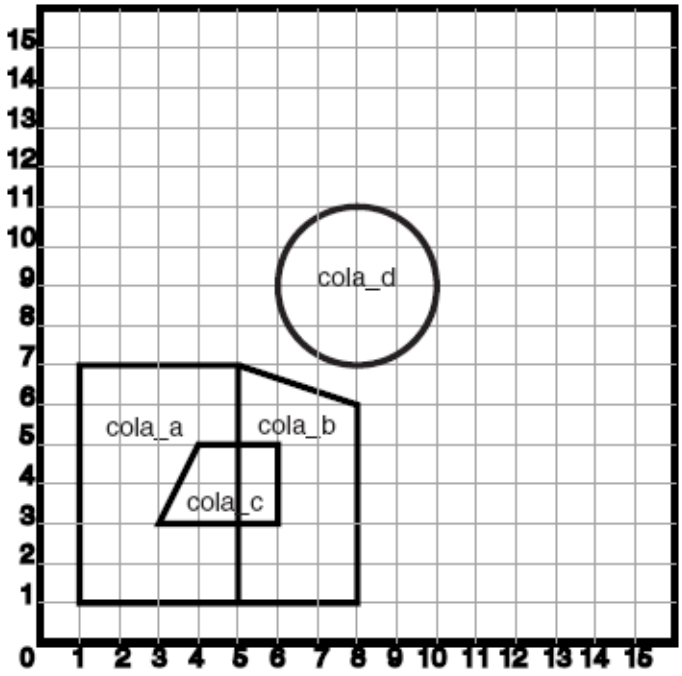
1. Untuk data geodetik (seperti data yang diidentifikasi oleh koordinat bujur dan lintang), nilai toleransi adalah meter. Sebagai contoh, sebuah nilai 100 dari tolerance menunjukkan toleransi bernilai 100 meter. Nilai toleransi untuk data geodetik tidak boleh lebih kecil dari 0,05 (5 cm), dan dalam kebanyakan kasus

bahkan nilainya harus lebih besar dari itu. Spasial menggunakan 0,05 sebagai nilai *tolerance* untuk data geodetik jika nilai yang ditetapkan lebih kecil dari 0,05.

2. Untuk data yang bukan geodetik, nilai *tolerance* adalah sebuah nilai yang di ambil dari sistem koordinat dari data tersebut. Misalnya, nilai pengukurannya adalah mil, nilai *tolerance* 0.005 mengindikasikan toleransi 0.005 (1/200) mil (mendekati 26 kaki) dan nilai *tolerence 2,* mengindikasikan 2 mil.

Semakin kecil nilai *tolerance*, semakin presisi data tersebut. Nilai *tolerance* ditetapkan dalam 2 kasus :

1. Jika ada sebuah fungsi yang dapat menerima parameter *tolerance* secara opsional dan parameternya *null* atau tidak diberikan, maka nilai *SDO_TOLERANCE* akan digunakan. Jika menggunakan data non-geodetik dari contoh di bawah ini :



Gambar 2.8 Ilustrasi Gambar Toleransi

(Chuck, 2006)

Jarak sebenarnya dari cola_b dan cola_d adalah 0.846049894 jika query menggunakan fungsi *SDO_GEOM.SDO_DISTANCE* untuk mengembalikan nilai jarak cola_b dan cola_d dan tidak memberikan nilai *tolerance* pada parameternya, nilainya tergantung pada nilai *SDO_TOLERANCE*. Misalnya, Jika nilai *SDO_TOLERANCE* adalah 0.005, maka jaraknya adalah 0.846049894. Jika nilai *SDO_TOLERANCE* adalah 0.5, maka jaraknya adalah 0. Nilai 0 terjadi karena Spasial pertama akan membuat *buffer* imajiner dengan nilai *tolerance* 0.5 pada setiap objek geometri yang akan dihitung, dan *buffer* disekitar cola_b dan cola_d saling melewati.

Oleh karena itu dapat mengambil salah satu dari dua pendekatan dalam memilih sebuah nilai *SDO_TOLERANCE* pada *layer* (lapisan):

a. Nilai dapat mencerminkan tingkat presisi yang dikehendaki dalam *query* untuk jarak antara objek. Sebagai contoh, jika dua geometri non-geodetik (geodesi) memiliki 0,8 unit dan ingin dianggap terpisahkan, tentukan nilai *SDO_TOLERANCE* kecil misalnya 0,05 atau lebih kecil.
b. Nilai dapat mencerminkan nilai-nilai ketepatan yang berkaitan dengan geometri dalam lapisan. Sebagai contoh, jika semua geometri dalam lapisan non-geodesi didefinisikan menggunakan bilangan bulat dan jika dua benda berjarak 0,8 unit terpisah dan dianggap sebagai tidak terpisah, sebuah *SDO_TOLERANCE* nilai 0,5 adalah tepat. Untuk memiliki presisi yang lebih besar dalam *query* apapun, harus mengabaikan nilai *default* dengan menentukan parameter *tolerance*.

2. Banyak fungsi spasial menerima parameter nilai *tolerance*, jika nilai *tolerance*-nya diberikan, maka akan menggantikan nilai *tolerance* layer tersebut. Jika jarak antara dua titik kurang dari atau sama dengan nilai dari *tolerance*, spasial akan

menganggap bahwa kedua titik tersebut adalah sama. Jadi *tolerance* adalah suatu refleksi seberapa akurat suatu data spasial.

Sebagai contoh:

Misalkan ingin diketahui restoran apa yang berjarak 5 km dari sebuah rumah. Asumsikan bahwa Restoran A berjarak 5.1 km dari rumah tersebut. Jika data spasial tersebut menggunakan sistem koordinat geodetik, dan di-*query* dengan memberikan nilai parameter *tolerance* 100 (5.1 km) atau 500 (5.5 km), maka restoran A akan termasuk karena jaraknya 5.1 km. jika nilai yang diberika tersebut lebih kecil dari 100, misalnya 50, maka Restoran A tidak akan termasuk.

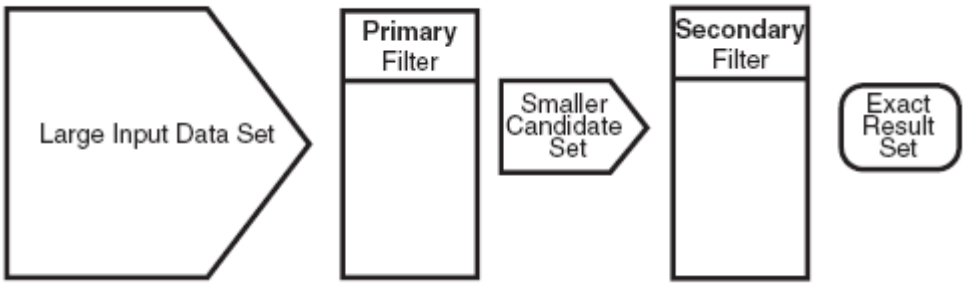
2.6.3 Query Model

Spasial menggunakan dua tingkat model *query* (Chuck, 2006, p8) untuk menyelesaikan *query* spasial dan penggabungan spasial. Istilah ini digunakan untuk mengindikasikan dua operasi yang berbeda yang dijalankan untuk menyelesaikan *query.* Keluaran dari dua operasi kombinasi ini menghasilkan suatu paket hasil yang tepat.

Dua operasi yang dimaksud tersebut adalah operasi penyaringan primer dan sekunder:

1. Penyaringan primer memungkinkan pemilihan kandidat *record* yang cepat untuk disampaikan kepada penyaringan sekunder. Penyaringan primer membandingkan pendekatan geometri untuk mengurangi kompleksitas perhitungan dan dianggap sebagai penyaringan berbiaya rendah. Karena penyaringan primer membandingkan pendekatan geometris, dan mengembalikan set hasil yang tepat.
2. Penyaringan sekunder menggunakan komputasi yang tepat untuk geometri yang dihasilkan dari penyaringan primer. Penyaringan sekunder menghasilkan jawaban

yang akurat terhadap *query* spasial. Penyaringan sekunder adalah penyaringan berbiaya tinggi, tetapi hanya diterapkan pada hasil penyaringan primer, bukan pada seluruh kumpulan data.



Gambar 2.9 Ilustrasi Penyaringan data spasial

(Chuck, 2006)

Seperti yang ditunjukkan oleh gambar, penyaringan primer beroperasi pada masukan data dalam jumlah besar dan menghasilkan set kandidat yang lebih kecil. Penyaringan sekunder beroperasi pada set kandidat yang lebih kecil dan menghasilkan set hasil akurat.

Spasial menggunakan indeks spasial untuk mengimplementasikan penyaringan primer. Spasial tidak membutuhkan penggunaan kedua penyaringan primer dan sekunder. Dalam beberapa kasus, hanya menggunakan penyaringan primer saja sudah cukup. Sebagai contohnya, fitur pembesaran pada aplikasi pemetaan meminta data yang berhubungan dengan persegi yang merepresentasikan batas yang terlihat. Penyaringan primer dengan cepat mengembalikan set *query.* Aplikasi mapping kemudian dapat menampilkan wilayah target.

Tujuan penyaringan primer adalah dengan cepat menciptakan subset data dan mengurangi beban pengolahan penyaringan sekunder. Karena itu, penyaringan primer haruslah seefisien mungkin, hal ini ditentukan oleh karekteristik data spasial.

2.6.4 Pengindeksan Data Spasial

Kemampuan indeks spasial di dalam mesin basis data *Oracle* (Chuck, 2006, p9) adalah fitur kunci dari produk Spasial. Sebuah indeks spasial, seperi indeks yang lainnya, penyediakan mekanisme untuk membatasi pencarian, tetapi dalam kasus ini mekanismenya didasarkan pada kriteria spasial seperti titik potong (*intersection*) dan pemuatan (*containment*). Indeks spasial diperlukan untuk :

1. Menemukan objek dengan data ruang yang terindeks yang berinteraksi dengan suatu nilai yang telah diberikan (*window query*).
2. Menemukan sepasang objek dari 2 data ruang yang terindeks yang berinteraksi secara spasial antara satu dengan yang lain (*spasial join*).

Sebuah indeks spasial dianggap sebagai indeks logikal. Entri pada indeks spasial tergantung pada lokasi geometri pada ruang koordinat, tetapi nilai indeks berada pada wilayah yang berbeda. Entri indeks mungkin tersusun menggunakan domain yang tersusun secara linear, dan koordinat untuk geometri mungkin sepasang nilai bulat, bilangan spasial, atau angka presisi ganda.

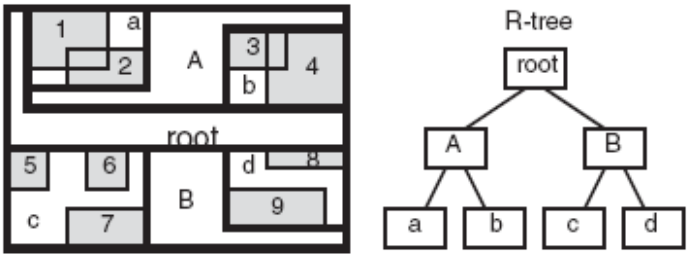
Oracle Spasial menggunakan pengindeksan *R-Tree* secara default, pengindeksan *Quadtree*, atau keduanya. Penggunaan untuk *R-Tree* lebih disarankan. Peningkatan performa yang signifikan telah dibuat untuk pengindeksan *R-Tree*.

Pengindeksan *R-Tree*

Sebuah indeks *R-Tree* mendekati setiap geometri dengan sebuah persegi tunggal yang secara minimal melampirkan geometri (disebut *minumum bounding rectangle*, MBR).

Gambar 2.10 MBR melampirkan geometri

(Chuck, 2006)



Gambar 2.11 Indeks hirarki *R-Tree* dalam MBR

(Chuck, 2006)

- 1 sampai 9 geometri ada di permukaan.
- a, b, c, d adalah *leaf node* dari indeks *R-Tree*, dan memiliki *minimum bounding rectangle* untuk geometri, beserta dengan penunjuk untuk geometri. Sebagai contoh, a berisi geometri MBR 1 dan 2, b berisi geometri MBR 3 dan 4, dan seterusnya.
- A berisi MBR a dan b, B berisi MBR c dan d.
- *Root* berisi MBR A dan B.

Indeks dari *R-Tree* disimpan di dalam tabel indeks spasial (*SDO_INDEX_TABLE*). Indeks *R-Tree* juga mengurus objek *sequence* (*SDO_RTREE_SEQ_NAME*) untuk memastikan pembaharuan yang simultan oleh pengguna yang berbarengan bisa dibuat ke dalam indeks.

Kualitas *R-Tree*

Jumlah operasi pemasukan dan penghapusan berefek pada indeks *R-Tree* mungkin menurunkan kualitas struktur dari *R-Tree*, yang akan berefek negatif pada performa *query*. Ketika itu terjadi, membangun kembali indek akan membantu. Untuk mengecek kualitas indek bisa digunakan *SDO_TUNE.QUALITY_DEGREDATION*, jika fungsi tersebut mengembalikan nilai lebih besar dari 2, maka pertimbangkan untuk membangun kembali indeks. *R-Tree* adalah struktur pohon hirarkis dengan *node* pada ketinggian yang berbeda pada pohon.

2.7 *Database Management System* (*MYSQL, Oracle,* dll)

Berdasarkan pendapat Connolly (1997, p16), *Database Management System* adalah Sebuah sistem piranti lunak yang memungkinkan pengguna untuk mendefinisikan, membuat, memelihara, dan mengontrol akses ke basis data.

2.8 Internet

2.8.1 Pengertian Internet

Berdasarkan pendapat Sidharta (1996, pxiii), internet adalah sumber daya informasi yang menjangkau seluruh dunia.

2.8.2 Sejarah Internet

Asal-usul internet Sidharta (1996, pxiii) berasal dari jaringan komputer yang dibentuk pada tahu 1970-an. Jaringan komputer tersebut disebut ARPANET (*Advanced Research Project Agency Network*), yaitu jaringan komputer yang dibentuk oleh departemen pertahanan Amerika Serikat. Selanjutnya, jaringan komputer tersebut

diperbaharui dan dikembangkan, dan sekarang penerusnya menjadi tulang punggung global untuk sumber daya informasi yang disebut dengan internet.

2.9 Jaringan Client Server

Dalam buku yang ditulis oleh Wahana Komputer (Menjadi Administrator Jaringan Komputer, 2005, p57), berikut adalah pengertian *client* dan pengertian *server.*

Client adalah perangkat keras yang digunakan untuk menerima layanan dan menerima dari *server* dalam suatu jaringan. *Client* umumnya memiliki sistem operasi dengan antarmuka grafis dan memiliki program-program penunjang layanan di jaringan.

Server adalah suatu komputer yang memberikan suatu layanan bagi komputer lain dalam jaringan. Layanan dari server bermacam-macam, mulai dari sharing, web hingga FTP server. Server dapat dibuat dari spesifikasi komputer yang rendah hingga tertinggi seperti *Mainframe* atau *Super Computer*.

client-server merupakan sebuah paradigma dalam teknologi informasi yang merujuk kepada cara untuk mendistribusikan aplikasi ke dalam dua pihak: pihak klien dan pihak server.

2.10 Interaksi Manusia dan Komputer

2.10.1 Pengertian Interaksi Manusia dan Komputer

Berdasarkan pendapat Shneiderman (1998), Interaksi manusia dan komputer adalah disiplin ilmu yang berhubungan dengan perancangan, evaluasi, dan implementasi sistem komputer interaktif untuk digunakan oleh manusia, serta studi fenomena-fenomena besar yang berhubungan dengannya.

2.10.2 Pengertian Antarmuka Pengguna (*User Interface*)

Berdasarkan pendapat Shneiderman (1998), Antarmuka pengguna merupakan bentuk tampilan grafis yang berhubungan langsung dengan pengguna (*user*). Antarmuka pengguna berfungsi untuk menghubungkan antara pengguna dengan sistem operasi, sehingga komputer tersebut bisa digunakan.

a. Delapan Aturan Emas Perancangan UI (*User Interface*)

1. Konsistensi

   Konsistensi dilakukan pada urutan tindakan, perintah, dan istilah yang digunakan pada prompt, menu, serta layar bantuan.

2. Memungkinkan pengguna untuk menggunakan *shortcut*

   Ada kebutuhan dari pengguna yang sudah ahli untuk meningkatkan kecepatan interaksi, sehingga diperlukan singkatan, tombol fungsi, perintah tersembunyi, dan fasilitas makro.

3. Memberikan umpan balik yang informatif

   Untuk setiap tindakan operator, sebaiknya disertakan suatu sistem umpan balik. Untuk tindakan yang sering dilakukan dan tidak terlalu penting, dapat diberikan umpan balik yang sederhana. Tetapi ketika tindakan merupakan hal yang penting, maka umpan balik sebaiknya lebih substansial. Misalnya muncul suatu suara ketika salah menekan tombol pada waktu input data atau muncul pesan kesalahannya.

4. Merancang dialog untuk menghasilkan suatu penutupan

   Urutan tindakan sebaiknya diorganisir dalam suatu kelompok dengan bagian awal, tengah, dan akhir. Umpan balik yang informatif akan meberikan indikasi

bahwa cara yang dilakukan sudah benar dan dapat mempersiapkan kelompok tindakan berikutnya.

5. Memberikan penanganan kesalahan yang sederhana

   Sedapat mungkin sistem dirancang sehingga pengguna tidak dapat melakukan kesalahan fatal. Jika kesalahan terjadi, sistem dapat mendeteksi kesalahan dengan cepat dan memberikan mekanisme yang sedehana dan mudah dipahami untuk penanganan kesalahan.

6. Mudah kembali ke tindakan sebelumnya

   Hal ini dapat mengurangi kekhawatiran pengguna karena pengguna mengetahui kesalahan yang dilakukan dapat dibatalkan, sehingga pengguna tidak takut untuk mengekplorasi pilihan-pilihan lain yang belum biasa digunakan.

7. Mendukung tempat pengendali internal (*internal locus of control*)

   Pengguna ingin menjadi pengontrol sistem dan sistem akan merespon tindakan yang dilakukan pengguna daripada pengguna merasa bahwa sistem mengontrol pengguna. Sebaiknya sistem dirancang sedemikan rupa sehingga pengguna menjadi inisiator daripada responden.

8. Mengurangi beban ingatan jangka pendek

   Keterbatasan ingatan manusia membutuhkan tampilan yang sederhana atau banyak tampilan halaman yang sebaiknya disatukan, serta diberikan cukup waktu pelatihan untuk kode, *mnemonic*, dan urutan tindakan.

b. Manfaat Warna

1. Menyejukkan atau merangsang mata.
2. Memberi aksen pada tampilan yang tidak menarik.
3. Memungkinkan pembedaan yang halus pada tampilan yang kompleks.
4. Menekankan organisasi logis informasi.
5. Menarik perhatian kepada peringatan.
6. Menimbulkan reaksi emosional yang kuat berupa sukacita, kegembiraan, ketakutan, atau kemarahan.

c. Pedoman Penggunaan warna

1. Gunakan warna secara konservatif.
2. Batasi jumlah warna.
3. Kenali kekuatan warna sebagai teknik pengkodean untuk mempercepat atau memperlambat tugas.
4. Pastikan bahwa *color coding* mendukung tugas.
5. Tampilkan *color coding* dengan usaha pemakai yang minimal.
6. Tempatkan *color coding* di bawah kendali pemakai.
7. Rancang untuk monokrom dulu.
8. Gunakan warna untuk membantu pemformatan.
9. Gunakan *color coding* yang konsisten.
10. Perhatikan ekspektasi umum tentang kode warna.
11. Gunakan perubahan warna untuk menunjukkan perubahan status.
12. Gunakan warna pada tampilan grafis untuk kerapatan informasi yang lebih tinggi.

d. Kesalahan Utama *Design Web* (Nielsen *et al*., 1996)

- Penggunaan *frame*.
- Penggunaan teknologi baru dengan serampangan.
- Gerakan teks dan animasi yang berjalan terus.
- URL yang kompleks.
- Halaman yatim.
- Halaman yang terlalu panjang gulungannya. Isi terpenting dan navigasi harus tampak di bagian atas.
- Kurangnya dukungan navigasi.
- Warna link yang tidak standar.
- Informasi yang sudah lama.
- Waktu *download* yang terlalu lama. Pemakai kehilangan minat dalam 10-15 detik.

2.11 Pengertian Sistem

Menurut O'Brien (2005, p5) sistem merupakan sekelompok dari elemen-elemen yang saling berhubungan atau saling mempengaruhi yang membentuk kesatuan.

2.12 Pengertian informasi

Menurut Whitten *et al.*(2004, p23) informasi adalah data yang telah diproses atau diorganisasi ulang menjadi bentuk yang berarti. Informasi dibentuk dari kombinasi data yang diharapkan memiliki arti ke penerima.

2.13 Pengertian sistem informasi

Menurut Whitten *et al.*(2004, p10) sistem informasi adalah pengaturan orang, data, proses, dan teknologi informasi yang berinteraksi untuk mengumpulkan, memproses, menyimpan, dan menyediakan sebagai *output* informasi yang diperlukan untuk mendukung sebuah organisasi.

2.14 *Unified Modeling Language* (UML)

2.14.1 Pengertian UML

Unified Modeling Language (Simon Bennett, 2005, p5) adalah bahasa visual yang memberikan jalan bagi orang-orang yang menganalisis dan mendesain sistem berorientasi objek untuk memvisualisasikan, membangun, dan mendokumentasikan artifak dari sistem perangkat lunak dan model organisasi bisnis yang menggunakan sistem itu.

2.14.2 Sejarah UML

Menurut Booch (1998, pxviii) Bahasa pemodelan berorientasi objek muncul pada pertengahan 1970-an dan pada akhir 1980-an sebagai suatu metodologi, berhadapan dengan genre baru bahasa pemrograman berorientasi objek dan semakin bertambahnya aplikasi yang rumit, maka dimulailah percobaan dengan pendekatan analisis dan rancangan alternatif. Jumlah metode berorientasi objek meningkat dari jumlah yang kurang dari 10 hingga jumlah yang lebih besar dari 50 selama periode 1989 dan 1994. Banyak pengguna metode ini mempunyai masalah menemukan bahasa pemodelan yang mereka butuhkan sepenuhnya. Belajar dari pengalaman, generasi baru dari metode ini mulai muncul, dan beberapa metode yang menonjol muncul, terutama Booch,

Jacobson's OOSE (*Object Oriented Software Engenering*), and Rumbaugh's OMT (*Object Modeling Technique*). Setiap metode memiliki kelemahan dan kekuatan masing-masing. Metode Booch sangan ekspresif selama fase desain dan konstruksi proyek, OOSE menyediakan dukungan yang bagus untuk *use cases* dalam analisis, dan design level tinggi, dan OMT-2 sangat berguna untuk analisis dan sistem informasi data-intensif.

Banyak ide kritis muncul mulai dari pertengahan 1990, ketika Grady Booch (*Rational Software Corporation*), Ivar Jacobson (*Objectory*), dan James Rumbaugh (*General Electric*) mulai saling mengadopsi ide dari metode yang lain, yang mana pada akhirnya menjadi metode berorientasi objek yang diakui di seluruh dunia. Pada tahun 1995 dirilislah UML (versi 0.8). Standar spesifikasi UML dijadikan standar defacto oleh OMG (Object Management Group) pada tahun 1997. Sejak itulah UML menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.

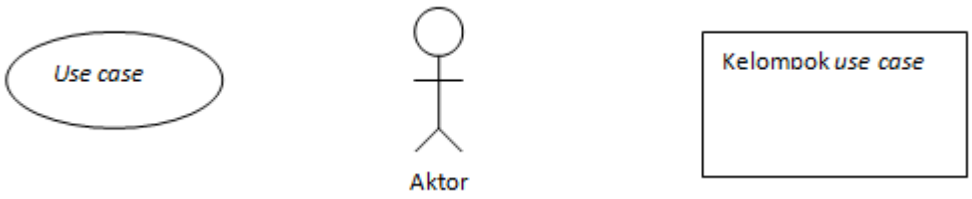
2.14.3 Tujuan UML

Menurut Booch (1998, pxviii) Organisasi piranti lunak yang berhasil adalah organisasi yang secara konsisten membangun kualitas piranti lunak yang memenuhi kebutuhan dari pengguna. Untuk membangun piranti lunak dengan cepat, efisien, dan efektif dibutuhkan orang, alat, dan fokus yang tepat. Modeling adalah bagian sentral dari semua kegiatan yang mengarah pada penyebaran piranti lunak yang baik. UML digunakan untuk memvisualisasikan dan mengontrol arsitektur sistem. Model dibangun untuk pemahaman yang lebih baik yang sedang dibangun, dan untuk mencari kesederhanaan dan penggunaan kembali. UML dapat dimengerti secara umum, karena merupakan bahasa pemodelan yang telah dikenal secara luas.

2.14.4 Beberapa bagian dari UML

2.14.4.1 Use Case Diagram

Menurut Mathiassen *at el.*(2000, p343) *Use case* adalah pola interaksi antara sistem dan aktor pada *application domain*. Aktor adalah sebuah abstraksi pengguna atau sistem lain yang berinteraksi dengan sistem. Hasil dari aktivitas penggunaan adalah *use case* dan aktor. Sebuah kumpulan *use case* yang lengkap adalah semua penggunaan dari sistem. Notasi yang digunakan adalah sebagai berikut.



Gambar 2.12 *Use Case Diagram*

(Mathiassen, 2000)

2.14.4.2 Activity Diagram

Berdasarkan pendapat Bennet (2002, p253) Diagram aktifitas menyediakan banyak deskripsi yang dibutuhkan oleh sistem dengan menyediakan langkah selanjutnya dalam menganalisis suatu sistem mengikuti diagram *use cases*. Diagram aktifitas sangat membantu, khususnya pada *use cases*, karena dapat memberikan keadaan awal dan akhir kepada pembaca. Beberapa komponen yang digunakan dalam *Activity Diagram* :

1. *Initial State*

   Initial State merupakan awal dari suatu aktfitas. Setiap *activity Diagram* hanya boleh mempunyai 1 *initial* state. *Initial state* digambarkan dengan sebuah titik hitam solid.

Gambar 2.13 *Initial state*

(Bennet, 2002)

2. *Final State*

   Final State merupakan akhir dari suatu aktifitas. Setiap *activity Diagram* boleh mempunyai lebih dari 1 *final state*. *Final state* digambarkan dengan sebuah titik hitam solid serta lingkaran di pinggirnya.

Gambar 2.14 *Final state*

(Bennet, 2002)

3. *Actions*

   Actions adalah sebuah unit yang harus dilakukan. biasanya diidentifikasikan dengan sebuah kata atau frase yang menunjukkan keadaan sistem saat ini.

Gambar 2.15 *Actions*

(Bennet, 2002)

4. *Control Flow*

   Contol Flow digunakan untuk menunjukkan aliran kontrol dari *state* satu ke yang lain. *Control Flow* bisa menunjukkan aliran dari *state* ke *activity*, diantara *activities*, dan diantara *states*

Gambar 2.16 *Control Flow*

(Bennet, 2002)

5. *Decision*

   Decision digunakan untuk menggambarkan kontrol aliran yang bersifat kondisional. *Decision* di gambarkan dengan opsi-opsi yang ditulis di kedua sisi dari panah yang muncul dari objek *decision* dan diberi kurung kotak.

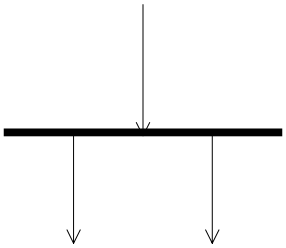
Gambar 2.17 *Decision*

(Bennet, 2002)

6. *Fork Nodes* dan *Join Nodes*

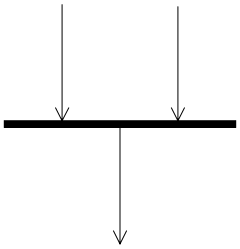
   Terkadang masuk akal untuk membolehkan beberapa *actions* untuk jalan berbarengan. Sebuah busur *activity* dapat dibagi menjadi beberapa busur dan beberapa jalur dapat digabung menjadi satu jalur dengan *fork node* dan *join node. Fork nodes* hanya boleh memiliki satu jalur busur masuk dan dua atau lebih jalur

busur keluar. *Join nodes* harus memiliki beberapa jalur busur masuk dan satu jalur busur keluar. *Fork node* dan *join node* dapat digunakan bersama dalam satu diagram.



Gambar 2.18 *Fork Node*

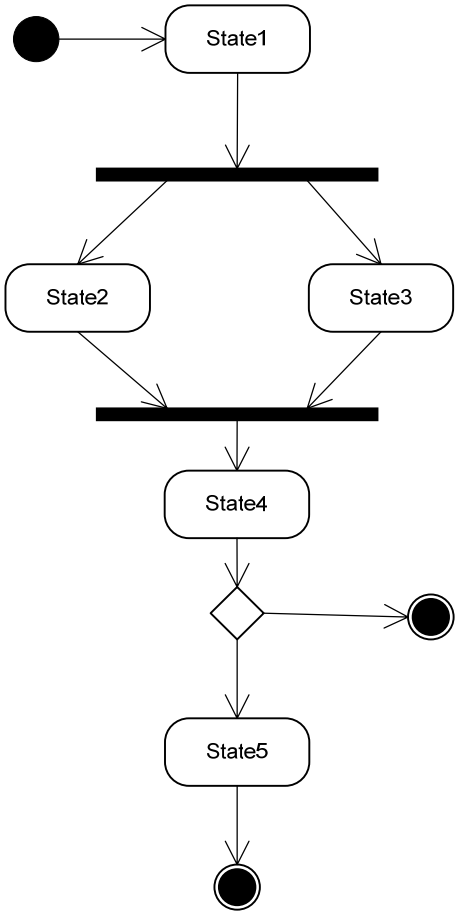
(Bennet, 2002)



Gambar 2.19 *Join node*

(Bennet, 2002)

Contoh Penggunaan *Activity Diagram* :



Gambar 2.20 *Activity Diagram*

(Bennet, 2002)

2.14.4.3 Class Diagram

Menurut Mathiassen *at el.*(2000, p336) Kelas adalah deskripsi koleksi objek yang saling berbagi struktur, pola perilaku, dan atribut. Untuk memodelkan problem domain, aktivitas dimulai dengan aktivitas kelas dan pertanyaan penting tentang objek dan kejadian (*event*) apa yang harus dimasukan dan yang tidak dimasukan ke dalam model. Mathiassen (2000,p49) menjelaskan bahwa kejadian adalah sebuah peristiwa instan yang berhubungan dengan satu objek atau lebih.

Class diagram menggambarkan struktur dan deskripsi *class, package,* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class Diagram* terdiri dari nama, atribut, *method.* Atribut dan *method* dapat memiliki salah satu sifat dari :

Tabel 2.7 Sifat atribut dan *method*

+	*Public*	Dapat dipanggil oleh siapa saja
#	*Protected*	Hanya dapat dipanggil oleh *class* yang bersangkutan dan *class* anak yang mewarisinya
-	*Private*	Tidak dapat dipanggil dari *class* luar yang besangkutan

(Mathiassen, 2000)

Gambar 2.21 *Class diagram*

(Mathiassen, 2000)

Hubungan antara *class* dengan *class* lainnya di dalam *class* diagram terbagi menjadi beberapa hubungan yaitu :

1. Asosiasi

   Asosiasi merupakan 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*.

Gambar 2.22 Asosiasi

(Mathiassen, 2000)

2. Agregasi dan *Composite*

   Agregasi merupakan gabungan tetapi tidak mutlak sedangkan composite merupakan hasil gabungan yang tidak terpisahkan dari *class* lainnya.

Gambar 2.23 Hubungan Agregasi

(Mathiassen, 2000)

Gambar 2.24 Hubungan *Composite*

(Mathiassen, 2000)

3. Generalisasi

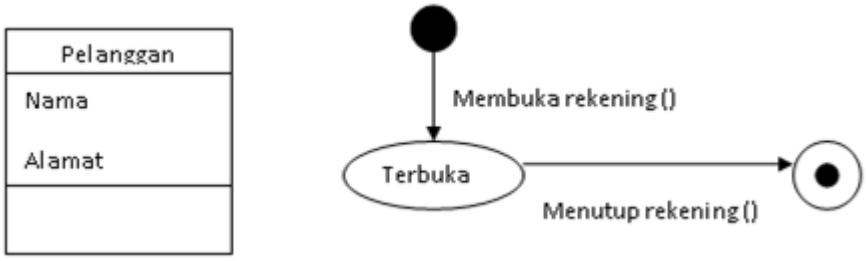
   Generalisasi merupakan hubungan dimana dua atau lebih class dapat melakukan berbagi *attribut* dan *method.*

Gambar 2.25 Generalisasi

(Mathiassen, 2000)

2.14.4.4 Statechart Diagram

Menurut Mathiassen (2000, p341) Definisi kelas dalam kelas diagram dikembangkan lagi dengan menambahkan deskripsi pola perilaku (*behavioral pattern*) dan atribut pada setiap kelas. *Behavioral pattern* adalah sebuah deskripsi jejak kejadian (*event trace*) yang mungkin untuk semua objek di dalam sebuah kelas. Hasilnya diekspresikan dalam diagram *statechart* seperti berikut.



Gambar 2.26 *Statechart Diagram*

(Mathiassen, 2000)

Perilaku sebuah objek didefinisikan sebuah jejak kejadian (*event trace*) yang memperlihatkan kejadian-kejadian dalam urutan waktu. Jejak kejadian adalah urutan kejadian yang berhubungan dengan sebuah objek spesifik, misalnya "membuka rekening-menutup rekening."

Fokus pada analisis *problem domain* adalah pada objek. Oleh karena itu, mendeskripsikan perilaku pada setiap objek dihindari di problem domain. Sebagai gantinya, dideskripsikan pola perilaku untuk objek-objek. Pola perilaku adalah deskripsi jejak kejadian yang mungkin untuk semua objek dalam kelas. Gambar di atas menunjukan sebuah pola perilaku dari objek pelanggan.

Ketika problem domain dimodelkan, juga diformulasikan kebutuhan data yang akan digunakan sistem. Untuk menspesifikasikan data, digunakan atribut. Atribut adalah sebuah properti deskriptif sebuah kelas atau kejadian. Nama dan alamat merupakan atribut kelas pelanggan. Atribut dibuat dengan memperhatikan pola perilaku.

Dalam pola perilaku terdapat pola urutan (*sequence*), pola pemilihan (*selection*), dan pola berulang (*iteration*). Masing masing dinotasikan dengan "+", "|", dan "*". Contohnya membuka rekening + (menyetor tunai | menarik tunai)* + menutup rekening.

2.14.4.5 Sequence Diagram

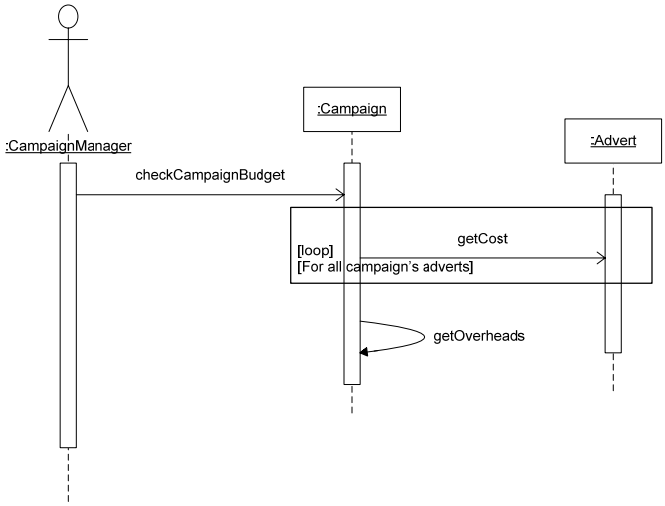
Berdasarkan pendapat Bennet (2002, p253) menyatakan *sequence diagram* digunakan untuk merepresentasikan detail dari interaksi antar objek yang diperlukan untuk satu usecase dalam satu operasi.

Beberapa operator interaksi yang sering digunakan :

Tabel 2.8 Operator interaksi

Operator interaksi	Kegunaan
Alt	menggambarkan behavior alternative
Opt	diekseskusi hanya saat kondisi *constraint true*
Loop	menggambarkan perulangan selama memenuhi *constraint* yang ada

(Mathiassen, 2000)

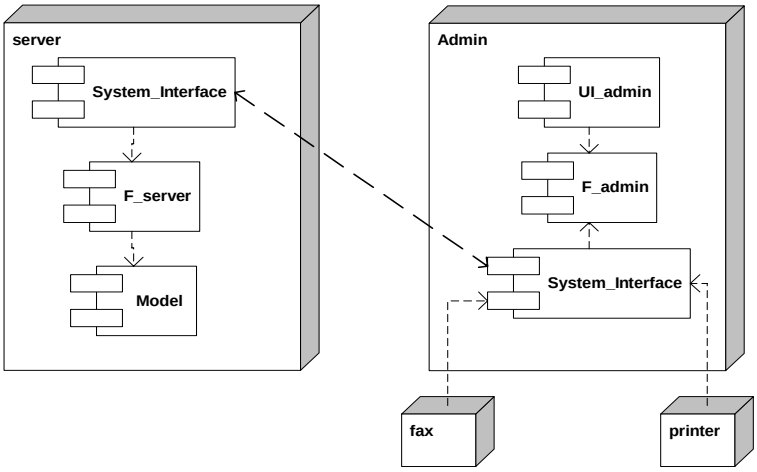


Gambar 2.27 *Sequence Diagram*

(Mathiassen, 2000)

2.14.4.6 Deployment Diagram

Menurut Mathiassen (2000, p340) Arsitektur proses adalah sebuah struktur eksekusi sistem yang terdiri atas proses-proses yang saling bergantung. Unit dasar dalam mengeksekusi sistem adalah sebuah *prossesor*. *Prosessor* adalah peralatan yang dapat mengeksekusi sebuah program. Objek aktif adalah objek yang aktif selama eksekusi sistem. Sedangkan komponen program adalah sebuah modul fisik dari kode program. Aktivitas arsitektur proses menggunakan kelas diagram dan spesifikasi komponen untuk membuat sebuah *deployment diagram* yang menunjukkan prosesor dengan komponen program dan aktif objek.



Gambar 2.28 *Deployment Diagram*

(Mathiassen, 2000)

2.15 Pemrograman Berorientasi Objek

Pemrograman berorientasi objek adalah pemrograman dengan menggunakan paradigma objek. Tiga karakteristik utama dari bahasa yang berorientasi objek adalah *encapsulation, inheritance, polymorphism. Encapsulation* adalah pengemasan data dan fungsi dalam wadah bernama objek. *Inheritance* adalah merupakan sifat dari suatu kelas diturunkan ke kelas lain. *Polymorphism* adalah konsep dimana sesuatu yang sama dapat memiliki berbagai bentuk dan perilaku yang berbeda.

2.16 Bahasa Pemrograman PHP

Menurut Sidik (2004, p1), PHP merupakan *script* untuk pemrograman *script web server-side*, *script* yang membuat dokumen HTML secara *on the fly*, dokumen HTML yang dihasilkan dari suatu aplikasi bukan dokumen HTML yang dibuat dengan menggunakan editor teks atau editor HTML.

Dengan menggunakan PHP maka *maintenance* suatu situs web manjadi lebih mudah. Proses *update* data dapat dilakukan dengan menggunakan aplikasi yang dibuat dengan menggunakan *script* PHP.

PHP secara resmi merupakan kependekan dari PHP:*HyperText Preprocessor*, merupakan bahasa *script server-side* yang disisipkan pada HTML.

2.17 Metode Rekayasa Piranti Lunak

Waterfall Model

Menurut Pressman (2005) Model ini adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*. Berikut ini adalah fase-fase dalam *Waterfall Model* :

1. *Communication* (*project initiation* and *requiements gathering*) menginisialisasikan projek yang akan dikerjakan dan mengumpulkan kebutuhan secara lengkap.
2. *Planning* (*estimating*, *scheduling*, *tracking*)

   Memperkirakan jangka waktu pembangunan projek, membuat jadwal pengerjaan projek.
3. *Modeling* (*analysis and design*)

   Menganalisis kebutuhan rancangan layar dan kemudian mendesain tampilan.
4. *Construction* (*code and test*)

   Mulai memprogram projek yang akan di bangun dan melakukan tes pada program yang telah selesai dibuat.
5. *Deployment* (*delivery*, *support*, dan *feedback*)

   Implementasikan sistem yang telah dibuat serta mendukung aplikasi tersebut untuk berjalan dengan baik, dan mendapatkan umpan balik dari pemakai.