# BAB II
# TINJAUAN PUSTAKA

## II.1. Sistem Informasi Geografis

Menurut Adam Suseno & Ricky Agus (2012) Sistem informasi yang memiliki kepaduan antara teknologi informasi dan aktifitas dari orang yang menggunakan teknologi itu untuk mengembangkan dan mengaplikasikan dalam mendukung sebuah operasi atau manajemen di bidang geografis, merupakan bagian dari perkembangan di ilmu sistem informasi geografis.

Dalam pengembangannya, sistem informasi ini dibuat dengan tujuan memanfaatan teknologi informasi. Hal ini tidak terlepas dari semakin banyaknya *software* yang dibuat untuk membantu dalam pengerjaannya khususnya dalam sistem informasi geografis. Sesuatu yang berhubungan dengan sistem informasi tentunya tidak terlepas dari hubungan dengan sistem data dan aktifitas lain dalam penggunaan *software* nya. Oleh karena itu dibutuhkan sebuah *software* yang mendukung dalam sistem informasi geografis.

Sistem Informasi Geografis atau disingkat SIG dalam bahasa Inggris *Geographic Information System* merupakan sistem informasi khusus yang mengelola data yang memiliki informasi spasial (bereferensi keruangan). Atau dalam arti yang lebih sempit adalah sistem komputer yang memiliki kemampuan untuk membangun, menyimpan, mengelola dan menampilkan informasi bereferensi geografis atau data geospasial untuk mendukung pengambilan

keputusan dalam perencanaan dan pengelolaan suatu wilayah, misalnya data yang diidentifikasi menurut lokasinya, dalam sebuah database.

Teknologi Sistem Informasi Geografis dapat digunakan untuk investigasi ilmiah, pengelolaan sumber daya, perencanaan pembangunan, kartografi dan perencanaan rute. Misalnya, SIG bisa membantu perencana untuk secara cepat menghitung waktu tanggap darurat saat terjadi bencana alam, atau SIG dapat digunaan untuk mencari lahan basah (*wetlands*) yang membutuhkan perlindungan dari polusi atau dapat digunakan mencari informasi sebuah tempat khusus dan banyak manfaat lain yang dapat dikembangkan dalam sistem informasi geografis ini.

### II.1.1. Komponen Sistem Informasi Geografis

Menurut Adam Suseno & Ricky Agus (2012) Komponen-komponen pendukung SIG terdiri dari lima komponen yang bekerja secara terintegrasi yaitu perangkat keras (*hardware*), perangkat lunak (*software*), data, manusia, dan metode yang dapat di lihat pada gambar II.1 sebagai berikut :


**Gambar II.1 : Komponen Sistem Informasi Geografis**

(Sumber : Adam Suseno & Ricky Agus : 2012 )

1. **Perangkat Keras (*hardware*)**

   Perangkat keras SIG adalah perangkat-perangkat fisik yang merupakan bagian dari sistem komputer yang mendukung analisis goegrafi dan pemetaan. Perangkat keras SIG mempunyai kemampuan untuk menyajikan citra dengan resolusi dan kecepatan yang tinggi serta mendukung operasioperasi basis data dengan volume data yang besar secara cepat. Perangkat keras SIG terdiri dari beberapa bagian untuk menginput data, mengolah data, dan mencetak hasil proses. Berikut ini pembagian berdasarkan proses :

   a. Input Data : mouse, digitizier, scanner
   b. Olah Data : harddisk, processor, RAM, VGA card
   c. Output Data : plotter, printer, screening

2. **Perangkat Lunak (*software*)**

   Perangkat lunak digunakan untuk melakukan proses menyimpan, menganalisa, memvisualkan data-data baik data spasial maupun non-spasial. Perangkat lunak yang harus terdapat dalam komponen software SIG adalah :

   a. Alat untuk memasukkan dan memanipulasi data SIG
   b. *Data Base Management System* (DBMS)
   c. Alat untuk menganalisa data-data
   d. Alat untuk menampilkan data dan hasil analisa

**3. Data**

Pada prinsipnya terdapat dua jenis data untuk mendukung SIG yaitu :

a. Data Spasial

Data spasial adalah gambaran nyata suatu wilayah yang terdapat di permukaan bumi. Umumnya direpresentasikan berupa grafik, peta, gambar dengan format digital dan disimpan dalam bentuk koordinat x,y (vektor) atau dalam bentuk image (raster) yang memiliki nilai tertentu.

b. Data Non Spasial (Atribut)

Data non spasial adalah data berbentuk tabel dimana tabel tersebut berisi informasi- informasi yang dimiliki oleh obyek dalam data spasial. Data tersebut berbentuk data tabular yang saling terintegrasi dengan data spasial yang ada.

**4. Manusia**

Manusia merupakan inti elemen dari SIG karena manusia adalah perencana dan pengguna dari SIG. Pengguna SIG mempunyai tingkatan seperti pada sistem informasi lainnya, dari tingkat spesialis teknis yang mendesain dan mengelola sistem sampai pada pengguna yang menggunakan SIG untuk membantu pekerjaannya sehari-hari.

**5. Metode**

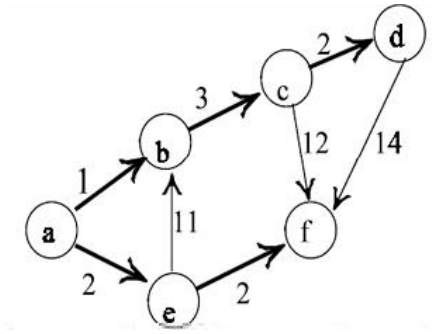
Metode yang digunakan dalam SIG akan berbeda untuk setiap permasalahan. SIG yang baik tergantung pada aspek desain dan aspek realnya.

## II.2 Metode Dijkstra

Menurut Abdul & Agung (2012), Algoritma *Dijkstra* adalah algoritma yang popular untuk menentukan jalur terpendek. Algoritma ini pertama kali dikemukakan oleh Edsger W. Dijkstra pada tahun 1959 dan telah secara luas digunakan dalam menentukan rute tersingkat atau jalur terpendek berdasarkan kriteria tertentu yang digunakan sebagai batasan. Algoritma ini disebut sebagai sumber tunggal algoritma untuk menentukan alur terpendek. Dalam algoritma ini, untuk setiap vertex tertentu disebut sebagai titik jalur terpendek ke semua titik yang lain. Dalam algoritma ini tidak hanya terfokus untuk mencari jalur terpendek dari setiap vertex tetapi ke semua vertex yang tersedia. Algoritma ini dapat diterapkan pada grafik hanya dengan bobot non-negative. Algoritma Dijkstra menemukan jalur terpendek ke graph bervertex diurutan jarak dari sumber tertentu. Dalam proses pencarian jalur terpendek, Algoritma Dijkstra menemukan jalur terpendek dari sumber titik vertex terdekat menuju titik selanjutnya (Abdul & Agung : 2012).

Jalur terpendek (shortest path) dipublikasikan pada tahun 1959 yang berada di Numerische Mathematik yang di edit oleh F.L. Bauer. Pada saat itu algoritma untuk jalur terpendek hampir tidak dianggap. Ada banyak cara untuk pergi dari suatu titik A ke titik B, akan tetapi cara tersebut masih tidak dianggap. Jalur terpendek dapat didefinisikan sebagai masalah kombinatorial dalam grafik dengan bobot terbatas, atau sebagai masalah optimasi yang terus berkelanjutan dalam Geometri Euclidian. Kunci untuk menemukan jalur terpendek adalah dengan

mengetahui bagaimana cara menggambarkannya, salah satunya menggunakan struktur pohon.




| Vertex | a | b | c | d | e | f |
|---|---|---|---|---|---|---|
| Distance | 0 | 1 | 4 | 6 | 2 | 4 |
| Prior | / | a | b | c | a | e |

**Gambar II.2 : Grafik dengan Pohon Jalur Terpendek dalam Sebuah Tabel**

(Sumber : Abdul & Agung : 2012)

Jalur terpendek dari node a ke semua node lain digambarkan dengan menunjukkan node terakhir yang dikunjungi sebelumnya pada setiap node. Sebuah contoh ilustratif digambarkan pada Gambar II.2. Dalam gambar tersebut, jalur terpendek dari node a ditandai dengan tepi yang lebih gelap. Pada gambar di atas dicontohkan suatu graph yang memiliki jarak yang berbeda - beda dari satu titik ke titik lainnya. Diperlukan sebuah algoritma untuk menyelesaikan permasalahan rute pada gambar tersebut, salah satunya menggunakan Algoritma *Dijkstra*. Algoritma *Dijkstra* menurut penemunya, seorang ilmuwan komputer, Edsger Dijkstra), adalah sebuah algoritma rakus (*greedy algorithm*) yang dipakai dalam memecahkan permasalahan jarak terpendek (*shortest path problem*) untuk sebuah graf berarah

(*directed graph*) dengan bobot-bobot sisi (*edge weights*) yang bernilai tak negatif. Misalnya, bila vertices dari sebuah graf melambangkan kota-kota dan bobot sisi (*edge weights*) melambangkan jarak antara kota-kota tersebut, maka algoritma Dijkstra dapat digunakan untuk menemukan jarak terpendek antara dua kota. Input algoritma ini adalah sebuah graf berarah yang berbobot (*weighted directed graph*) G dan sebuah sumber vertexs dalam G dan V adalah himpunan semua vertexs dalam graph. Setiap sisi dari graf ini adalah pasangan vertices (u,v) yang melambangkan hubungan dari vertex u ke vertex v. Himpunan semua tepi disebut E. Bobot (*weights*) dari semua sisi dihitung dengan fungsi :

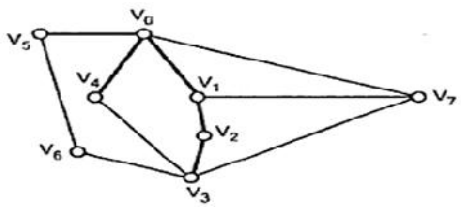
$$w : E \rightarrow [0, \infty]$$

jadi w (u,v) adalah jarak tak negatif dari vertex u ke vertex v.

Biaya (cost) dari sebuah sisi dapat dianggap sebagai jarak antara dua vertex, yaitu jumlah jarak semua sisi dalam jalur tersebut. Untuk sepasang vertex s dan t dalam V, algoritma ini menghitung jarak terpendek dari s ke t.

Algoritma *Dijkstra* adalah algoritma yang popular untuk menentukan jalur terpendek. Algoritma ini disebut sebagai sumber tunggal algoritma untuk menentukan jalur terpendek. Dalam algoritma ini, untuk setiap vertex tertentu disebut sebagai titik jalur terpendek ke semua titik yang lain. Dalam algoritma ini tidak hanya terfokus untuk mencari jalur terpendek dari setiap vertex tetapi ke semua vertex yang tersedia. Algoritma ini dapat diterapkan pada grafik hanya dengan bobot non-negative. Algoritma *Dijkstra* menemukan jalur terpendek ke graph bervertex diurutan jarak dari sumber tertentu. Dalam proses pencarian jalur

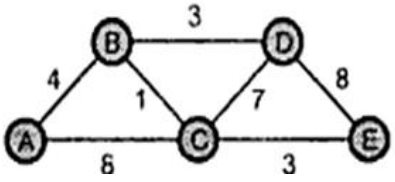
terpendek, Algoritma *Dijkstra* menemukan jalur terpendek dari sumber titik vertex terdekat menuju titik selanjutnya. Contoh graph dapat dilihat pada Gambar II.3. berikut.




**Gambar II.3 : Contoh Graph**

(Sumber : Abdul & Agung : 2012)

Jalur terpendek dari V0 telah diperoleh. Pertama temukan jalur terpendek dari V0 – V1 kemudian V1 – V2 kemudian dari V2 – V3 maka jalur terpendek diperoleh (Gambar II.4).

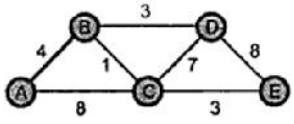


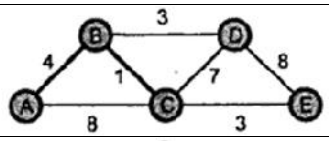
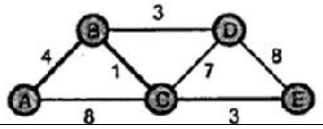

**Gambar II.4 : Contoh Jalur yang Memiliki Jarak**

(Sumber : Abdul & Agung : 2012)

Setiap vertex sebagai sumber akan menemukan jalur terpendek dari titik satu ke titik yang lainnya dan tampak pada Tabel II.1 berikut :

**Tabel II.1 Penyelesaian pada Sebuah Kasus**

| Source Vertex | Distance With other Vertices | Path Shown in Graph |
|---|---|---|
| A | A-B, Path = 4<br>A-C, Path = 8<br>A-D, Path = ∞<br>A-E, Path = ∞ |  |

| | | |
|---|---|---|
| B | B-C, Path = 4 +1<br>B-D, Path = 4 +3<br>B-E, Path = ∞ |  |
| C | C-D, Path = 5 + 7 = 12<br>C-E, Path = 5 +3 = 8 |  |
| D | D-E, Path = 7 + 8 = 15 | |

Dari Tabel II.1 diperoleh jalur terpendek dari titik A sampai dengan E dan titik itu adalah A – B – C – E dengan panjang jarak = 4 + 1 + 3 = 8. Demikian pula jalur terpendek lainnya dapat diperoleh dengan memilih titik awal dan tujuannya.

### II.3. PHP

Menurut (Anhar ; 2010) PHP singkatan dari *hypertext preprocessor* yaitu bahasa pemrograman *web server-side* yang bersifat *open source*. PHP merupakan *script* yang terintegrasi dengan HTML dan berada pada server (*server side HTML embedded scripting*). Selanjutnya PHP adalah *script* yang digunakan untuk membuat halaman *website* yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Mekanisme ini menyebabkan informasi yang diterima client selalu yang terbaru/*up to date*. Semua *script* PHP dieksekusi pada *server* dimana *script* tersebut dijalankan.

#### II.3.1. Kombinasi HTML dan PHP

PHP dalam sistem aplikasi web berfungsi sebagai *server side scripting language*, yaitu sebagai sederetan kode yang dieksekusi seluruhnya di server, kemudian hasil dari eksekusi tersebut dikirim ke klien. Kita dapat

mengkombinasikan HTML dan PHP, yaitu dengan menuliskan script HTML dibagian luar tag PHP seperti terlihat pada contoh berikut.

```
<html>
<head>
<title>Penggabungan PHP dan HTML</title>
</head>
<body>
<?php
echo “Ini adalah contoh penggabungan script HTML
dan PHP”;
?>
</body>
</html>
```

Setelah selesai, simpan dengan nama kombinasi.php, lalu lihat hasilnya di browser, maka tampilanya tampak seperti dibawah ini.


**Gambar II.5. Mengakses kombinasi.php**

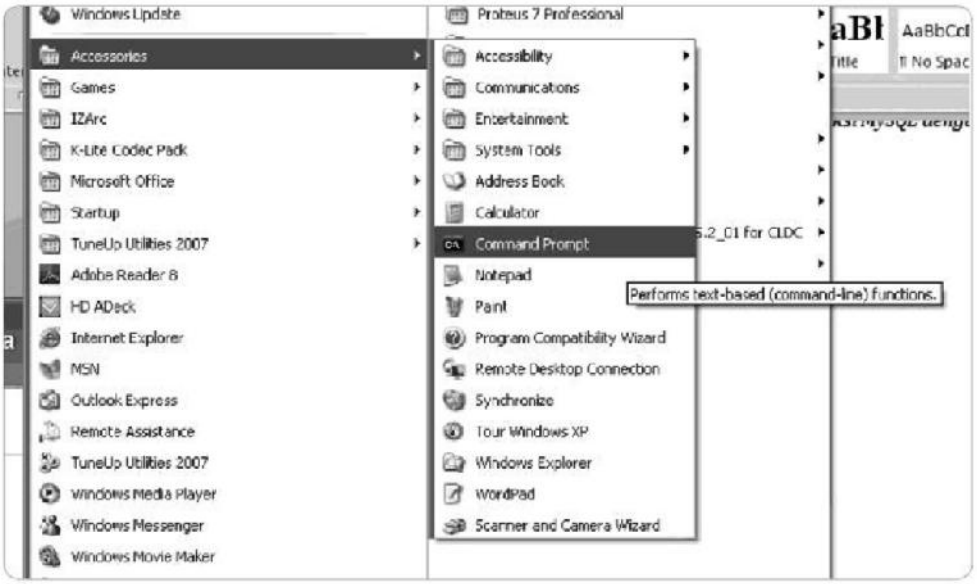
## II.4. MySQL

Menurut Anhar (2010), MySQL (*My Structure Query Language*) adalah salah satu *Database Management System* (DBMS) dari sekian banyak DBMS seperti *Oracle, MS SQL, Postgre SQL*, dan lainya. Mysql berfungsi untuk mengolah database menggunakan bahasa SQL. Mysql bersifat *open source* sehingga kita bisa menggunakan secara gratis. Pemrograman PHP juga sangat mendukung/support dengan database MySQL.

### II.4.1. Mengakses MySQL

Sebelum mengakses MySQL, terlebih dahulu kita harus menghidupkan *server*. Caranya dengan klik dua kali file winmysqladmin.exe yang berada di direktori C:\apache\mysql\bin. Kemudian tekan OK. Untuk mengakses MySQL dengan menggunakan *Command Prompt*, ikuti langkah-langkah berikut ini :

1. Klik menu Start → All Program → Accesories → Command Prompt.




**Gambar II.6. Klik Command Prompt**

2. Aplikasi *Command Prompt* akan dijalankan, ketik cd\.
3. Lalu ketikkan cd apache tekan Enter, cd mysql tekan Enter, kemudian ketik cd bin dan tekan Enter, serta mysql dengan diakhiri tekan Enter lagi.


**Gambar II.7. Mengakses MySQL dengan CommandPrompt**

Apabila kita ingin menggunakan cara yang lebih mudah dan cepat, klik dua kali atau jalankan aplikasi bernama mysql.exe yang berada pada direktory C:\apache\mysql\bin.

### II.4.2. Perintah Dasar MySQL

Perintah dasar membuat databasenya, yaitu :

- CREATE DATABASE <nama database>

  Contoh : CREATE DATABASE daftar;

Perintah menggunakan database yaitu :

- USE <nama database>

  Contoh : USE daftar;

Perintah membuat tabel yaitu

- CREATE TABLE NamaTabel (NamaKolom1 tipedata (ukuran), NamaKolom2 tipedata (ukuran));

  Contoh : CREATE TABLE anggota (name varchar(20), pswd varchar(32));

**II.5. Database**

Database adalah sekumpulan tabel-tabel yang berisi data dan merupakan kumpulan dari *field* atau kolom. Struktur file yang menyusun sebuah database adalah *Data Record* dan *Field* (Anhar : 2010).

a. Data adalah satu satuan informasi yang akan diolah. Sebelum diolah, data dikumpulkan di dalam suatu *file database*.
b. *RECORD* adalah data yang isinya merupakan satu kesatuan seperti NamaUser dan *Password.* Setiap keterangan yang mencakup Nama *User* dan *Password* dinamakan satu *record*. Setiap *record* diberi nomor urut yang disebut nomor record *(Record Number)*.
c. *FIELD* adalah sub bagian dari *record*. Dari contoh isi *record* di atas, maka terdiri dari 2 *field*, yaitu: *field* Nama *User* dan *Password*

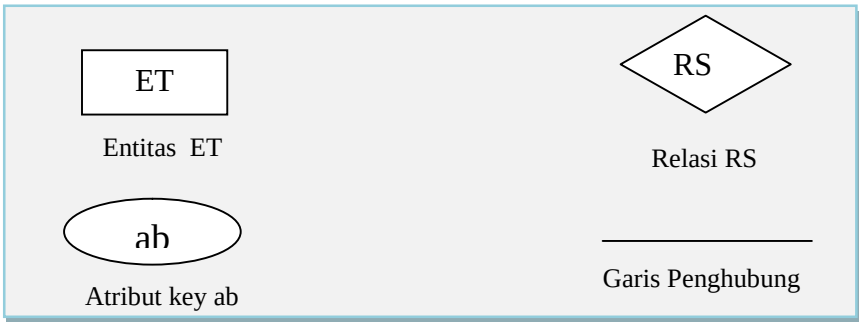
Selain berisi data, *database* juga berisi *metadata*. Metadata adalah data yang menjelaskan tentang struktur dari data itu sendiri. Sebagai contoh, Anda dapat memperoleh informasi tentang nama-nama kolom dan tipe yang ditampilkan tersebut disebut *metadata.*

### II.5.1. Pemodelan Data

Menurut Yudi Priyadi (2014) Terdapat beberapa penjelasan mengenai pemodelan basis data. Suatu basis data dapat digunakan secara bebas untuk menggambarkan dan memberikan deskripsi mengenai kumpulan informasi uang tersimpan dalam *data storage* komputer. Secara sederhana, defenisi untuk model basis data adalah sekumpulan notasi atau simbol untuk menggambarkan data dan relasinya, berdasarkan suatu konsep dan aturan tertentu suatu pemodelan.

### II.5.2. Notasi Diagram E-R

Menurut **(**Yudi Priyadi ; 2014 : 20) Pemodelan basis data dengan menggunakan diagram relasi antar entitas, dapat dilakukan dengan menggunakan suatu pemodelan basis data yang bernama Diagram *Entity-Relational* (selanjutnya disingkat Diagram E-R). Pada Gambar II.8, terdapat suatu simbol/notasi dasar yang digunakan pada Diagram E-R,yaitu entitas,relasi, atribut, dan garis penghubung.




**Gambar II.8 : Notasi Dasar Diagram E-R**

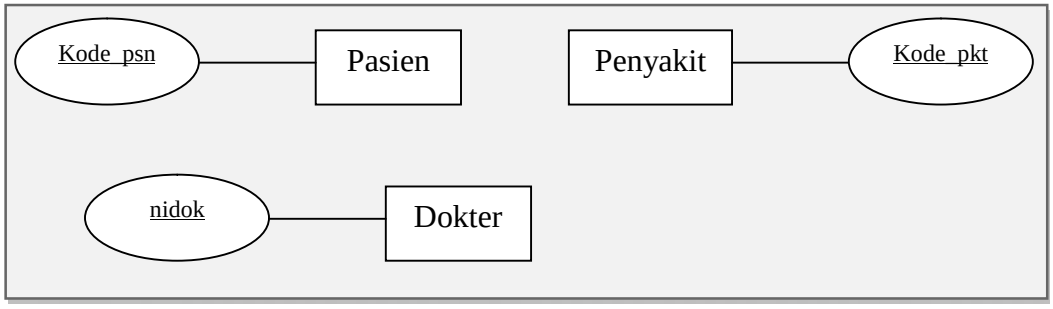
(Sumber : Yudi Priyadi ; 2014 : 20)

1. Entitas

   Merupakan notasi untuk mewakili suatu objek dengan karakteristik sama, yang dilengkapi oleh atribut, sehingga pada suatu lingkungan nyata setiap objek akan berbeda dengan objek lainya. Pada umumnya, objek dapat berupa benda, pekerjaan, tempat dan orang.

2. Atribut

   Merupakan notasi yang menjelaskan karakteristik suatu entitas dan juga relasinya. Atribut dapat sebagai key yang bersifat unik, yaitu *Primary Key* atau *Foreign Key.* Selain itu, atribut juga dapat sebagai atribut deskriptif saja, yaitu sebagai pelengkap deskripsi suatu entitas dan relasi.

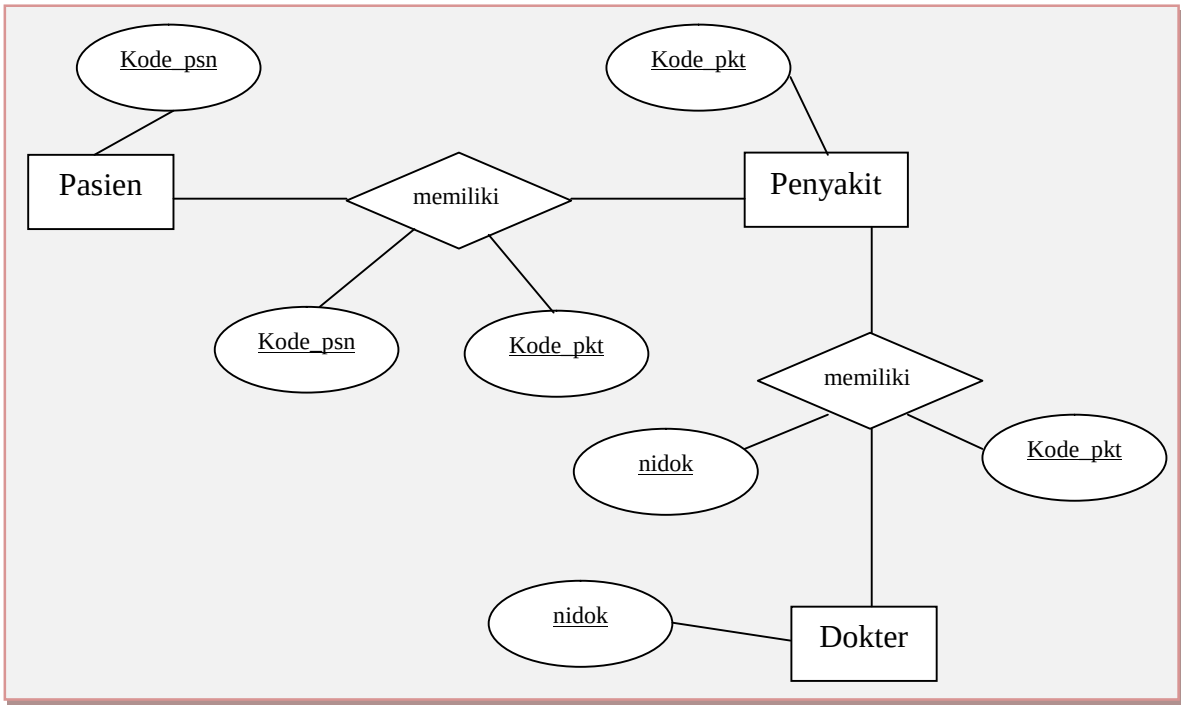



**Gambar II.9 : Atribut Key pada Entitas**

(Sumber : Yudi Priyadi ; 2014 : 23)

3. Relasi

   Merupakan notasi yang digunakan untuk menghubungkan beberapa entitas berdasarkan fakta pada suatu lingkungan.




**Gambar II.10 : Pemilihan Relasi untuk Entitas**

(Sumber : Yudi Priyadi ; 2014 : 25)

4. Garis penghubung

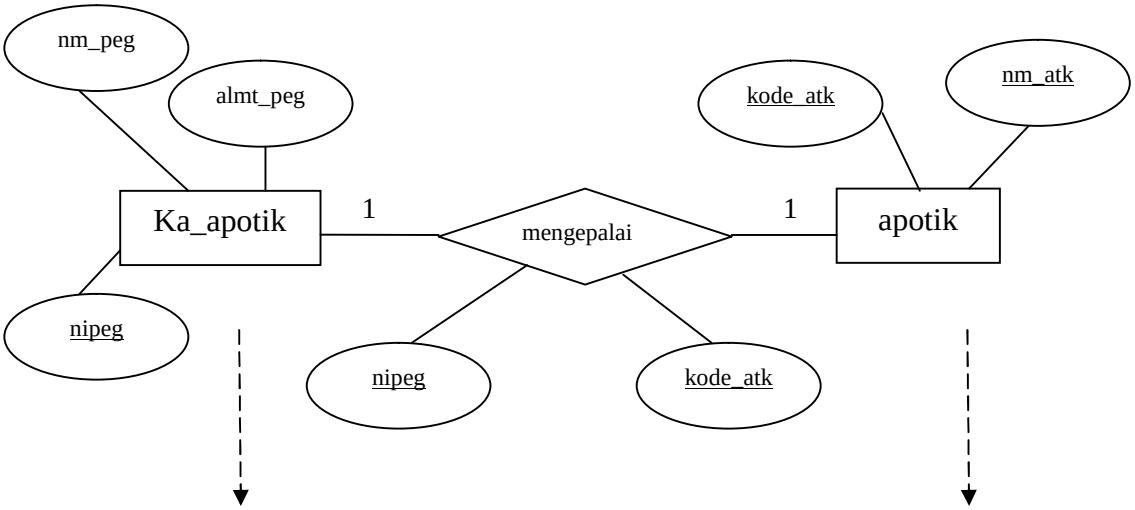
   Merupakan notasi untuk merangkaikan keterkaitan antar notasi yang digunakan dalam Diagram E-R, yaitu entitas, relasi dan atribut.

### II.5.3. Kardinalitas

Variasi kardinalitas untuk satu ke satu (1:1), satu ke banyak (1:N), banyak ke satu (N:1), dan banyak ke banyak (N:N) akan mempertegas perbedaan bentuk tabel fisik hasil dari konversi Diagram E-R.

1. Kardinalisasi Satu ke Satu

   Berikut disajikan suatu ilustrasi yagn memiliki kardinalisasi satu ke satu (1:1)




Tabel ka_apotek

| nipeg | nm_peg | almt_peg |
|---|---|---|
| | | |
| | | |

Tabel apotik

| kode_atk | nm_atk | nipeg |
|---|---|---|
| | | |
| | | |

**Gambar II.11 : Konveris Kardinalitasa Satu ke Satu**

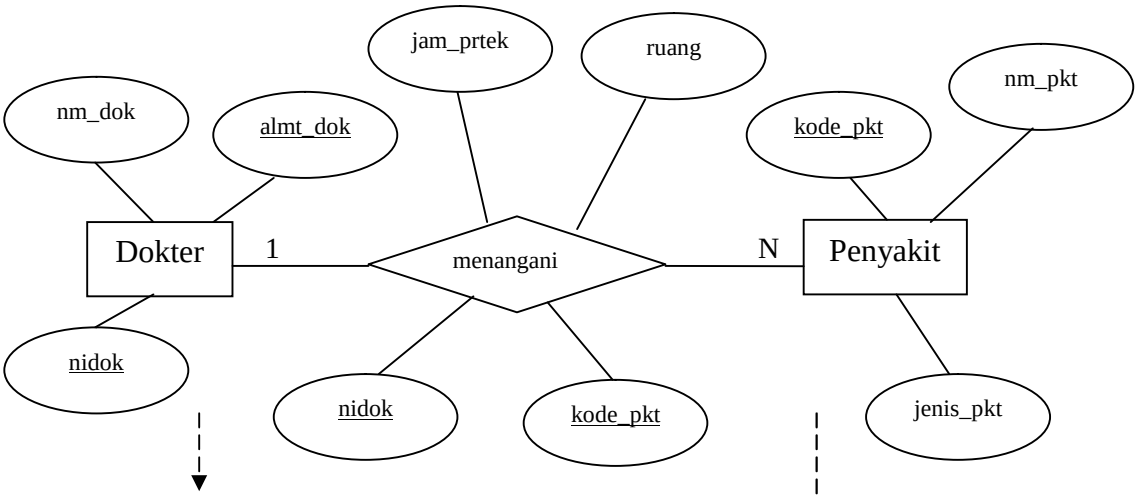
(Sumber : Yudi Priyadi ; 2014 : 43)

Berdasarkan ilustrasi pada Gambar II.11, terdapat keteraturan bahwa entitas ka_apotik akan dikonversikan menjadi tabel ka_apotik, kemudian semua atributyang ada pada entitas tersebut dikonversikan menjadi kolomatau field. Begitu juga pada entitasapotik yang dikonversikan menjadi tabel apotik, kemudian semua atribut yang ada pada entitas tersebut dikonversikan menjadi kolom atau field.

2. Kardinalitas satu ke banyak

   Pada bagian ini terdapat gambar yang menyajikan suatu ilustrasi untuk konversi Diagram E-R yang memiliki kardinalitas satu ke banyak (1:N).

Jumlah atribut yang digunakan pada simbol relasi akan sangat menentukan terhadap konversi bentuk tabel fisiknya, untuk mempertegas perbedaan kardinalitas satu ke satu dengan satu ke banyak




Tabel Dokter

| nidok | nm_dok | almt_dok |
|---|---|---|
| | | |
| | | |

Tabel penyakit

| kode_pkt | nm_pkt | jenis_pkt | nidok | ruang | jam_prtek |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |

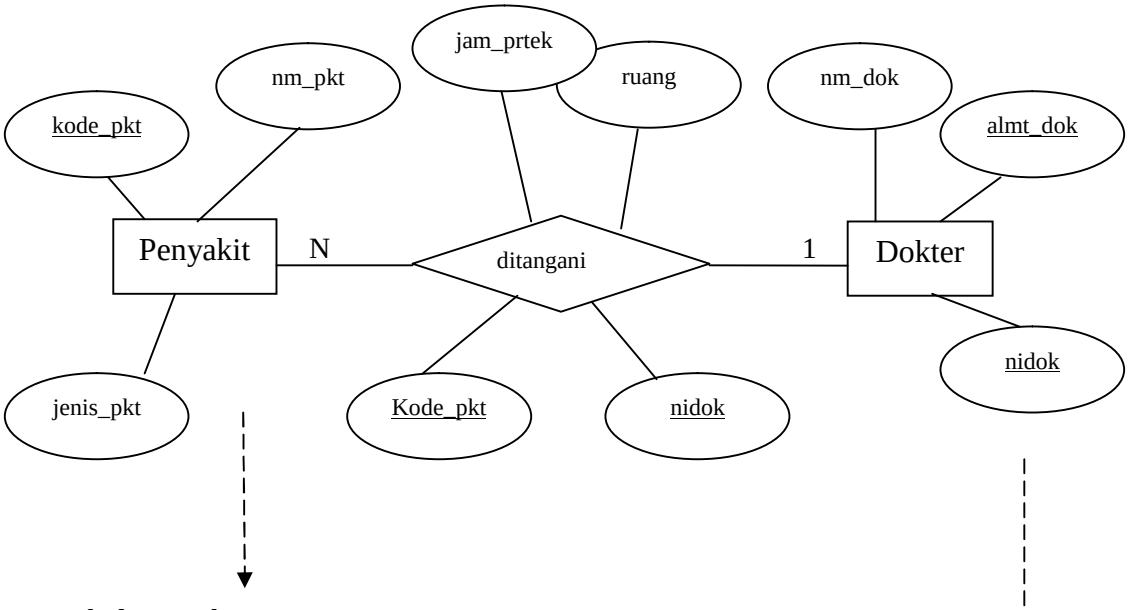
**Gambar II.12 : Konveris Kardinalitasa Satu ke Banyak**

(Sumber : Yudi Priyadi ; 2014 : 45)

3. Kardinalitas Banyak ke Satu

Merujuk pada penjelasan mengenai konversi Diagram E-R untuk konsep kardinalitas satu ke banyak (1:N) yang terdapat pada bagian sebelumya, pada kardinalitas banyak ke satu (N:1) merupakan konsep sebaliknya.

Penggunaan kata pada simbol relasi untuk konsep ini, umumnya dibaca menjadi kalimat pasif. Jika dibaca secara bebas, arti dari konversi Diagram E-R menjadi tabel fisik sebagai berikut




Tabel penyakit

| kode_pkt | nm_pkt | jenis_pkt | nidok | ruang | jam_prtek |
|---|---|---|---|---|---|
| | | | | | |
| | | | | | |

Tabel dokter

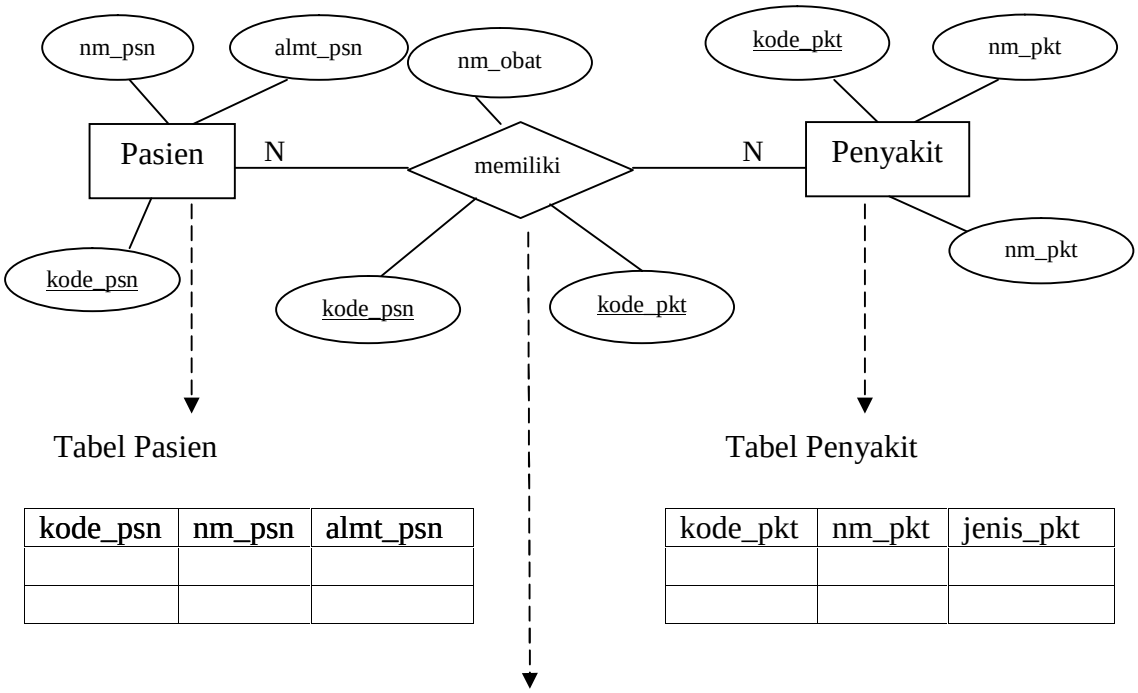
| nidok | nm_dok | almt_dok |
|---|---|---|
| | | |
| | | |

**Gambar II.13 : Konveris Kardinalitasa Banyak ke Satu**

(Sumber : Yudi Priyadi ; 2014 : 47)

4. Kardinalisasi Banyak ke Banyak

Pada relasi yang menggunakan kardinalisasi banyak ke banyak (N:N) sebagai penghubung antar entitas, relasi tersebut akan selalu menjadi tabel fisik pada basis datanya. Pada entitas pasien, entitas penyakit, dan relasi memiliki, dapat dipastikan menjadi tabel




Tabel Pasien

| kode_psn | nm_psn | almt_psn |
|---|---|---|
| | | |
| | | |

Tabel Penyakit

| kode_pkt | nm_pkt | jenis_pkt |
|---|---|---|
| | | |
| | | |

Relasi memiliki ↔ Tabel pemilik_obat

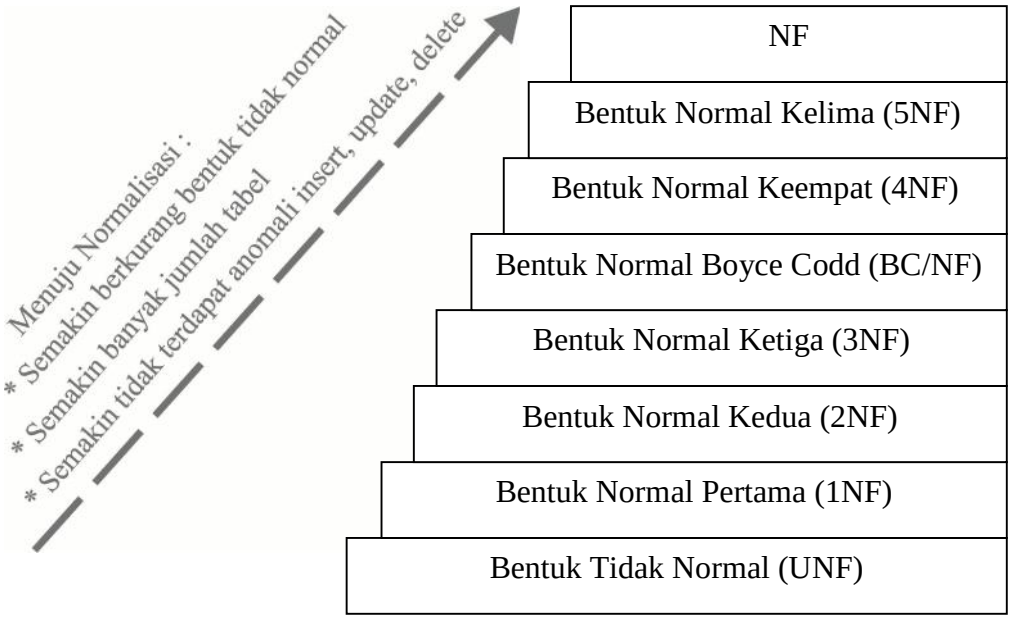
| kode_psn | Kode_pkt | nm_obat |
|---|---|---|
| | | |
| | | |

**Gambar II.14 : Konveris Kardinalitasa Banyak ke Banyak**

(Sumber : Yudi Priyadi ; 2014 : 49)

### II.5.4. Normalisasi

Menurut Yudi Priyadi (2014) Normalisasi merupakan proses sistematis yang dilakukan pada struktur tabel basis data menjadi struktur tabel yang memiliki integritas data, sehingga tidak memiliki data anomali pada saat melakukan *insert*, *delete*, dan *update*. Pada Gambar II.14, tahapan proses sistematis yang dilakukan mulai dari bentuk tidak normal menjadi bentuk normal memiliki suatu syarat yang harus dipenuhi pada saat menuju suatu bentuk yang lebih baik (*well structured relation*).




**Gambar II.15 : Tahapan Proses Bentuk Normalisasi**

(Sumber : Yudi Priyadi ; 2014 : 67)

Setiap syarat dalam tahapan suatu bentuk normal memiliki keterkaitan, hal ini disebabkan karena pada setiap bentuk normal mengalami penyempurnaan untuk bentuk normal selanjutnya. Bentuk tidak normal akan semakin berkurang, setelah

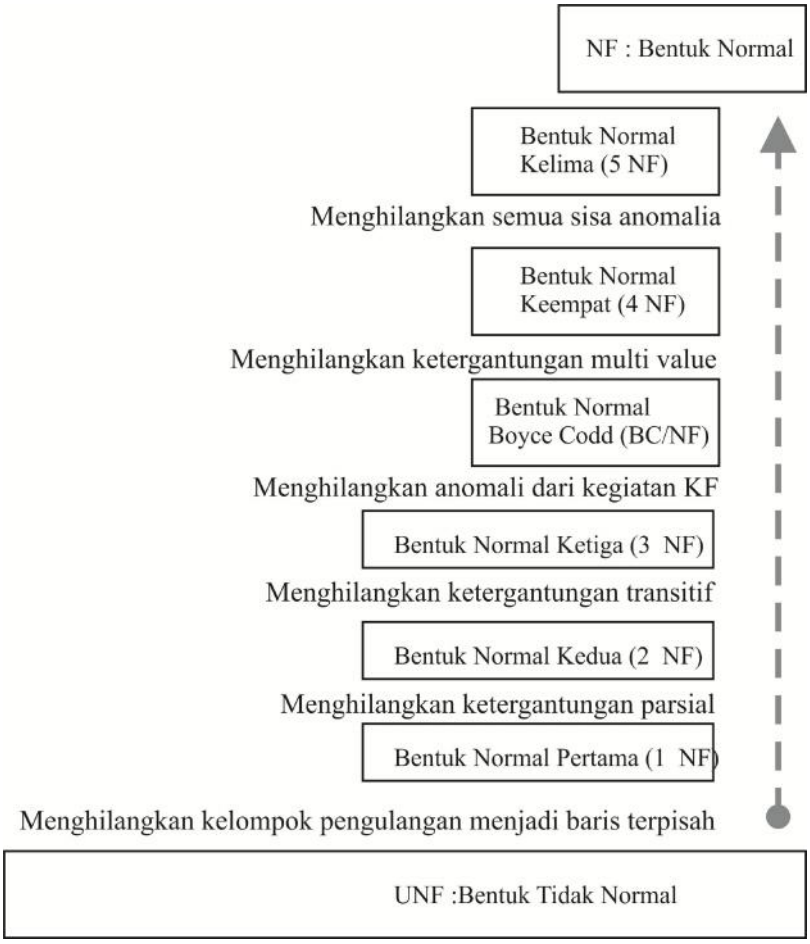
melalui tahapan berubahan bentuk normalisasi, sehingga berdampak pada jumlah tabel yang semakin banyak, tetapi menuju perbaikan ke dalam bentuk *well structured relation.* Hal ini terjadi akibat dari pengelompokan data suatu tabel agar memiliki ketergantungan secara fungsional.

**II.5.5. Aturan Proses Normalisasi**

Menurut **(**Yudi Priyadi ; 2014 : 68) Secara sederhana, kegiatan normalisasi adalah melakukan dekomposisi atau penguraian tabel beserta datanya, menjadi tabel yang normal menurut konsep RDBMS. Merujuk pada gambar II.6, dekomposisi diawali dengan melakukan analisis pada suatu tabel atau beberapa contoh formulir yang sudah memiliki data lengkap dalam basis data, tetapi masih dalam bentuk yang tidak normal (UNF). Oleh karena itu agar dapat memenuhi syarat bentuk normal pertama (1NF), pada setiap barisnya diisikan suatu *value* dengan kelompok data yang sama, berdasarkan suatu atribut key. Dengan demikian, kelompok pengulangan dalam suatu baris dapat dihilangkan, karena sudah tidak terdapat *value* yang kosong untuk setiap *field* dan *record*nya

Setelah memenuhi syarat bentuk normal pertama (1NF), proses berikutnya adalah menghilangkan ketergantungan secara parsial, yaitu dengan cara melakukan dekomposisi tabel menjadi beberapa kelompok tabel berdasarkan field yang memiliki status sebagai key. Hal ini dapat dilakukan oleh salah satu field saja, dengan tetap tidak mengubah arti relasi dan ketergantungannya. Oleh sebab itu, disebut ketergantungan fungsional sebagian (*partiallly functional*), sehingga syarat bentuk normal kedua (2NF) sudah tercapai.

Bentuk normal kedua (2NF) merupakan syarat yang harus dimiliki untuk menuju bentuk normal ketiga (3NF). Pada proses ini, dilakukan dengan menghilangkan ketergantungan secara transitif, yaitu suatu konsep untuk tabel dari hasil relasi yang didalamnya terdapat ketergantungan secara tidak langsung pada beberapa atributnya. Pada umumnya proses normalisasi sudah dapat tercapai pada bentuk normal ketiga (3NF), yaitu dengan menghasilkan tabel yang tidak mengalami anomali basis data pada saat proses *insert, delete,* dan *update.*




**Gambar II.16 : Tahapan Aturan Proses Normalisasi**

(Sumber : Yudi Priyadi ; 2014 : 69)

## II.6. Unified Modeling Language (UML)

Menurut (Rosa A.S & M. Shalahuddin ; 2011 : 118) Pada perkembangan teknik pemrograman berorientasi objek, muncullah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakanteknik pemrograman berorientasi objek, yaitu *Unified Modeling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasikan, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak. UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks-teks pendukung.

UML hanya berfungsi untuk melakukan pemodelan, jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metode berorientasi objek.

Menurut (Prabowo Pudjo Widodo & Herlawati ; 2011 : 6) UML diaplikasikan untuk maksud tertentu, biasanya antara lain :

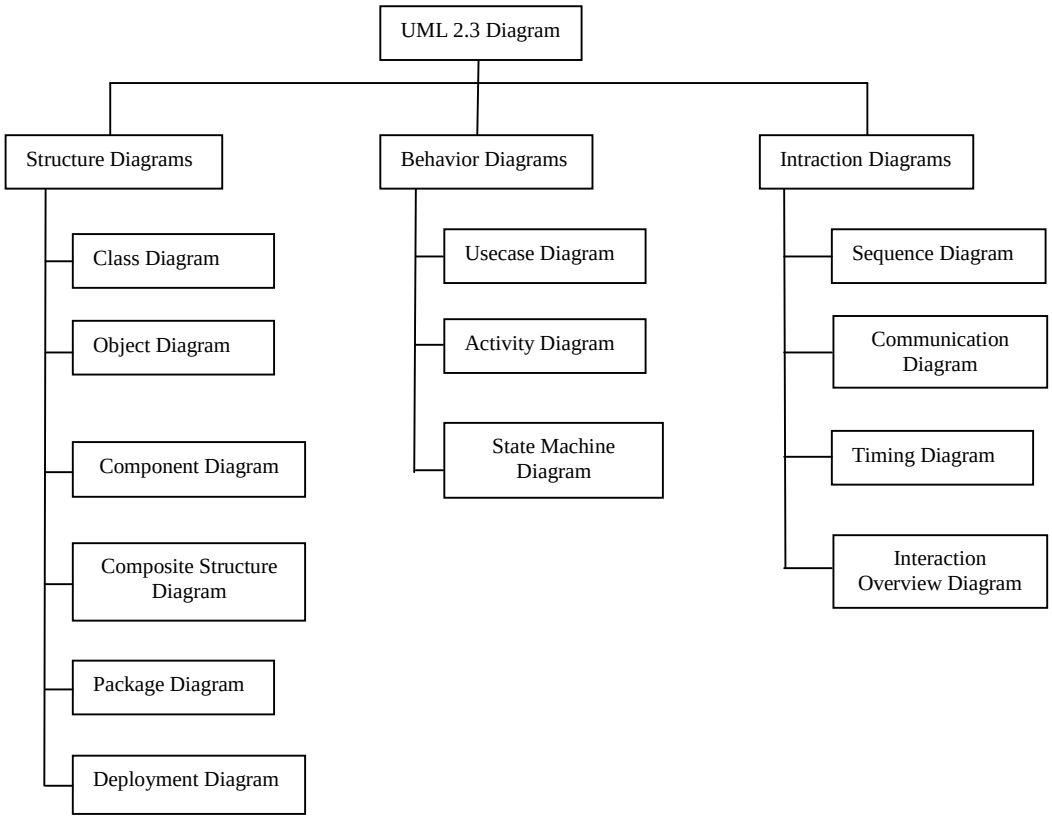
1. Merancang perangkat Lunak.
2. Sarana Komunikasi antara perangkat lunak dengan proses bisnis.
3. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
4. Mendokumentasikan sistem yang ada, proses-proses dan organisasinya.

Blok pembangunan utama UML adalah diagram. Beberapa diagram ada yang rinci (jenis *timming diagram*) dan lainya ada yang bersifat umum (misalnya diagram kelas). Para pengembang sistem berorientasi objek menggunakan bahasa

model untuk menggambarkan, membangun dan mendokumentasikan sistem yang mereka rancang. UML memungkinkan para anggota team untuk bekerja sama dengan bahasa model yang sama dengan mengaplikasikan beragam sistem. Intinya UML merupakan alat komunikasi yang konsisten dalam mendukung para pengembang sistem saat ini.

### II.6.1 Diagram-Diagram UML

Menurut (Rosa A.S & M. Shalahuddin ; 2011 : 120) Pada UML 2.3 terdiri dari 13 macam diagram yang dikelompokan dalam 3 kategori. Pembagian kategori dan macam-macam diagram tersebut dapat dilihat pada gambar II.16 di bawah ini




**Gambar II.17 : Diagram UML**

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 121)

Berikut ini penjelasan singkat dari pembagian kategori tersebut

1. *Structure Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.
2. *Behavior Diagrams* yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.
3. *Interaction Diagrams* yaitu kumpulan diagram yang digunakan untukmenggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

## A. *Class Diagram*

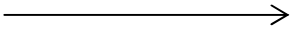
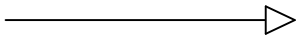
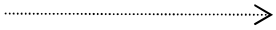
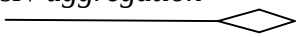
Diagram kelas atau *Class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem.

Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1) Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas
2) Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Berikut gambar II.18 menerangkan simbol-simbol pada diagram kelas :

| Simbol | Deskripsi |
|---|---|
| Kelas<br>Nama_kelas<br>+atribut<br>+operasi() | Kelas pada struktur sistem |
| Antarmuka / *interface*<br>Nama_*interface* | Sama dengan konsep *interface* dalam pemrograman berorientasi objek |
| Asosiasi / *association* | Relasi antar kelas dengan makna umum, |

| | |
|---|---|
| Asosiasi berarah/ *directed association*  | asosiasi biasanya juga disertai dengan *multiplicity*<br>Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity* |
| Generalisasi  | Relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus) |
| Kebergantungan  | Relasi antar kelas dengan makna kebergantungan antar kelas |
| Agregasi / *aggregation*  | Semua bagian (*whole part*) |

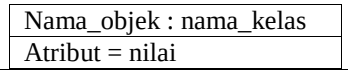
**Gambar II.18 : Diagram Kelas**

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 124)

## B. *Object Diagram*

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefenisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefenisian kelas itu tidak dapat dipertanggungjawabkan. Untuk apa mendefenisikan sebuah kelas sedangkan pada jalannya sistem, objeknya tidak pernah dipakai.

Berikut adalah gambar II.18 menerangkan simbol-simbol diagram objek

| Simbol | Deskripsi |
|---|---|
| Objek   | Objek dari kelas yang berjalansaat sistem dijalankan |
| Link  | Relasi antar objek |

**Gambar II.19 : Diagram Paket**

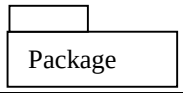
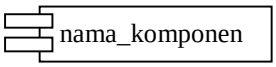
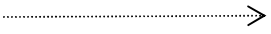
(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 124)

## C. *Component Diagram*

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan di antara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada didalam sistem. Komponen dasar yang biasanya ada dalam suatu sistem adalah sebagai berikut :

1) Komponen *user interface* yang menangani tampilan
2) Komponen *bussiness procesiing* yang menangani fungsi-fungsi proses bisnis
3) Komponen data yang menangani manipulasi data
4) Komponen *security* yang menangani keamanan sistem

Komponen lebih terfokus pada penggolongan secara umum fungsi-fungsi yang diperlukan, berikut gambar II.20 yang menerangkan simbol-simbol yang ada pada diagram komponen

| Simbol | Deskripsi |
|---|---|
| Package   | *Package* merupakan sebuah bungkusan dari satu atau lebih komponen |
| Komponen   | Komponen Sistem |
| Kebergantungan / *dependency*  | Kebergantungan antar komponen, arah panah mengarah pada komponen yang dipakai |
| Antar muka / *interface*  nama_interface | Sama dengan konsep *interface*pada pemrograman berorientasi objek, yaitu sebagai antarmuka komponen agar tidak mengakses langsung komponen |
| | |

| | |
|---|---|
| Link ——————— | Relasi antar komponen |

**Gambar II.20 : Diagram Komponen**

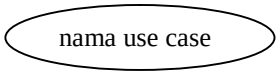
(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 126)

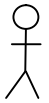
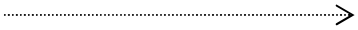
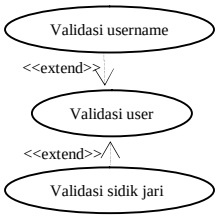
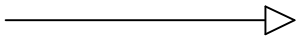
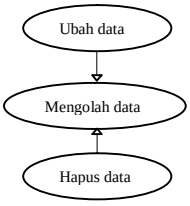
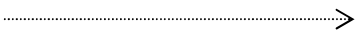
**D. *Use Case Diagram***

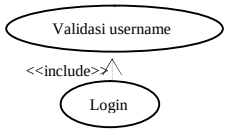
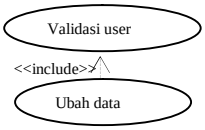
*Use case* atau diagram *use case* merupakan pemodelan untuk kelakuan (*behaviour)*sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar,*use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu. Syarat penamaan pada *use case* adalah nama didefenisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefenisian apa yang disebut aktor dan *use case*.

1) Aktor merupakan orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang.
2) *Use case* merupakan fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor.

Berikut gambar II.21 menerangkan simbol-simbol pada diagram *use case*

| Simbol | Deskripsi |
|---|---|
| Use case   | Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata |

| | |
|---|---|
| | kerja di awal frase nama *use case* |
| Aktor / *actor*<br>  <br>nama aktor | Orang, proses, atau sistem yang lain berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan di buat itu sendiri |
| Asosiasi / *association*<br> | Komunikasi antara aktor dan *use case* yang berpartisipasi pada *use case*, atau *usecase* memiliki interasi dengan aktor |
| Ekstensi / *extend*<br><<extend>><br>  | Relasi usecase tambahan ke subuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanppa *use case* tambahan itu, mirip dengan prinsip inheritance pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan misal<br>  <br>arah panah mengarah pada *use case* yang ditambahkan |
| Generalisasi / *generalization*<br>  | Hubungan generalisasi dan spesialisasi (umum – khusus) antara dua buah use case dimana fungsi yang satu adalah fungsi yang lebih umum dari lainya misalnya :<br>  <br>Arah panah mengarah pada use case yang menjadi generalisasinya (umum) |
| Menggunakan / *include* / *uses*<br><<include>><br>  | Relasi use case tambahan ke sebuah use case dimana use case yang ditambahkan memerlukan use case ini untuk menjalankan fungsinya atau sebagai |

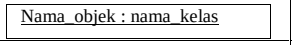
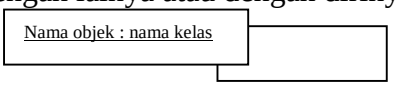
| | |
|---|---|
| <<uses>> | syarat dijalankan use case ini<br>Ada 2 sudut pandang yang cukup besar mengenai include di usecase<br>1. include berarti use case yang ditambahkan akan selalu dipanggil saat use case dijalankan misal pada kasus berikut :<br>  <br>2. include berarti use case yang tambahan akan selalu melakukan pengecekan apakah use case yang di tambahkan telah di jalankan sebelum use case tambahan di jalankan, misal pada kasus berikut :<br>  <br>Kedua interpretasi di atas dapat dianut salah satu atau keduanya tergantung pada pertimbangan dan interpretasi yang dibutuhkan. |

**Gambar II.21 : Diagram Use case**

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 131)

## E. Communication Diagram

Diagram komunikasi mengelompokkan message pada kumpulan diagram sekuen menjadi sebuah diagram. Dalam diagram komunikasi yang dituliskan adalah operasi / metode yang di jalankan antara objek yang satu dengan objek lainnya secara keseluruhan, oleh karna itu dapat di ambil dari jalanya interaksi pada semua diagram sekuen. Berikut adalah gambar II.21 yang menerangkan simbol-simbol yang ada pada diagram komunikasi :

| Simbol | Deskripsi |
|---|---|
| Objek   | Objek yang melakukan interaksi pesan |
| Link  | Relasi antar objek yang menghubungkan objek satu dengan lainya atau dengan dirinya sendiri   |
| Arah pesan / stimulus  | Arah pesan yang terjadi, jika pada suatu link ada dua arah pesan yang berbeda, maka arah juga deigambarkan dua arah pada dua sisi link  |

**Gambar II.22 : Diagram Komunikasi**

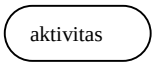
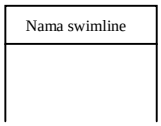
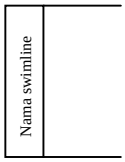
(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 140)

## F. *Activity Diagram*

Diagram aktivitas atau activity diagram menggambarkan workflow (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Diagram aktivitas juga banyak digunakan untuk mendefenisikan hal-hal berikut :

1) Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sitemyang didefenisikan
2) Urutan atau pengelompokan tampilan dari sistem/user interface dimana setiap aktivitasdianggap memiliki sebuah rancangan antarmuka tampilan
3) Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefenisikan kasus ujinya.

Berikut adalah gambar II.23 yang menggambarkan simbol-simbol yang ada pada diagram aktivitas :

| Simbol | Deskripsi |
|---|---|
| Status awal  | Status awal aktivitas sistem, sebuah diagram aktivitas memiliki status awal |
| Aktivitas   | Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja |
| Percabangan / decesion  | Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu |
| Penggabungan / join  | Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu |
| Status akhir  | Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir |
| Swimlane   atau   | Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi |

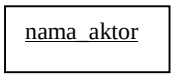
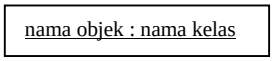
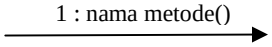
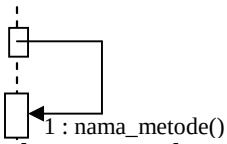
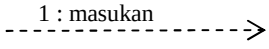
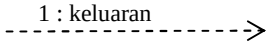
**Gambar II.23 : Diagram Aktivitas**

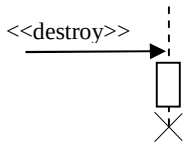
(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 134)

## G. Sequence Diagram

Diagram sekuen menggambarkan kelakuan objek pada use case dengan mendeskripsikan waktu hidup objek dan messasge yang dikirimkan dan diterima antar objek. Banyaknya diagram objek yang digambarkan adalah sebanyak pendefinisian use case yang memiliki proses sendiri atau yang penting semua use case yang telah didefenisikan interaksi jalanya pesan sudah dicakup dapa diagram sekuen sehingga semakin banyak use case yang didefenisikan maka diagram sekuen yang harus dibuat juga semakin

banyak.Berikut adalah gambar II.24 yang menerangkan simbol-sombol yang ada pada diagram sekuen :

| Simbol | Deskripsi |
|---|---|
| Aktor   atau   tampa waktu aktif | Orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang, biasanya di nyatakan menggunakan kata benda di awali frase nama aktor |
| Garis hidup / lifeline  | Menyatakan kehidupan suatu objek |
| Objek   | Menyatakan objek yang berintaraksi pesan |
| Waktu aktif  | Menyatakan objek dalam keadaan aktif dan berinteraksi pesan |
| Pesan tipe create   | Objek yang lain, arah panah mengarah pada objek yang dibuat |
| Pesan tope call   | Menyatakan suatu objek memanggil operasi / metode yang ada pada objek lain atau dirinya sendiri   Arah panah mengarah pada objek yang memiliki operasi / metode, karena ini memanggil operasi / metode maka operasi / metode yang di panggil harus ada pada diagram kelas sesuai dengan kelas objek yang berinteraksi |
| Pesan tipe send   | Menyatakan bahwa suatu objek mengirimkan data / masukan / informasi ke objek lainya, arah panah mengarah pada objek yang dikirimi |
| Pesan tipe return   | Menyatakan bahwa suatu objek yang telah menjalankan suatu operasiatau metode menghasilkan suatu kembalian ke objek |

| | |
|---|---|
| | tertentu, arah panah mengarah pada objek yang menerima kembalian |
| Pesan tipe destroy   | Menyatakan suatu objek mengakhiri hidup objek yang lain, arah panah mengarah pada objek yang diakhiri, sebaiknya jika ada create maka ada destroy |

**Gambar II.24 : Diagram Squence**

(Sumber : Rosa A.S & M. Shalahuddin ; 2011 : 138)

## II.7. Teori Tentang Sekolah

Kata sekolah berasal dari Bahasa Latin yaitu: *skhole, scola, scolae* atau *skhola* yang memiliki arti: waktu luang atau waktu senggang, dimana ketika itu sekolah adalah kegiatan di waktu luang bagi anak-anak di tengah-tengah kegiatan utama mereka, yaitu bermain dan menghabiskan waktu untuk menikmati masa anak-anak dan remaja. Kegiatan dalam waktu luang itu adalah mempelajari cara berhitung, cara membaca huruf dan mengenal tentang moral (budi pekerti) dan estetika (seni).

Untuk mendampingi dalam kegiatan *scola* anak-anak didampingi oleh orang ahli dan mengerti tentang psikologi anak, sehingga memberikan kesempatan yang sebesar-besarnya kepada anak untuk menciptakan sendiri dunianya melalui berbagai pelajaran di atas. Namun saat ini kata sekolah telah berubah arti menjadi suatu bangunan atau lembaga untuk belajar dan mengajar serta tempat menerima dan memberi pelajaran. Sekolah dipimpin oleh seorang Kepala Sekolah. Kepala sekolah dibantu oleh wakil kepala sekolah. Jumlah wakil kepala sekolah di setiap

sekolah berbeda-beda tergantung dengan kebutuhannya. Bangunan sekolah disusun meninggi untuk memanfaatkan tanah yang tersedia dan dapat diisi dengan fasilitas yang lain. Ketersediaan sarana dalam suatu sekolah mempunyai peran penting dalam terlaksananya proses pendidikan.

Ukuran dan jenis sekolah bervariasi tergantung dari sumber daya dan tujuan penyelenggara pendidikan. Sebuah sekolah mungkin sangat sederhana di mana sebuah lokasi tempat bertemu seorang pengajar dan beberapa peserta didik, atau mungkin, sebuah kompleks bangunan besar dengan ratusan ruang dengan puluhan ribu tenaga kependidikan dan peserta didiknya. Berikut ini adalah sarana prasarana yang sering ditemui pada institusi yang ada di Indonesia, berdasarkan kegunaannya:

1. Ruang Belajar

Ruang belajar adalah suatu ruangan tempat kegiatan belajar mengajar dilangsungkan. Ruang belajar terdiri dari beberapa jenis sesuai fungsinya yaitu:

a. Ruang kelas atau ruang Tatap Muka, ruang ini berfungsi sebagai ruangan tempat siswa menerima pelajaran melalui proses interaktif antara peserta didik dengan pendidik, ruang belajar terdiri dari berbagai ukuran, dan fungsi.Sistem kelas terbagi 2 jenis yaitu kelas berpindah (moving class) dan kelas tetap.
b. Ruang Praktik/Laboratorium ruang yang berfungsi sebagai ruang tempat peserta didik menggali ilmu pengetahuan dan meningkatkan keahlian melalui praktik, latihan, penelitian, percobaan. Ruang ini mempunyai kekhususan dan diberi nama sesuai kekhususannya tersebut, diantaranya:

1) Laboratorium Fisika/Kimia/Biologi,
2) Laboratorium bahasa,
3) Laboratorium komputer,
4) Ruang keterampilan, dll

2. Ruang Kantor

Ruang kantor adalah suatu tempat dimana tenaga kependidikan melakukan proses administrasi sekolah tersebut, pada institusi yang lebih besar ruang kantor merupakan sebuah gedung yang terpisah.

3. Perpustakaan

Sebagai satu institusi yang bergerak dalam bidang keilmuan, maka keberadaan perpustakaan sangat penting. Untuk meminjam buku, murid terlebih dahulu harus mempunyai kartu peminjaman agar dapat meminjam sebuah buku.

4. Halaman / Lapangan

Merupakan area umum yang mempunyai berbagai fungsi diantaranya:

a. tempat upacara
b. tempat olahraga
c. tempat kegiatan luar ruangan
d. tempat latihan
e. tempat bermain/beristirahat

5. Lain-lain

Kantin/cafetaria

a. Ruang organisasi peserta didik (OSIS, Pramuka, Senat Mahasiswa, dll)

b. Ruang Komite
c. Ruang keamanan
d. Ruang produksi, penyiaran dll.
e. Ruang Unit Kesehatan Sekolah (UKS)

Di Indonesia, sekolah menurut statusnya dibagi menjadi 2 macam yaitu:

1. Sekolah negeri, yaitu sekolah yang diselenggarakan oleh pemerintah, mulai dari sekolah dasar, sekolah menengah pertama, sekolah menengah atas, dan perguruan tinggi.
2. Sekolah swasta, yaitu sekolah yang diselenggarakan oleh non-pemerintah/swasta, penyelenggara berupa badan berupa yayasan pendidikan yang sampai saat ini badan hukum penyelenggara pendidikan masih berupa rancangan peraturan pemerintah.