BAB II
TINJAUAN PUSTAKA

Bab ini menjelaskan tentang tinjauan pustaka yang terdiri atas kajian pustaka dan dasar teori. Kajian pustaka menjelaskan tentang sistem yang akan dibuat, yaitu Sistem Informasi Geografis dan Manajemen *Traffic Light* Kota Malang. Sedangkan dasar teori menjelaskan teori tentang *Traffic Light*, data spasial, aplikasi pengolah data spasial ArcView, teori basis data, MySQL, konsep pemrograman Visual Basic dan komponen pendukungnya yaitu MapObjects serta pemodelan sistem perangkat lunak.

2.1 Sistem Informasi Geografis dan Manajemen *Traffic Light*

Pendekatan-pendekatan kelokasian atau lebih dikenal dengan istilah pendekatan keruangan atau spasial merupakan hal yang sangat penting di dalam melakukan analisis-analisis fenomena yang terjadi di bumi ini. Fenomena yang terjadi ada yang sifatnya fisik maupun yang bersifat sosial kemasyarakatan seperti ekonomi, politik, lingkungan, budaya, dan sebagainya. Jika fenomena tersebut bisa ditangkap informasinya secara utuh berikut lokasi dan polanya, hal tersebut bisa membantu dalam menyelesaikan atau mencari solusi dari permasalahan terkait muka bumi.

Tentunya untuk mendapatkan informasi yang utuh tersebut diperlukan satu metode yang tepat dan akurat. Metode tersebut antara lain adalah pemetaan, yang bisa menggambarkan obyek-obyek di muka bumi ini ke dalam media yang lebih kecil sehingga lebih mudah dipahami. Namun pemetaan saja belumlah cukup. Data-data atau informasi lainnya yang terkait perlu digambarkan atau tersaji secara lokasional pula sehingga peta dan data (*database*) nya perlu dihubungkan dengan alat yang tepat. Alat tersebut adalah *Geographical Information System* (GIS) atau Sistem Informasi Geografis (SIG). Selain itu, SIG juga diperlukan karena penanganan data spasial yang sulit, terutama karena peta dan data statistik cepat kadaluarsa sehingga pelayanan penyediaan data dan informasi yang diberikan menjadi tidak akurat. Oleh karena itu, SIG diharapkan mampu memberikan kemudahan dalam revisi dan pemutakhiran data [HUS-06:3].

Sistem Informasi Geografis (SIG) merupakan suatu sistem informasi yang dapat memadukan antara data grafis dengan data teks (atribut) yang dihubungkan secara geografis di bumi *(georeference)*. Selain itu, Sistem Informasi Geografis ini juga dapat menggabungkan data, mengatur data dan menganalisis data untuk selanjutnya menghasilkan *output* yang dapat dijadikan acuan dalam pengambilan keputusan [SIA-02:1]. Secara rinci SIG tersebut dapat beroperasi membutuhkan komponen-komponen sebagai berikut [LON-05:22]:

- Orang

  SIG tidak akan menghasilkan banyak manfaat tanpa orang yang melakukan desain, pemrograman, pemeliharaan, menyediakan data dan menginterpretasikan hasilnya. Orang-orang ini memiliki kemampuan yang bervariasi tergantung peran yang mereka lakukan.

- *Software*

  Perangkat lunak (*software*) SIG adalah program komputer yang dibuat khusus dan memiliki kemampuan pengelolaan, penyimpanan, pemrosesan, analisis dan penayangan data spasial. Adapun merk perangkat lunak ini cukup beragam, misalnya Arc/Info, ArcView, ArcGIS dari vendor ESRI, Map Info dari vendor MapInfo Corp. dan masih banyak lagi.

- *Hardware*

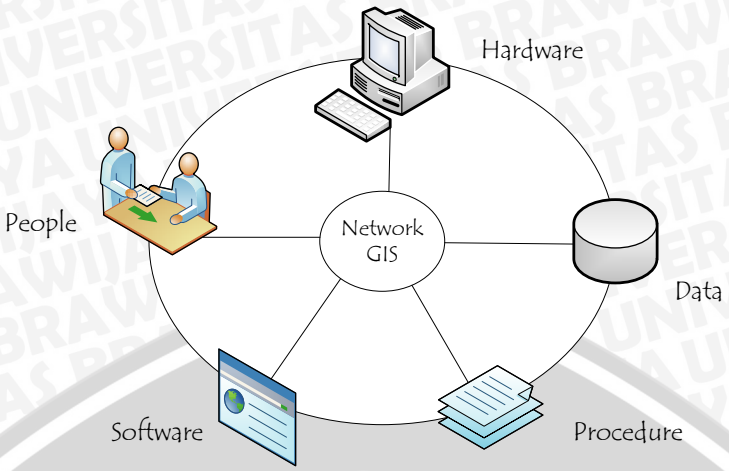
  Perangkat keras *(hardware)* ini berupa seperangkat komputer yang dapat mendukung pengoperasian perangkat lunak yang dipergunakan.

- Data

  Data yang digunakan dalam SIG dapat berupa data grafis dan data atributnya. Kumpulan data-data dalam jumlah besar dapat disusun menjadi sebuah basis data. Basis data SIG bisa berukuran dari *megabyte* sampai *petabyte*.

- Prosedur

  Setiap organisasi harus membuat prosedur, pengontrolan dan mekanisme lainnya untuk memastikan aktivitas SIG berjalan sesuai dengan kebutuhan.



Gambar 2.1 Komponen SIG
Sumber: [LON-05:24]

Dalam SIG terdapat berbagai peran dari berbagai unsur, baik manusia sebagai ahli dan sekaligus operator, perangkat alat (lunak / keras) maupun objek permasalahan [BUD-02:3]. SIG adalah sebuah rangkaian sistem yang memanfaatkan teknologi digital untuk melakukan analisis spasial. Sistem ini memanfaatkan perangkat keras dan lunak komputer untuk melakukan pengolahan data seperti :

1. Perolehan dan verifikasi
2. Kompilasi
3. Penyimpanan
4. Pembaruan dan perubahan
5. Manajemen dan pertukaran
6. Manipulasi
7. Penyajian
8. Analisis

Berdasarkan berbagai kemampuan SIG dalam pengolahan data, tentunya SIG akan memberikan banyak kemudahan jika diterapkan dalam berbagai bidang, termasuk dalam bidang manajemen transportasi. Skripsi ini akan dikhususkan pada pembuatan sistem yang ditujukan untuk inventarisasi data lampu lalu lintas *(traffic light)* yang ada di kota Malang. Lampu lalu lintas didefinisikan sebagai semua peralatan lalu lintas yang menggunakan tenaga listrik untuk mengarahkan atau memperingatkan pengemudi kendaraan bermotor, pengendara sepeda atau

pejalan kaki [OGL-88:391]. Beberapa fungsi dari pemasangan lampu lalu lintas, antara lain:

1. Mendapatkan gerakan lalu lintas yang teratur.
2. Meningkatkan kapasitas lalu lintas pada perempatan jalan.
3. Mengurangi frekuensi jenis kecelakaan tertentu.

Sistem ini tidak hanya mengolah data-data spasial dan non-spasial saja, akan tetapi sistem juga mendukung kegiatan yang bersifat manajerial seperti perawatan dan perbaikan *traffic light*. Oleh karena itu sistem ini juga disebut dengan Sistem Informasi Manajemen. Sistem Informasi Manajemen (SIM) menyediakan informasi yang digunakan dalam perencanaan, pengevaluasian, dan perbaikan berkelanjutan. Selain itu, SIM juga menyediakan informasi yang dapat digunakan sebagai pendukung dalam pengambilan keputusan [WIK-09].

2.2 *Traffic Light*

Untuk sebagaian besar fasilitas jalan, kapasitas dan perilaku lalu lintas terutama adalah fungsi dari keadaan geometrik dan tuntunan lalu lintas. Dengan menggunakan sinyal, perancang atau insinyur dapat mendistribusikan kapasitas kepada berbagai pendekat melalui pengalokasian waktu hijau pada masing-masing pendekat. Maka dari itu untuk menghitung kapasitas dan perilaku lalu lintas, pertama-tama perlu ditentukan fase dan waktu sinyal yang paling sesuai untuk kondisi yang ditinjau [DIR-97:2].

Penggunaan sinyal dengan lampu tiga warna (hijau, kuning, merah) diterapkan untuk memisahkan lintasan dari gerakan-gerakan lalu lintas yang saling bertentangan dalam dimensi waktu. Hal ini adalah keperluan yang mutlak bagi gerakan-gerakan lalu lintas yang datang dari jalan yang saling berpotongan = konflik-konflik utama. Sinyal-sinyal juga dapat digunakan untuk memisahkan gerakan membelok dari lalu lintas lurus melawan, atau untuk memisahkan gerakan lalu lintas membelok dari pejalan kaki yang menyeberang = konflik-konflik kedua.

Jika hanya konflik-konflik primer yang dipisahkan, maka adalah mungkin untuk mengatur sinyal lampu lalu lintas hanya dengan dua fase, masing-masing sebuah untuk jalan yang berpotongan. Metode ini selalu dapat diterapkan jika gerakan belok kanan dalam suatu simpang telah dilarang. Karena pengaturan dua

fase memberikan kapasitas tertinggi dalam beberapa kejadian, maka pengaturan tersebut disarankan sebagai dasar dalam kebanyakan analisa lampu lalu lintas.

2.2.1 Prinsip Umum

Metodologi untuk analisa simpang bersinyal didasarkan pada prinsip-prinsip utama sebagai berikut :

1. Geometri

   Perhitungan dikerjakan secara terpisah untuk setiap pendekat. Satu lengan simpang dapat terdiri dari satu pendekat, yaitu dipisahkan menjadi dua atau lebih sub-pendekat. Hal ini terjadi jika gerakan belok kanan dan/atau belok kiri mendapat sinyal hijau pada fase yang berlainan dengan lalu lintas yang lurus, atau jika dipisahkan secara fisik dengan pulau-pulau lalu lintas dalam pendekat.

2. Arus lalu lintas

   Perhitungan dilakukan per satuan jam untuk satu atau lebih periode, misalnya didasarkan pada kondisi arus lalu lintas rencana jam puncak pagi, siang dan sore.

3. Model dasar

   Permulaan arus berangkat menyebabkan terjadinya apa yang disebut sebagai ‘kehilangan awal’ dari waktu hijau efektif, arus berangkat setelah akhir waktu hijau menyebabkan suatu ‘tambahan akhir’ dari waktu hijau efektif. Jadi besarnya waktu hijau efektif, yaitu lamanya waktu hijau dimana arus berangkat terjadi dengan besaran tetap sebesar S, dapat kemudian dihitung sebagai: Waktu hijau efektif = Tampilan waktu hijau + Kehilangan awal + Tambahan akhir.

4. Penentuan waktu sinyal

   Penentuan waktu sinyal untuk keadaan dengan kendali waktu tetap dilakukan berdasarkan metodea Webster (1966) untuk meminimumkan tundaan total pada suatu simpang.

5. Kapasitas dan derajat kejenuhan

6. Perilaku lalu lintas (kualitas lalu lintas)

   Berbagai ukuran perilaku lalu lintas dapat ditentukan berdasarkan pada arus lalu lintas (Q), derajat kejenuhan (DS), dan waktu sinyal (c dan g).

2.2.2 Fungsi *Traffic Light*

Pada umumnya *traffic light* (sinyal lalu lintas) digunakan dengan satu atau lebih alasan berikut ini :

- Untuk menghindari kemacetan sebuah simpang oleh arus lalu lintas yang berlawanan, sehingga kapasitas simpang dapat dipertahankan selama keadaan lalu lintas puncak.
- Untuk mengurangi jumlah kecelakaan lalu lintas yang disebabkan oleh tabrakan antara kendaran-kendaraan yang berlawanan arah. Pemasangan sinyal lalu lintas dengan alasan keselamatan lalu lintas umumnya diperlukan bila kecepatan kendaraan yang mendekati simpang sangat tinggi dan/atau jarak pandang terhadap gerakan lalu lintas yang berlawanan tidak memadai yang disebabkan oleh bangunan-bangunan atau tumbuhan-tumbuhan yang dekat pada sudut-sudut simpang.
- Untuk mempermudah menyeberangi jalan utama bagi kendaraan dan/atau pejalan kaki dari jalan minor.

2.3 Data Spasial

Data spasial mempunyai pengertian sebagai suatu data yang mengacu pada posisi, obyek, dan hubungan diantaranya dalam ruang bumi. Pemanfaatan data spasial semakin meningkat setelah adanya teknologi pemetaan digital dan pemanfaatannya pada Sistem Informasi Geografis (SIG). Data spasial mempunyai dua bagian penting yang membuatnya berbeda dari data lain, yaitu informasi lokasi dan informasi atribut yang dapat dijelaskan sebagai berikut [TAR-03:8]:

- Informasi lokasi atau informasi spasial. Contoh yang umum adalah informasi lintang dan bujur, termasuk diantaranya informasi datum dan proyeksi. Contoh lain dari informasi spasial yang bisa digunakan untuk mengidentifikasikan lokasi misalnya adalah Kode Pos.
- Informasi deskriptif (atribut) atau informasi non spasial. Suatu lokalitas bisa mempunyai beberapa atribut atau properti yang berkaitan dengannya; contohnya jenis vegetasi, populasi, pendapatan per tahun dan sebagainya.

Data non-spasial yang berupa data teks maupun statistik yang merupakan data atribut dari data spasial disimpan pada basis data non-spasial. Selanjutnya

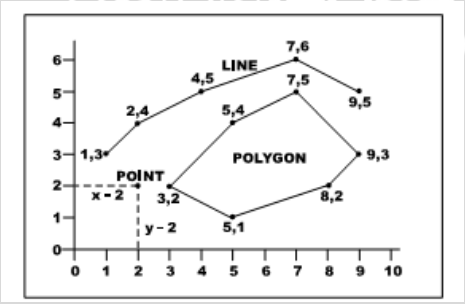
data spasial dan data non-spasial diintegrasikan pada sistem manajemen basis data yang akan menghasilkan suatu aplikasi Sistem Informasi Geografis.

Bentuk representasi *entity* spasial adalah konsep vektor dan raster. Dengan demikian, data spasial direpresentasikan di dalam basis data sebagai vektor atau raster, sehingga untuk menyajikan *entity* spasial digunakan model data raster atau vektor [ANO-10:1].

2.3.1 Struktur Data Vektor

Model data vektor menampilkan, menempatkan, dan menyimpan data spasial dengan menggunakan titik, garis (kurva atau poligon) beserta atributnya. Bentuk dasar representasi data spasial dalam model data vektor didefinisikan oleh sistem koordinat kartesian dua dimensi (x,y) [ANO-10:5]. Keuntungan utama dari format data vektor adalah ketepatan dalam merepresentasikan fitur titik, batasan dan garis lurus. Hal ini sangat berguna untuk analisa yang membutuhkan ketepatan posisi. Kelemahan data vektor yang utama adalah ketidakmampuannya dalam mengakomodasi perubahan gradual [TAR-03:9].

Pada data format vektor, bumi kita direpresentasikan sebagai suatu mosaik dari garis (*arc/line*), polygon (daerah yang dibatasi oleh garis yang berawal dan berakhir pada titik yang sama), titik/*point* (node yang mempunyai label), dan *nodes* (merupakan titik perpotongan antara dua buah garis) dari beberapa fitur.



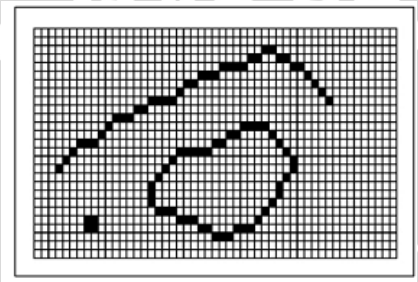
Gambar 2.2 Model data vektor
Sumber: [TAR-03:9]

2.3.2 Struktur Data Raster

Model data raster menampilkan, menempatkan, dan menyimpan data spasial dengan menggunakan struktur matriks atau piksel-piksel yang membentuk *grid*. Setiap piksel memiliki atribut tersendiri, termasuk koordinatnya yang unik (di pojok, pusat, atau ditempat lain dalam *grid*). Entitas spasial raster di dalam

layer secara fungsionalitas direlasikan dengan unsur-unsur petanya. Akurasi model ini sangat tergantung pada resolusi atau ukuran pikselnya di permukaan bumi. Contoh unsur spasial raster adalah citra *satellite* (NOAA, *spot*, Landsat, Ikonos, dll), citra radar, dan model ketinggian digital (DTM) [ANO-10:1].

Data raster sangat baik untuk merepresentasikan data-data yang bersifat *continous* / gradual seperti dalam memodelkan permukaan bumi [TAR-03:9]. Misalnya jenis tanah, kelembaban tanah, vegetasi, suhu tanah dan sebagainya. Keterbatasan utama dari data raster adalah besarnya ukuran file, semakin tinggi resolusi *grid*-nya semakin besar pula ukuran filenya.



Gambar 2.3 Model data raster
Sumber: [TAR-03:9]

2.3.3 Sistem Pemasukan Data

Sistem Informasi Geografis (SIG) membutuhkan masukan data yang bersifat spasial maupun deskriptif [TAR-03:10]. Beberapa sumber data tersebut antara lain peta analog, data dari sistem penginderaan jauh, data hasil pengukuran dan data GPS. Teknik memasukkan data spasial dari sumber-sumber tersebut ke dalam SIG, terbagi menjadi 3 macam, antara lain :

1. Digitasi

   Metode digitasi dapat dilakukan secara manual dengan alat digitizer atau menggunakan perangkat lunak dengan teknik digitasi *on screen*. Perangkat lunak yang dapat digunakan untuk digitasi ini, misalnya Auto CAD, Arc Info, R2V, dan lain-lain [BUD-02:21]. Cara kerjanya adalah dengan mengkonversi fitur-fitur spasial yang ada pada peta menjadi kumpulan koordinat x,y. Untuk menghasilkan data yang akurat, dibutuhkan sumber peta analog dengan kualitas tinggi. Proses digitasi memerlukan ketelitian dan konsentrasi tinggi dari operator.

2. Penggunaan GPS

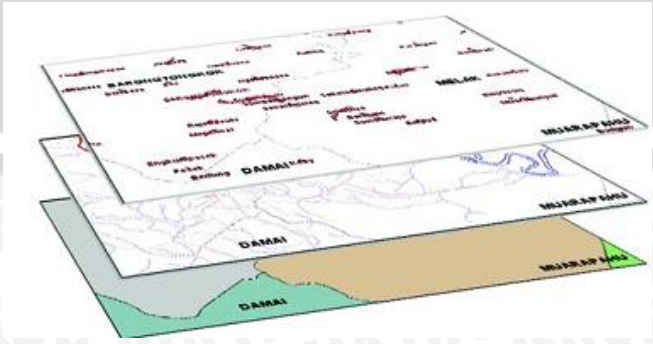
   Data spasial lain dalam bentuk digital seperti data hasil pengukuran lapangan dan data dari GPS bisa dimasukkan dalam sistem SIG [TAR-03:31]. Sebenarnya GPS adalah suatu sistem yang dapat membantu kita mengetahui posisi koordinat dimana kita berada. Sedangkan untuk menerima sinyal yang dipancarkan oleh GPS, kita membutuhkan suatu alat yang dapat membaca sinyal tersebut. Yang biasa kita sebut sebagai GPS sebenarnya merupakan alat penerima. Karena alat ini dapat memberikan nilai koordinat dimana ia digunakan maka keberadaan GPS merupakan terobosan besar bagi SIG.

3. Konversi dari sistem lain

   Salah satu teknik pemasukan data ke dalam SIG yaitu dengan menggunakan lajur elektronik (*spreadsheet*). Misalnya perangkat lunak Microsoft Excel yang digunakan untuk mengolah data dari survei lapangan yang akan dipakai sebagai referensi bagi klasifikasi citra satelit. Data juga dapat diperoleh dari sistem lain berupa file berekstensi .dbf. Format file tersebut berasal dari sistem dBase III+ dan dBase IV [BUD-02:69].

2.3.4 Sistem Tampilan Data

Sistem tampilan data pada SIG menggunakan konsep *layer* data dan atribut. Konsep *layer* data adalah representasi data spasial menjadi sekumpulan peta tematik yang berdiri sendiri-sendiri sesuai dengan tema masing-masing, tetapi terikat dalam suatu kesamaan lokasi. Keuntungan dari konsep data *layer* adalah mudahnya proses penelusuran dan analisa spasial serta efisiensi pengelolaan data. Untuk menampilkan data spasial yang telah di digitasi bisa menggunakan perangkat lunak ArcView.



Gambar 2.4 Konsep *layer*
Sumber: [TAR-03:37]

2.4 *Shapefile*

Pada umumnya peta digital dibuat dalam bentuk *shapefile*. *Shapefile* merupakan bentuk format penyimpanan vektor digital (*non-topological*) untuk menyimpan posisi geometri dan informasi dari data spasialnya [PET-09]. *Shapefile* terdiri dari 3 *file* utama yaitu *main file* ".shp", *index file* ".shx", dan *dBASE file* ".dbf". Adapun data yang disimpan pada *main file* berupa berapa banyak *record* yang ada, dan ukuran dari masing-masing *record.* Setiap *record* mendeskripsikan sebuah *shape* (titik, garis, atau poligon) beserta *vertex* yang bersangkutan. Pada *index file* tersimpan alamat *offset* dari setiap *record* yang ada (pada *main file* secara berurutan). dBASE *file* berisi informasi dari setiap *record* yang ada.

Ketiga ekstensi *file* tersebut merupakan *file* utama yang dibutuhkan untuk membangun sebuah *Shapefile*. Selain itu juga ada beberapa ekstensi *file* lain yang juga dapat ditambahkan yang berupa *file-file* pelengkap pilihan, antara lain :

- ".sbn" dan ".sbx" yaitu *file* yang menyimpan *spatial index* dari *feature*.
- ".fbn" dan ".fbx" yaitu *file* yang menyimpan *spatial index* dari *feature* untuk *Shapefile* dan bersifat *read-only*.
- ".ain" dan ".aih" yaitu *file* yang menyimpan *attribute index* dari bagian yang aktif dalam sebuah tabel.
- ".prj" yaitu *file* yang menyimpan koordinat dari sistem informasi.

Secara normal, keseluruhan data yang dimiliki oleh sebuah *shapefile* akan dibaca sebagai sebuah *dataset*, dan masing-masing ekstensi *file* yang terdapat di dalamnya akan dibaca sebagai sebuah *layer*. Dalam hal ini, maka nama dari *directory* harus digunakan juga sebagai nama dari *dataset*-nya.

2.5 ArcView

ArcView merupakan salah satu perangkat lunak desktop GIS dan pemetaan yang dikembangkan oleh ESRI (*Environmental System Research Institute,* Inc). Aplikasi ini memiliki berbagai keunggulan yang dapat dimanfaatkan oleh kalangan pengolah data spasial. ArcView memiliki kemampuan-kemampuan untuk melakukan visualisasi, meng-*explore*, menjawab *query* (baik basis data spasial maupun basis data non-spasial), menganalisis data secara geografis dan sebagainya [GUN-04:1].

Model data pada ArcView terdiri dari data spasial dan data non-spasial (atribut). Data atribut biasanya berbentuk matriks atau tabular dimana informasi untuk tiap obyek geografis disimpan kedalam *record*. Semua *record* dikombinasikan menghasilkan tabel data atribut yang berisi informasi semua obyek geografis yang terdapat pada *layer* SIG. Pada tiap *record*, terdapat *field* atribut. Tiap *field* atribut mempunyai atribut item yang menunjukkan nilai *field* atribut pada *record*. Paket SIG mengatur hubungan antara data spasial dan data atribut. Hubungan ini biasanya bisa dilihat oleh user. Secara khas, tiap obyek geografis menugaskan fitur pengenal yang biasanya tersedia pada *record* yang sesuai pada tabel atribut [WON-05:30].

ArcView dapat menerima berbagai macam sumber data yang selanjutnya akan diolah [BUD-02:14]. Secara langsung ArcView dapat menerima data vektor yang berasal dari *software* Arc Info. Data vektor olahan ini dapat lebih jauh diolah atau langsung disajikan dalam *layou*t. Sumber-sumber data lain adalah yang berasal dari :

- Citra satelit dengan format BSQ, BIL, BIP
- Data raster dengan format BMP, JPG, TIFF
- Data CAD
- Data ERDAS
- Data tabular dari Arc Info, dBase

2.6 Basis Data

Basis data (*database*) merupakan mekanisme yang digunakan untuk menyimpan informasi atau data. Informasi adalah sesuatu yang kita gunakan sehari-hari untuk berbagai alasan. Dengan basis data, pengguna dapat menyimpan data secara terorganisasi. Setelah data disimpan, informasi harus mudah diambil. Cara data disimpan dalam basis data menentukan seberapa mudah mencari informasi berdasarkan banyak kriteria. Data pun harus mudah ditambahkan, dimodifikasi dan dihapus ke dalam basis data [SIM-06:1].

2.6.1 *Database Management System* (DBMS)

Database Management System (DBMS) merupakan perangkat lunak yang dirancang untuk dapat melakukan utilitas dan mengelola koleksi data dalam jumlah yang besar. DBMS juga dirancang untuk dapat melakukan manipulasi data

secara lebih mudah. DBMS merupakan antar muka pengguna *database* (baik pengguna langsung maupun aplikasi) dengan data yang tersimpan. Beberapa contoh DBMS adalah MySQL, DB2, Oracle, SQL Server dan lain-lain. Dengan menggunakan DBMS maka pengguna dapat mendefinisikan data yang akan disimpan dalam suatu bentuk yang dinamakan dengan model data.

Relational Database Management System (RDBMS) adalah tipe basis data yang paling popular digunakan saat ini. Dalam model data relasional, *database* dapat dikatakan sebagai kumpulan satu atau lebih relasi dimana setiap relasi merupakan koleksi dari data disajikan dalam betuk tabel yang terdiri dari baris dan kolom. Relasi merupakan konstruksi deskripsi data yang utama dalam model relasional. Relasi dapat dibayangkan sebagai kumpulan *record (tuple)* dimana setiap *record* mempunyai jumlah *field* yang sama serta domain yang bersesuaian sama. Dalam *database* relasional model *Entity Relationship* (ER) merupakan model semantik yang banyak digunakan untuk mendeskripsikan entitas dan relasi yang menyertainya [UTA-08:12].

2.6.2 Kunci

Kunci (*key*) merupakan suatu atribut yang unik yang dapat digunakan untuk membedakan suatu entitas dengan entitas yang lain dalam suatu himpunan entitas. Ada 4 macam *key* yang dapat diterapkan pada suatu tabel dalam model basis data relasional, antara lain :

a. Superkey

Merupakan satu atau lebih atribut (kumpulan atribut) yang dapat membedakan setiap baris data dalam sebuah tabel secara unik.

b. Candidate-Key

Merupakan kumpulan atribut minimal yang dapat membedakan setiap baris data dalam sebuah tabel secara unik.

c. Primary-Key

Merupakan *candidate-key* yang dipilih untuk mengidentifikasi baris data secara unik dalam relasi, dengan didasari pada:

- *Key* tersebut lebih sering (lebih natural) untuk dijadikan sebagai acuan.
- *Key* tersebut lebih ringkas.
- Jaminan keunikan *key* tersebut lebih baik.

d. *Foreign-Key*

Merupakan kunci tamu atau asing dalam suatu tabel dimana kunci ini juga merupakan *primary key* dalam tabel lain yang berelasi.

2.6.3 Normalisasi

Normalisasi merupakan proses pengelompokan elemen data menjadi tabel yang menunjukkan entitas sekaligus relasinya [UTA-08:32]. Tujuan dari normalisasi adalah mencegah terjadinya *insertion anomaly* (kesalahan penambahan data), *delete anomaly* (kesalahan dalam menghapus data) dan *update anomaly* (kesalahan dalam mengubah data). Berikut ini merupakan bentuk-bentuk normalisasi :

1. Bentuk Normalisasi Pertama (1NF)

   Suatu tabel dikatakan dalam bentuk normal pertama apabila:

   a. Tidak ada baris data yang berulang dalam tabel.
   b. Setiap sel memiliki nilai tunggal artinya tidak ada perulangan *group* atau *array*.
   c. Data dalam kolom (atribut dan *field*) memiliki tipe data yang sejenis.

2. Bentuk Normalisasi Kedua (2NF)

   Tabel dalam keadaan 2NF apabila tabel sudah dalam keadaan 1NF dan semua atribut yang bukan kunci bergantung pada semua kunci dalam tabel. Dengan kata lain, 2NF bertujuan untuk menghilangkan ketergantungan parsial.

3. Bentuk Normalisasi Ketiga (3NF)

   Tabel dalam keadaan 3NF apabila tabel dalam keadaan 2NF dan dalam tabel tersebut tidak ada ketergantungan transitif. Artinya sebuah *field* dapat menjadi atribut biasa pada suatu relasi tetapi menjadi kunci pada relasi lain. Setiap atribut yang bukan kunci haruslah hanya bergantung pada *primary key*.

4. Bentuk Normalisasi Boyce-Codd (BCNF)

   Tabel dalam keadaan 3NF dan setiap determinan merupakan kunci kandidat. Determinan adalah satu atribut / *field* atau gabungan atribut dimana beberapa atribut lain bergantung pada atribut tersebut. Pada tahap BCNF, kita harus menghilangkan kunci kandidat yang bukan determinan.

2.7 MySQL

MySQL adalah *software* RDBMS *(Relational Database Management System)* yang bersifat *open source*. *Relational Database* menyimpan data pada table-tabel yang berbeda yang akan lebih baik daripada apabila data disimpan pada ruang penyimpanan yang besar, karena data-data pada tabel lebih cepat diakses dan akan menambah fleksibilitas [SUN:09].

MySQL sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu SQL (*Structured Query Language*). SQL adalah sebuah konsep pengoperasian *database*, terutama untuk pemilihan/seleksi dan pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis [PRA-03:1].

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

2.7.1 Keistimewaan MySQL

1. *Open source*, artinya *source* MySQL dapat diperoleh dengan mudah dan gratis.
2. *Multiuser*, artinya dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.
3. *Scalibility* dan *Limits*, artinya MySQL mampu menangani *database* dalam skala besar, dengan jumlah *record* lebih dari 50 juta dan 60 ribu tabel serta 5 miliar baris. Selain itu, batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.
4. *Portobility*, artinya dapat berjalan stabil pada berbagai sistem operasi.
5. *Performance running*, artinya memiliki kecepatan yang handal.
6. *Security*, artinya MySQL memiliki beberapa lapisan sekuritas seperti level *subnetmask*, nama *host* dan izin akses *user* dengan sistem perizinan yang mendetail serta *password* terenkripsi.
7. MySQL menggunakan suatu bahasa permintaan standar yang bernama SQL (*Structure Query Language*) yaitu sebuah bahasa permintaan yang

distandarkan pada beberapa *database server* seperti Oracle, PostGreSQL, dan lain-lain.

2.8 Visual Basic

Visual Basic adalah salah satu pemrograman berbasis visual yang dikembangkan oleh Microsoft sejak tahun 1991. Visual Basic merupakan pengembangan dari bahasa pemrograman BASIC (*Beginner's All-Purpose Symbolic Instruction Code*) yang dikembangkan pada tahun 1960-an.

Visual Basic menggunakan model *Event-Driven* (program merespon berdasarkan *event* yang diberikan oleh *user*) dan merupakan bahasa pemrograman visual dimana program dibuat menggunakan *Integrated Development Environment* (IDE). Dengan menggunakan IDE, *user* bisa menulis, menjalankan, menguji dan men-*debug* (memperbaiki *error*) program Visual Basic dengan baik. Dengan demikian bisa mengurangi waktu dalam membuat program yang berjalan secara terpisah bila tanpa menggunakan IDE [DEI-06:8].

Visual Basic merupakan salah satu bahasa pemrograman komputer yang mendukung object (*Object Oriented Programming*/OOP). Perbedaan mendasar antar teknik pemrograman BASIC (terstruktur) dengan Visual Basic (*Object Oriented*) adalah fokus dalam rangka memberikan suatu solusi [WAH-02:3]. Dengan kata lain, teknik OOP lebih fokus pada elemen dari suatu problem, sedangkan teknik terstruktur lebih fokus pada metode atau cara untuk mencapai solusi dari problem tersebut. Untuk setiap objek dalam Visual Basic pasti memiliki sebagaian atau semua komponen berikut :

1. Properti

   Suatu properti didefinisikan sebagai elemen dari suatu objek yang bisa diubah, baik secara langsung (melalui kode), maupun tidak langsung (melalui *property explorer*). Properti cenderung untuk tetap, tidak berubah (selama tidak ada tindakan untuk mengubahnya, baik via kode maupun dari *property explorer*), selama aplikasi berjalan. Tidak seperti variabel, yang hanya berlaku selama masa hidup prosedur atau modul tempat variabel tersebut dideklarasikan.

   Sebagai contoh, *Command Button control*. Objek ini memiliki properti seperti *Weight, Height,* dan *Caption.* Setiap *control* pada Visual Basic berasal dari

kelas. Berikut ini adalah contoh mengakses *command button, text box,* dan *label control*.

Command1.Caption = "Test"

Text1.Text = "Visual Basic"

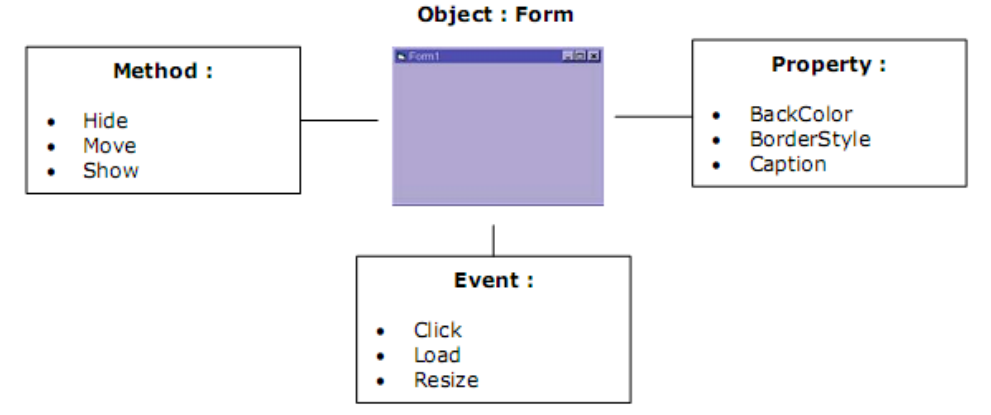
Label1.Caption = " OOP"

2. Metode

   Metode merupakan suatu cara dari *programmer* yang membuat *control* beraksi.

3. *Event*

   Event bisa dikatakan sebagai sesuatu yang terjadi selama program berjalan. Sebagai contoh, ketika *Command Button* di klik, maka akan muncul *event Command_Click*. Tiap *event* yang bisa digabungkan dengan kode disebut *event procedure*. Program dalam Visual Basic dikatakan sebagai *event-driven* karena kode yang menjalankan fungsinya – biasanya – terikat oleh GUI (*Graphical User Interface*), cenderung dipicu oleh berbagai *event*.

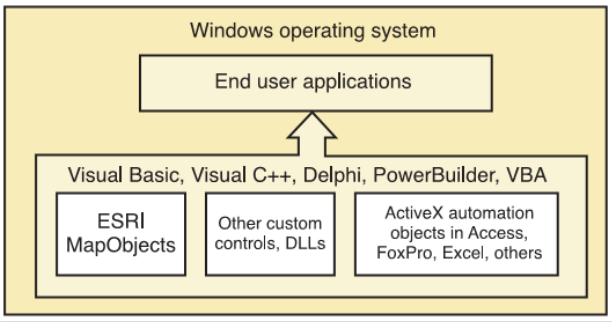


Gambar 2.5 Komponen Visual Basic
Sumber: [OCT-03:37]

2.9 MapObjects

MapObjects merupakan komponen pendukung yang akan digunakan untuk menampilkan peta pada *compiller* yang dipilih. Cara kerja komponen ini sama dengan cara kerja SIG, yaitu menggunakan sistem *layer* untuk menampilkan peta. MapObjects terdiri atas kontrol *ActiveX* yang disebut *Map Control* dan memiliki 46 objek otomasi ActiveX yang digunakan pada lingkungan pemrograman seperti Visual Basic, Visual C++, Delphi, PowerBuilder dan *Visual Basic for Application*

(VBA). Program yang dibangun menggunakan MapObjects berjalan pada sistem operasi Windows.



Gambar 2.6 Konsep MapObjects
Sumber: [ESR-99:15]

ActiveX biasa disebut dengan *Object Linking and Embedding* (OLE). Kontrol otomasi *ActiveX* merupakan komponen perangkat lunak yang akan menambah fungsi pada *container ActiveX* sesuai dengan kebutuhan. Dengan adanya *container ActiveX*, misalnya Visual Basic, kita dapat mengkombinasikan objek dari MapObjects, kontrol lain, maupun objek *ActiveX* dari program lain untuk menciptakan aplikasi *end-user*. *Container ActiveX* bisa mempunyai satu atau lebih kontrol *ActiveX* yang bisa langsung digunakan sesuai kebutuhan [ESR-99:15].

Komponen MapObjects bersifat *ActiveX* sehingga dapat digunakan dalam bahasa pemrograman yang berbeda-beda. Lewat komponen inilah maka peta yang dihasilkan dapat bersifat dinamis. Untuk menampilkan sebuah peta, MapObjects akan menggunakan sistem *layer* dalam menampilkan komponen peta yang terdiri dari beberapa *shapefile*. Setiap *shapefile* yang ditampilkan tersebut akan dikenali sebagai sebuah *layer* [PET-06].

Komponen MapObjects mempunyai kegunaan sebagai berikut:

- Menampilkan *shapefile* ke layar *monitor* dalam bentuk *layer*.
- Mempunyai fungsi *zoom* dan *pan*.
- Memungkinkan untuk dapat meng-*edit record* yang terdapat pada *shapefile*.
- Mengetahui tipe dari sebuah *shapefile*.
- Melakukan pengolahan *database* secara *spatial*.
- Melakukan manipulasi pada *layer* sesuai dengan kriteria yang diharapkan.

- Membuat *shapefile* baru.

MapObjects juga mempunyai banyak kelebihan bila dibandingkan dengan *software* pembuat peta yang lain. Kelebihan dari MapObjects ini antara lain:

- Mempunyai variasi sumber data dan format gambar yang sangat luas.
- Mempunyai optimasi performa yang baik dalam penanganan data.

2.9.1 Prosedur, Fungsi dan Properti MapObjects

Beberapa hal penting mengenai prosedur, fungsi dan properti dari *object* MapObjects yang digunakan dalam membangun aplikasi peta ini adalah sebagai berikut:

1. *Map* : Berfungsi untuk mengontrol tampilan sekumpulan *layer* yang ada. Beberapa properti yang terdapat pada *object Map* ini adalah:
   - *Extent*: Tipe dari properti ini adalah *rectangle*. Berfungsi untuk menentukan besarnya dari sebuah tampilan *layer* peta.
   - *FullExtent*: Sama dengan properti *extent*, hanya saja tampilan yang dihasilkan oleh properti ini merupakan besarnya tampilan peta secara keseluruhan.
2. *Pan*: Berfungsi untuk menggeser tampilan peta sesuai dengan pergerakan *mouse.*
3. *Rectangle*: *Object* ini akan merepresentasikan bentuk segiempat dengan empat tepi dan empat sudut siku-siku. Dimensi segiempat tersebut memberikan nilai tinggi dan lebar serta memiliki sebuah titik pusat. *Object* ini dapat dibuat berdasarkan posisi koordinat keempat titik sudutnya yaitu lewat properti *top, left, bottom, right* yang mempunyai tipe data *double*.
4. *DataConnection*: *Object* ini digunakan untuk menghubungkan ke sumber data geografi, dalam hal ini adalah sebuah *shapefile*. Beberapa properti yang terdapat pada object *Data Connection* ini adalah:
   - *Database*: Memberikan nama *database* yang digunakan untuk mengakses data dari sebuah *shapefile*.

   Berikut ini merupakan *Function* dari *Data Connection*:
   - AddGeoDataset (Name: String; ShapeType: ToleNum: TableDesc: TableDesc; HasZ, HasMeasure: Boolean);

Fungsi ini akan mengembalikan nilai berupa *geodataset* dan menambahkannya ke dalam *object DataConnection*. Penjelasan mengenai parameter yang ada pada fungsi ini adalah:

- *Name*: merupakan nama dari *GeoDataSet* yang akan ditambahkan ke dalam *object DataConnection.*
- *ShapeType*: merepresentasikan tipe dari *GeoDataSet* yang akan ditambahkan, apakah berupa *point*, *line*, atau *poligon*.
- *TableDesc*: menyimpan deskripsi karakteristik dari *field* yang ada pada *recordset*.
- *HasZ*: Bernilai *true*, jika *GeoDataSet* yang akan dibuat dapat menyimpan *shape* yang mempunyai nilai Z. Bernilai *false*, jika sebaliknya
- *HasMeasure*: Bernilai *true*, jika *GeoDataSet* yang akan dibuat dapat menyimpan *shape* yang mempunyai nilai *Measure*. Bernilai *false*, jika sebaliknya.

5. *Recordset*: *Object* ini merepresentasikan semua *record* yang berhubungan dengan *record* yang dihasilkan oleh proses *query*. Beberapa properti yang terdapat pada object *Recordset* ini adalah:
   - *Count*: properti ini akan memberikan nilai banyaknya *record* yang terdapat pada *recordset*.
   - *EOF*: properti ini akan memberitahu apakah posisi kursor pada sebuah *recordset* sudah berada pada *record* yang terakhir.
   - *Fields*: properti ini akan menyimpan nilai dari setiap *field* yang tersimpan pada sebuah *recordset.*
   - *TableDesc*: properti ini akan memberikan penjelasan tentang karakteristik dari *field* yang terdapat pada *recordset.*

   Berikut ini beberapa *Function Recordset:*
   - *Delete*: menghapus *record.*
   - *Edit*: meng-*edit record.*
   - *Update*: mengganti nilai sebuah *record* dengan nilai yang terbaru.
   - *StopEditing*: keluar dari mode *edit.*

6. *MapLayer*: *Object* ini berfungsi untuk menampilkan data *shapefile* peta dalam bentuk *layer* melalui *feature* yang terdapat pada *GeoDataset*. Beberapa properti yang terdapat pada object *MapLayer* ini adalah:
   - *GeoDataset*: berfungsi untuk menampilkan data geografi yang ada kedalam bentuk *layer*. Properti ini juga terhubung dengan properti *MapLayer* untuk dapat membuat tampilan *layer* kedalam peta.
   - *Name*: merupakan nama *layer* yang akan dibuat oleh *GeoDataset*. Nama ini dibuat dan ditentukan sendiri oleh pengguna.
   - *ShapeType*: merupakan properti yang akan mendefinisikan jenis *shape* apa saja yang disediakan oleh *MapObject*. Properti ini berguna agar *MapLayer* dapat mengetahui tipe yang sedang digunakan oleh sebuah *layer*.

   Berikut ini merupakan *Function MapLayer:*
   - *SearchExpression*: *query* pencarian terhadap isi *MapLayer* berdasar kriteria yang diinginkan oleh *user*.
7. *Point*: *object* ini digunakan untuk merepresentasikan sebuah titik yang berupa kordinat x dan y.
8. *Symbol*: *object* ini digunakan untuk menentukan bagaimana sebuah simbol akan ditampilkan dalam sebuah *layer*. Beberapa properti yang terdapat pada *object Symbol* ini adalah:.
   - *Color*: untuk menentukan warna dari sebuah simbol.
   - *Size*: untuk menentukan ukuran sebuah simbol.
   - *Style*: untuk menentukan *style* simbol.
   - *Font*: untuk menentukan jenis *font* yang digunakan.
   - *CharacterIndex*: untuk menentukan kode karakter *font* yang akan digunakan.
9. *Line*: *object* ini merepresentasikan bentuk geometri yang terdiri dari dua atau lebih titik yang saling berhubungan sehinga membentuk sebuah garis.

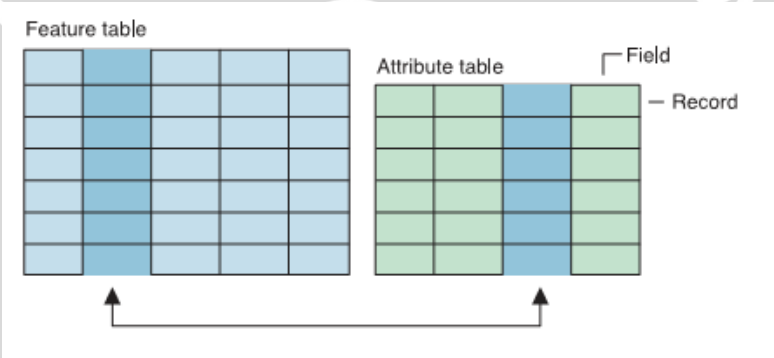
2.9.2 Tabel Atribut

Informasi tabular (non-spasial) direpresentasikan oleh *object tabel* yang bisa berasal dari sumber yang bervariasi, seperti [ESR-99:34]:

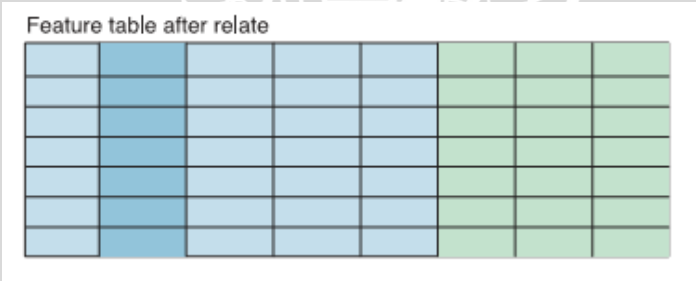
- SDE ESRI
- INFO ESRI

- *Open Database Connectivity* (ODBC)
- Microsoft Data Access Object (DAO)

Data tabular dapat diakses secara langsung atau melalui sebuah relasi. Tabel yang diakses secara langsung menggunakan *query*. Pengaksesan *record* pada tabel atribut akan mengembalikan nilai, nama *field* dan tipe *field*. Jika menggunakan MapObjects kebanyakan orang akan menyimpan data pada sumber lain seperti Microsoft Access yang akan dihubungkan melalui suatu relasi. Relasi merupakan penggabungan tabel antara tabel fitur dan tabel atribut. Tabel fitur bisa berupa *shapefile*, ARC/INFO, *layer* SDE, atau file VPF. Relasi hanya berlangsung selama aplikasi berjalan dan tidak disimpan pada *file system*. Berikut merupakan implementasi dari relasi.



Gambar 2.7 Tabel fitur dan tabel atribut sebelum relasi
Sumber: [ESR-99:34]

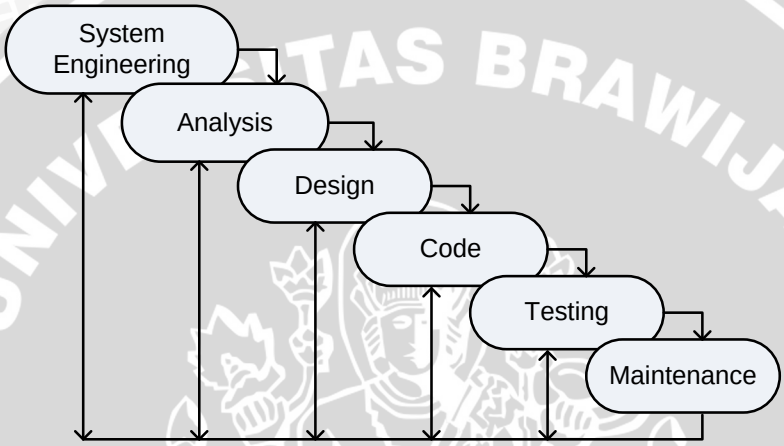


Gambar 2.8 Tabel fitur dan tabel atribut setelah relasi
Sumber: [ESR-99:35]

2.10 Metode Pengembangan Perangkat Lunak

Dalam melakukan proses pengembangan suatu perangkat lunak, diperlukan suatu metode yang digunakan sebagai panduan untuk menghasilkan sebuah perangkat lunak yang benar dan berkualitas. Sehingga perangkat lunak

yang dihasilkan bisa menguntungkan pihak *user* maupun *developer*. *User* bisa mendapatkan perangkat lunak yang sesuai dengan permintaan dan kebutuhan yang diinginkan, dan di pihak *developer* dapat menyelesaikan sebuah perangkat lunak dengan terjadwal, serta mendapatkan kemudahan dalam melakukan *maintenance* (pemeliharaan). Dalam hal inilah urgensi dari sebuah rekayasa terhadap perangkat lunak *(Software Engineering)*. *Software engineering* merupakan teknologi yang meliputi proses, sekumpulan metoda dan sederetan alat bantu untuk pengembangan perangkat lunak [PRE-92:24].



Gambar 2.9 Siklus pengembangan perangkat lunak model *Waterfall*
Sumber: [PRE-92:25]

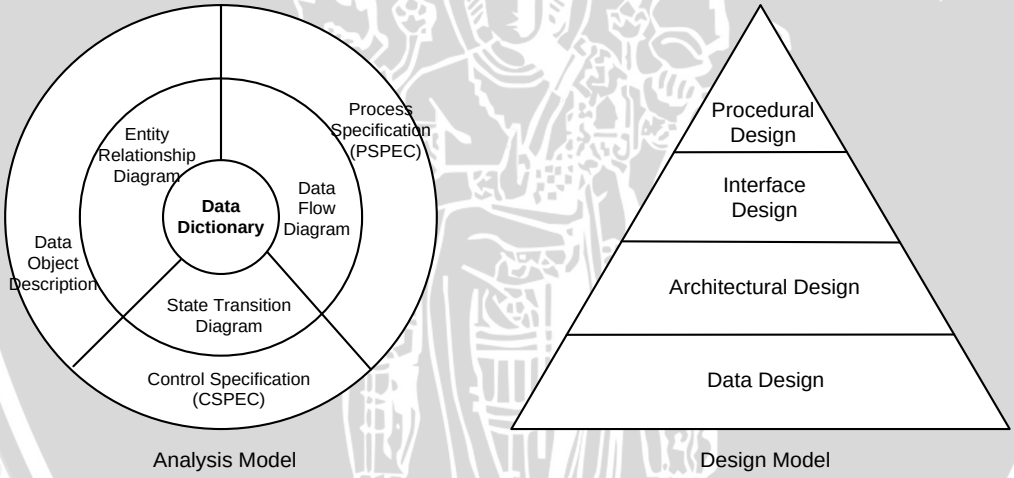
Pada Gambar 2.9 terlihat siklus pengembangan suatu sistem perangkat lunak. Proses pengembangan perangkat lunak akan selalu melakukan proses pengumpulan terhadap kebutuhan *(requirement)* dari *user* (pengguna) atau *customer* (pelanggan), memahami dan menetapkan kebutuhan-kebutuhan tersebut. Langkah ini menjadi suatu pekerjaan yang sangat penting. Tahap yang selanjutnya adalah melakukan analisis terhadap kebutuhan dilanjutkan dengan perancangan sistem perangkat lunak yang akan dibangun. Hasil dari perancangan dijadikan acuan dalam proses *coding* untuk merealisasikannya ke dalam bentuk yang bisa diterjemahkan oleh mesin. Proses selanjutnya adalah pengujian terhadap perangkat lunak yang telah dihasilkan.

2.10.1 Analisis dan Desain Model

Setelah kebutuhan dikumpulkan, analisis terhadap kebutuhan dilakukan dengan menggunakan beberapa alat (*tools*) seperti DFD (*Data Flow Diagram*),

STD (*State Transition Diagram*) dan ERD (*Entity Relationship Diagram)* [PRO-09:1]. *Data Dictionary* menjadi bekal dasar untuk menganalisis kebutuhan. *Data Dictionary* berisi gambaran dari semua objek data yang diperlukan dan dihasilkan oleh *software* nantinya. Diagram-diagram tadi mempunyai karakteristik masing-masing. DFD memberi gambaran bagaimana data berubah sejalan dengan alirannya dalam sistem dan menggambarkan fungsi-fungsi yang mengubah data-data tadi. ERD menggambarkan relasi antara objek data. STD menggambarkan bagaimana kerja sistem melalui kondisi (*state)* dan kejadian yang menyebabkan kondisi berubah. STD juga menggambarkan aksi yang dilakukan karena kejadian tertentu.

Hasil yang diperoleh dari analisis kebutuhan adalah model analisis yang kemudian menjadi bekal untuk melakukan desain [PRE-02:400]. Setiap bagian dari analisis model pada gambar 2.10 sebelah kiri menjadi bekal pada proses desain pada piramida model desain pada sebelah kanan.



Gambar 2.10 Hubungan antara Model Analisis dengan Model Desain
Sumber: [PRE-02:401]

Desain data mentransformasikan model domain informasi yang dibuat selama analisis ke dalam struktur data yang akan diperlukan untuk mengimplementasi perangkat lunak. Objek dan hubungan data yang ditetapkan dalam diagram hubungan entitas (ERD) dan isi detail yang digambarkan dalam kamus data, menjadi basis bagi aktivitas desain data.

Desain arsitektur mendefinisikan relasi antara elemen-elemen struktural utama, pola desain yang digunakan untuk mencapai kebutuhan yang ditentukan

untuk sistem dan batasan-batasan yang mempengaruhi bagaimana desain arsitektural ini diterapkan. Desain ini berdasarkan spesifikasi sistem, model-model analisis dan interaksi antara subsistem yang ditentukan dalam model analisis bagian DFD.

Desain *interface* menggambarkan bagaimana perangkat lunak berkomunikasi dalam dirinya sendiri, dengan sistem yang bertukar informasi dengannya dan dengan manusia yang menggunakannya. *Interface* mengimplikasikan aliran informasi (misal data dan atau kontrol). Dengan demikian, data dan diagram aliran kontrol yang diperoleh dari DFD diperlukan untuk desain *interface* ini.

Desain prosedural mentransformasikan elemen-elemen struktural dari arsitektur program ke dalam suatu deskripsi prosedural dari komponen-komponen perangkat lunak. Informasi yang diperoleh dari PSPEC, CSPEC dan STD berfungsi sebagai dasar dari desain prosedural.

2.10.1.1 *Entity Relational (ER) Model*

ER Model didasarkan pada bahwa dalam kehidupan nyata (*real work*) terdapat obyek yang saling berelasi baik antar obyek maupun dalam obyek itu sendiri [UTA-08:15]. ER Model digambarkan dalam bentuk diagram yang disebut dengan ER Diagram (ERD). ER Model terdiri atas 3 komponen utama yaitu:

1. Entitas

   Entitas merupakan suatu "obyek nyata" yang mampu dibedakan dengan obyek yang lain. Obyek tersebut dapat berupa orang, benda ataupun hal yang lainnya. Entitas digambarkan sebagai bentuk persegi panjang dengan nama entitas terletak didalamnya.

Gambar 2.11 Entitas
Sumber: [UTA-08:16]

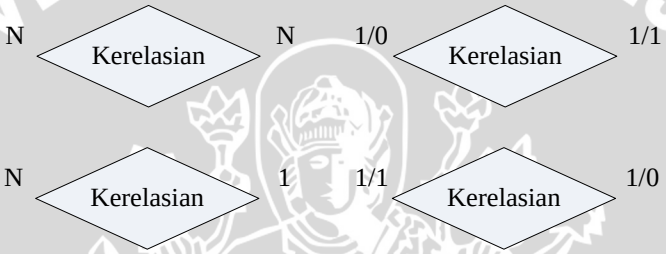
2. Atribut

   Atribut merupakan semua informasi yang berkaitan dengan entitas. Atribut digambarkan dengan suatu lingkaran. Jika terdapat atribut yang berfuingsi sebagai *key* maka atribut tersebut harus digaris bawahi.

Gambar 2.12 Atribut
Sumber: [UTA-08:16]

3. Kerelasian

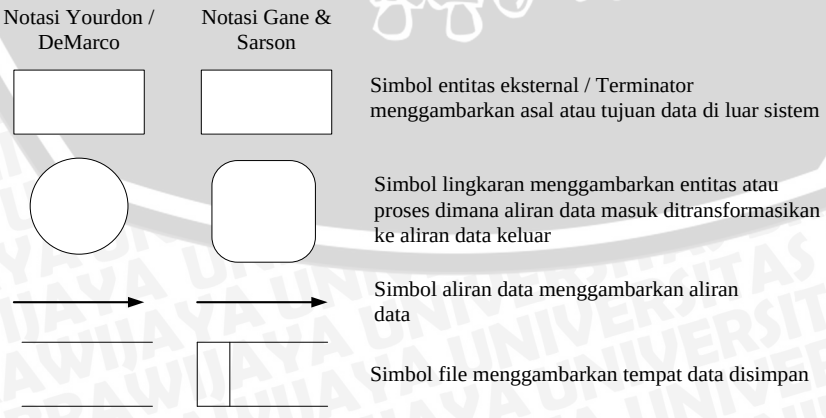
   Belah ketupat merupakan penggambaran hubungan antar entitas atau sering disebut dengan kerelasian. Pada kerelasian terdapat kardinalitas atau derajat hubungan antar entitas. Ukuran derajat hubungan antar entitas dilambangkan dengan N-N (*many-to-many*), N-1 (*many-to-one*), 1-N (*one-to-many*) dan 1-1 (*one-to-one*). Kadang juga ada notasi 0 jika terdapat kerelasian 1-1, untuk menentukan entitas mana yang lebih kuat.



Gambar 2.13 Kerelasian
Sumber: [UTA-08:17]

2.10.1.2 *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) adalah representasi grafik dari sebuah sistem. DFD menggambarkan komponen-komponen sebuah sistem, aliran-aliran data di mana komponen-komponen tersebut, dan asal, tujuan, dan penyimpanan dari data tersebut [WIN-09:1].



Gambar 2.14 Simbol *Data Flow Diagram*
Sumber: [WIN-09:1]

DFD dapat digambarkan dalam *Diagram Context* dan Level n. Huruf n dapat menggambarkan level dan proses di setiap lingkaran.

a. *Context Diagram* (CD)

Jenis pertama *Context Diagram* adalah *data flow diagram* tingkat atas (*DFD Top Level*), yaitu diagram yang paling tidak detail dari sebuah sistem informasi. Diagram ini menggambarkan aliran-aliran data ke dalam dan ke luar sistem dan ke dalam dan ke luar entitas-entitas eksternal. CD menggambarkan sistem dalam satu lingkaran dan hubungan dengan entitas luar. Lingkaran tersebut menggambarkan keseluruhan proses dalam sistem

b. Diagram Level *n* / *Data Flow Diagram Levelled*

Dalam diagram n DFD dapat digunakan untuk menggambarkan diagram fisik maupun diagram logis. Dimana Diagram Level *n* merupakan hasil pengembangan dari *Context Diagram* ke dalam komponen yang lebih detail tersebut disebut dengan *top-down partitioning*. Jika kita melakukan pengembangan dengan benar, kita akan mendapatkan DFD-DFD yang seimbang.

2.10.2 Pengujian *(Testing)*

Pengujian perangkat lunak adalah elemen penting dari jaminan kualitas perangkat lunak dan merepresentasikan kajian pokok dari spesifikasi, desain dan pengkodean. Dalam buku klasiknya mengenai pengujian perangkat lunak, Glen Myers menyatakan sejumlah aturan yang berfungsi sebagai sasaran pengujian:

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan suatu kesalahan.
2. *Test case* yang baik adalah *test case* yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum pernah ditemukan sebelumnya.

Bila pengujian dilakukan scara sukses (sesuai dengan sasaran tersebut), maka akan ditemukan kesalahan dalam perangkat lunak. Sebagai keuntungan sekunder, pengujian menunjukkan bahwa fungsi perangkat lunak bekerja sesuai dengan spesifikasi dan bahwa persyaratan kinerja telah dipenuhi [PRE-02:527]. Sebagai tambahan, data yang dikumpulkan pada saat pengujian dilakukan

memberikan indikasi yang baik mengenai reabilitas perangkat lunak dan beberapa menunjukkan kualitas perangkat lunak secara keseluruhan.

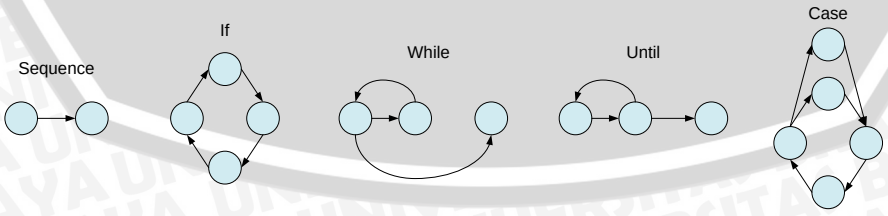
2.10.2.1 Teknik Pengujian

Dalam melakukan pengujian, diperlukan perancangan *test case* (kasus uji) yang akan digunakan untuk menguji perangkat lunak. Untuk melakukan perancangan *test case* tersebut, terdapat dua metode yaitu metode pengujian *white box* dan pengujian *black box*.

a) *White Box Testing*

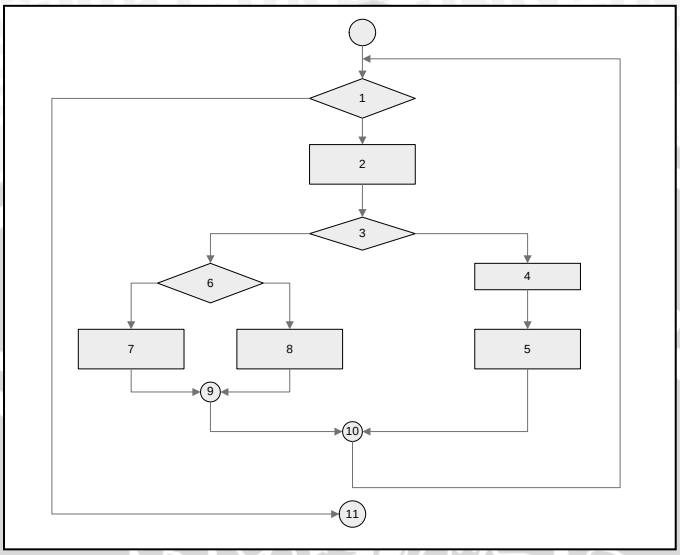
Pengujian *white box* suatu perangkat lunak didasarkan pada pengamatan yang teliti terhadap detail prosedural. Jalur-jalur logika yang melewati perangkat lunak diuji dengan memberikan *test case* yang menguji serangkaian kondisi dan atau *loop* tertentu. Ada beberapa metode untuk merealisasikan pengujian *white box*, diantaranya adalah *basis path testing*, *condition testing*, *loop tesing* serta *data flow testing*.

Pada skripsi ini, metode yang digunakan untuk merealisasikan *white box testing* adalah teknik *basis path testing*. Metode *basis path* ini memungkinkan perancang *test case* mengukur kompleksitas logis dari desain prosedural dan *basis set* dari jalur eksekusi. *Test case* yang dilakukan untuk menggunakan *basis set* tersebut dijamin untuk menggunakan setiap pernyataan di dalam program paling tidak sekali selama pengujian. Untuk melakukan teknik pengujian ini diperlukan penelusuran terhadap kontrol logika untuk menentukan *test case* dalam proses pengujian. Untuk menggambarkan aliran kontol logika, digunakan diagram alir (grafik alir) dengan notasi ditunjukkan pada Gambar 2.15.

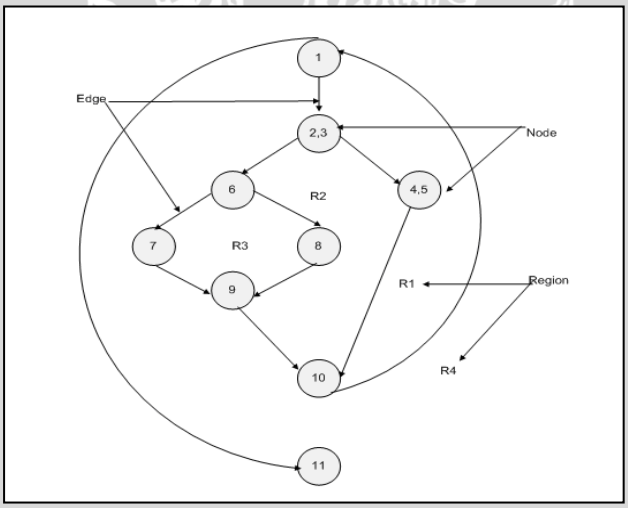


Gambar 2.15 Notasi grafik alir
Sumber: [PRE-02:535]

Salah satu cara untuk mendapatkan grafik alir adalah melalui struktur *flowchart*. Sebagai contoh, dari *flowchart* seperti yang ditunjukkan pada Gambar 2.16 bisa dimodelkan ke dalam sebuah grafik alir pada Gambar 2.17.



Gambar 2.16 *Flowchart*
Sumber: [PRE-02:536]



Gambar 2.17 Grafik alir dari *flowchart* pada Gambar 2.17
Sumber: [PRE-02:536]

Untuk menentukan jumlah jalur independen dalam basis set suatu program, serta memberikan batas atas bagi jumlah pengujian yang harus dilakukan untuk memastikan bahwa semua pernyataan telah dieksekusi sedikitnya satu kali, digunakan kompleksitas siklomatis. Kompleksitas siklomatis merupakan

metrik perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis atas suatu program. Untuk menentukan kompleksitas siklomatis bisa dilakukan dengan beberapa cara, diantaranya:

1. Jumlah *region* grafik alir sesuai dengan kompleksitas siklomatis.
2. Kompleksitas siklomatis V(G), untuk grafik G adalah V(G) = E – N + 2, dimana E adalah jumlah *edge*, dan N adalah jumlah *node*.
3. V(G) = P + 1, dimana P adalah jumlah simpul predikat yang diisikan dalam grafik alir G.

Langkah-langkah untuk melakukan *test case* adalah sebagai berikut:

1. Dengan menggunakan desain atau *source code* sebagai dasar untuk memodelkan ke dalam grafik alir.
2. Dari hasil pemodelan grafik alir, ditentukan kompleksitas siklomatis V(G).
3. Menentukan sebuah basis set dari jalur independent secara liner. Harga kompleksitas siklomatis memberikan jumlah jalur independen secara linier melalui struktur kontrol program.
4. Menyiapkan *test case* untuk tiap-tiap jalur.

b) *Black Box Testing*

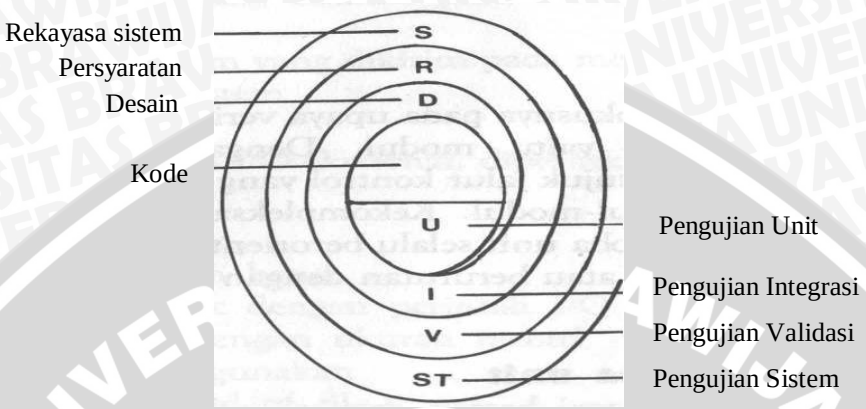
Pengujian *black box* fokus pada persyaratan-persyaratan fungsional perangkat lunak. Dengan demikian, pengujian *black box* memungkinkan perekayasa perangkat lunak mendapatkan serangkaian kondisi input yang sepenuhnya menggunakan semua persyaratan fungsional untuk suatu program. Pengujian *black box* berusaha menemukan kesalahan dalam kategori sebagai berikut:

1. Fungsi-fungsi yang tidak benar (hilang)
2. Kesalahan *interface*
3. Kesalahan dalam struktur data atau akses ke *database* eksternal
4. Kesalahan kinerja
5. Inisialisasi dan kesalahan terminasi

Tidak seperti teknik *white box testing,* yang dilakukan pada awal proses pengujian, *black box testing* dilakukan pada akhir proses pengujian. Karena *black box testing* ditujukan untuk mengabaikan struktur kontrol, perhatian difokuskan pada domain informasi.

2.10.2.2 Strategi Pengujian

Strategi pengujian perangkat lunak mengintegrasikan metode perancangan kasus uji (*test case design method*) kedalam sebuah rangkaian langkah-langkah pengujian yang terencana [PRE-01:477].



Gambar 2.18 Strategi pengujian
Sumber: [PRE-02:576]

Strategi pengujian perangkat lunak dapat dilihat dalam konteks spiral. *Unit testing* dimulai pada pusaran spiral dan terpusat pada masing-masing satuan perangkat lunak pada saat diimplementasikan di dalam kode sumber. Pengujian berjalan dengan bergerak keluar sepanjang spiral ke *integration testing* dimana fokusnya adalah desain dan arsitektur perangkat lunak. Dengan mengambil urutan keluar dari dalam spiral, kita menemukan *validation testing* dimana persyaratan yang dibangun sebagai bagian dari analisis persyaratan perangkat lunak divalidasi terhadap perangkat lunak yang telah di konstruksi. Akhirnya kita akan sampai pada *system testing* dimana perangkat lunak dan elemen sistem yang lain diuji secara keseluruhan [PRE:02-576].

a) Pengujian Unit

Pengujian unit berfokus pada usaha verifikasi pada inti terkecil dari desain perangkat lunak, yakni modul. Dengan menggunakan gambaran desain prosedural sebagai panduan, jalur kontrol yang penting diuji untuk mengungkap kesalahan didalam batas dari modul tersebut. Kompleksitas relatif dari pengujian dan kesalahan yang diungkap dibatasi oleh ruang lingkup batasan yang dibangun untuk pengujian unit. Pengujian unit menggunakan teknik pengujian *white-box*,

dengan menggunakan jalur spesifik di dalam sebuah struktur kontrol modul untuk memastikan cakupan lengkap dan deteksi kesalahan maksimum [PRE-02:581].

b) Pengujian Integrasi

Pengujian integrasi adalah teknik sistematis untuk mengkonstruksi struktur program sambil melakukan pengujian untuk mengungkap kesalahan sehubungan dengan *interfacing*. Sasarannya adalah untuk mengambil modul yang dikenai pengujian unit dan membangun struktur program yang ditentukan oleh desain [PRE-02:585].

c) Pengujian Validasi

Pada kulminasi pengujian integrasi, perangkat lunak secara lengkap dirakit sebagai suatu paket, kesalahan *interfacing* telah diungkap dan dikoreksi, dan seri akhir dari pengujian perangkat lunak - pengujian validasi – dapat dimulai. Validasi dapat ditentukan dengan berbagai cara, teteapi definisi yang sederhana adalah bahwa validasi berhasil bila perangkat lunak berfungsi dengan cara yang dapat diharapkan [PRE-02:594].