BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi Geografis

Secara umum SIG dapat diartikan sebagai suatu komponen yang terdiri dari perangkat keras, perangkat lunak, data geografis dan sumber daya manusia untuk memasukkan, menyimpan, memperbaiki, memperbaharui, mengelola, mengintegrasikan, menganalisa dan menampilkan data secara spasial yang terkait dengan permukaan bumi.

II.1.1. *Data Spasial*

Sebagian besar data yang akan ditangani dalam SIG merupakan data spasial, data yang berorientasi geografis. Data ini memiliki sistem koordinat tertentu sebagai dasar referensinya dan mempunyai dua bagian penting yang berbeda dari data lain, yaitu informasi lokasi (spasial) dan informasi deskriptif (atribut) yang dijelaskan berikut ini:

1. Informasi lokasi (spasial), berkaitan dengan suatu koordinat baik koordinat geografi (lintang dan bujur) dan koordinat XYZ, termasuk diantaranya informasi datum dan proyeksi.
2. Informasi deskriptif (atribut) atau informasi nonspasial, suatu lokasi yang memiliki beberapa keterangan yang berkaitan dengannya. Contoh jenis vegetasi, populasi, luasan, kode pos, dan sebagainya.

II.1.2. *Format Data Spasial*

Secara sederhana format dalam bahasa komputer berarti bentuk dan kode penyimpanan data yang berbeda antara file satu dengan lainnya. Dalam SIG, data spasial dapat direpresentasikan dalam dua format, yaitu:

a. Data vektor

   Data vektor merupakan bentuk bumi yang direpresentasikan ke dalam kumpulan garis, area (daerah yang dibatasi oleh garis yang berawal dan berakhir pada titik yang sama), titik dan nodes (titik perpotongan antara dua buah garis).

b. Data raster

   Data raster (disebut juga dengan sel grid) adalah data yang dihasilkan dari sistem penginderaan jauh. Pada data raster, obyek geografis direpresentasikan sebagai struktur sel grid yang disebut dengan *pixe*l (*picture element*) (Mohd. Ichsan ; 2012 : 51).

II.2. Pengertian Lokasi

Lokasi adalah posisi suatu tempat, benda, peristiwa, atau gejala di permukaan bumi dalam hubungannya dengan tempat, benda, hejala dan peristiwa lain. Terdapat dua komponen lokasi, yaitu arah dan jarak. Arah menunjukkan posisi suatu tempat jika dibandingkan dengan tempat di mana orang tersebut berada. Adapun jarak adalah ukuran jauh atau dekatnya dua benda atau gejala tersebut. Contoh, Bandung terletak di sebelah selatan Jakarta, arah tersebut akan berbeda jika orang yang bertanya berada di Semarang, Bandung terletak disebelah barat.

Ada dua macam lokasi yaitu absolut dan lokasi relatif. Lokasi absolut adalah posisi sesuatu berdasarkan koordinat garis lintang dan garis bujur. Misalnya, Indonesia terletak di antara 6^{o} LU-11^{o} LS dan antara 95^{o} BT-141^{o} BT. Lokasi absolut mutlak adanya dan dapat dipercaya karena masa daratan relatif tetap, aspek perubahannya kecil sekali, dan berlaku umum di seluruh dunia. Melalui lokaso absolut, sesorang dapat mengetahui jarak dan arah suatu tempat ke tempat lain di permukaan bumi. Dengan bantuan garis lintang seseorang dapat menggambarkan kondisi iklim suatu daerah, berarti vegetasinya bersifat heterogen, selalu menghijau, kehidupan hewannya beragam, penduduknya termasuk ras mongoloid, sebagaian besar penduduknya hidup dalam bidang pertanian, dan ciri-ciri lainnya. Dengan mengetahui lokasi suatu tempat berdasarkan garis lintang, seseorang akan memperoleh gambaran tentang kondisi iklim, kehidupan tumbuhan, hewan dan manusianya (Hartono ; 2008 : 9).

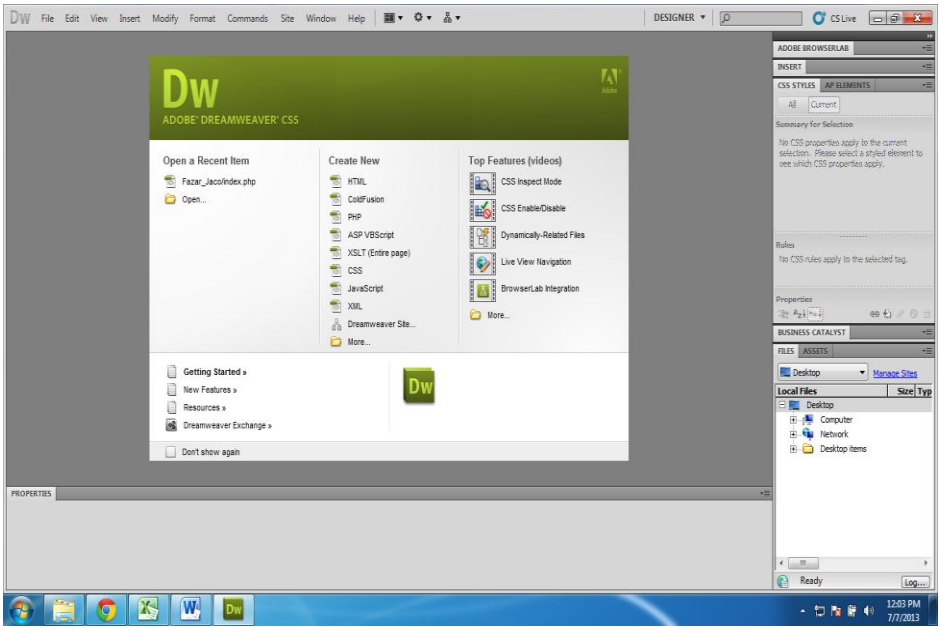
II.3. Quantum GIS

Quantum GIS (QGIS) adalah cross-platform perangkat lunak bebas (open source) desktop pada sistem informasi geografis (SIG). Aplikasi ini dapat menyediakan data, melihat, mengedit, dan kemampuan analisis. Quantum GIS berjalan pada sistem operasi yang berbeda termasuk Mac OS X , Linux , UNIX , dan Microsoft Windows . Dalam perizinan, QGIS sebagai perangkat lunak bebas aplikasi di bawah GPL(General Public License), dapat secara bebas dimodifikasi untuk melakukan tugas yang berbeda atau lebih khusus. Quantum GIS memungkinkan penggunaan shapefiles, pertanggungan, dan Geodatabases pribadi.

MapInfo , PostGIS , dan beberapa format lain yang didukung di Quantum GIS (Adam Suseno ; 2012 ; 15).

II.4. Pengertian Macromedia Dreamweaver

Macromedia Dreamweaver adalah sebuah *software web design software web design* yang menawarkan cara mendesain *website* dengan dua langkah sekaligus dalam satu waktu, yaitu mendesain dan meprogram. Dreamweaver memiliki satu jendela mini yang disebut HTML *Source*, tempat kode-kode HTML tertulis. Setiap kali kita mendesain *web*, seperti menulis kata-kata, meletakkan gambar, membuat tabel dan proses lainnya, tag-tag HTML akan tertulis secara langsung mengiringi proses pengaturan *website*. Artinya kita memiliki kesempatan untuk mendesain. *Website* sekaligus mengenal tag-tag HTML yang membangun *website* itu. Di lain kesempatan kita juga dapat mendesain *website* hanya dengan menulis tag-tag dan teks lain di jendela HTML *Source* dan hasilnya dapat dilihat langsung di layar (M. Suyanto ; 2009 : 244).



Gambar II.1. Tampilan Dreamweaver
(Sumber : M. Suyanto ; 2009 : 244)

II.5. Pengertian PHP

PHP merupakan bahasa *scripting* yang berjalan di sisi *server* (*server-slide*). Semua perintah yang ditulis akan dieksekuis oleh *server* dan hasil jadinya dapat dilihat melalui *browser*. Saat ini PHP versi 4 sudah di-*release* di pasaran, mengikuti jejak kesuksesan versi sebelumnya, PHP 3. Selain dapat digunakan untuk berbagai sistem operasi, koneksi *database* yang sangat mudah menyebabkan bahasa *scripting* ini digemari para *programmer web*. Beberapa perintah PHP yang kita pelajari sebatas pada perintah untuk menampilkan *tag-tag* wml, akses *database* MySQL dan pengiriman *email* (Ridwan Sanjaya ; 2009 : 73).

II.6. Pengertian Database

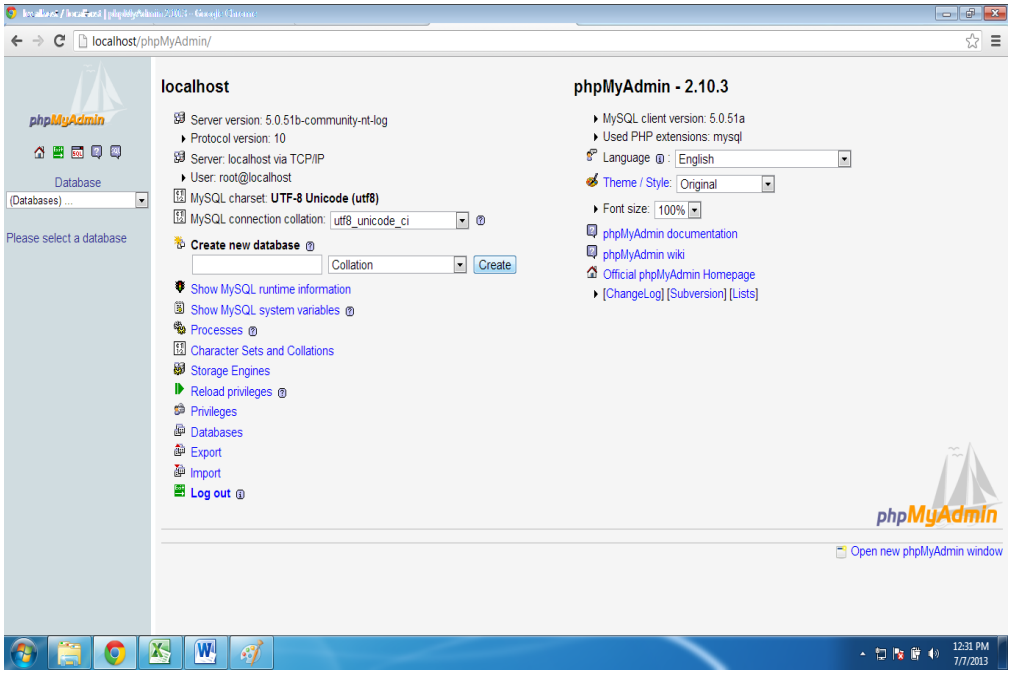
Secara sederhana *database* (basis data/pangkalan data) dapat diungkapkan sebagai suatu pengorganisasian data dengan bantuan komputer yang

memungkinkan data dapat diakses dengan mudah dan cepat. Pengertian akses dapat mencakup pemerolehan data maupun pemanipulasian data seperti menambah serta menghapus data. Dengan memanfaatkan komputer, data dapat disimpan dalam media pengingat yang disebut *harddisk.* Dengan menggunakan media ini, keperluan kertas untuk menyimpan data dapat dikurangi. Selain itu, data menjadi lebih cepat untuk diakses terutama jika dikemas dalam bentuk *database*. (Agustinus Mujilan ; 2012 : 23).

II.7. Pengertian MySQL

MySQL adalah suatu sistem manajemen basis data relasional (RDBMS-*Relational Database Management System*) yang mampu bekerja dengan cepat, kokoh, dan mudah digunakan. Contoh RDBMS lain adalah *Oracle, Sybase*. Basis data memungkinkan anda untuk menyimpan, menelusuri, menurutkan dan mengambil data secara efesien. *Server* MySQL yang akan membantu melakukan fungsionaliitas tersebut. Bahasa yang digunakan oleh MySQL tentu saja adalah SQL-*standar* bahasa basis data relasional di seluruh dunia saat ini.

MySQL dikembangkan, dipasarkan dan disokong oleh sebuah perusahaan Swedia bernama MySQL AB. RDBMS ini berada di bawah bendera GNU GPL sehingga termasuk produk *Open Source* dan sekaligus memiliki lisensi komersial. Apabila menggunakan MySQL sebagai basis data dalam suatu situs Web. Anda tidak perlu membayar, akan tetapi jika ingin membuat produk RDBMS baru dengan basis MySQL dan kemudian mengualnua, anda wajib bertemu mudah dengan lisensi komersial (Antonius Nugraha Widhi Pratama ; 2010 : 10).



Gambar II.2. Tampilan MySQL
(Sumber : Antonius Nugraha Widhi Pratama ; 2010 : 10)

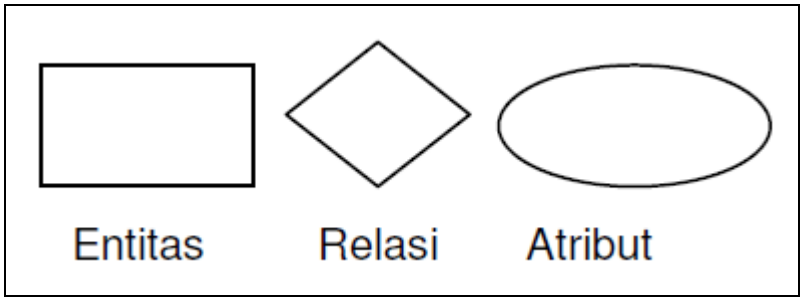
II.8. *Entity Relationship Diagram* (ERD)

Model *entity-relationship* pertama kali diperkenalkan oleh Peter Chen pada tahun 1976. Dalam pemodelan ini dilakukan dengan tahapan sebagai berikut:

a. Memilih entitas-entitas yang akan disusun dalam basis data dan menentukan hubungan antar entitas yang telah dipilih.

b. Melengkapi atribut-atribut yang sesuai pada entitas dan hubungan sehingga diperoleh bentuk tabel normal penuh (ternormalisasi).

Elemen-elemen dalam model ER dapat digambarkan pada gambar diagram di bawah ini :



Gambar II.1. Elemen ERD
(*Sumber : Haidar Dzacko, 2007 : 21*).

Entitas merupakan sesuatu yang dapat diidentifikasikan dalam lingkungan kerja pengguna. Entitas yang diberikan tipe dikelompokkan ke kelas entitas. Perbedaan antara kelas entitas dan instansi entitas adalah sebagai berikut:

a. Kelas entitas adalah kumpulan entitas dan dijelaskan oleh struktur atau format entitas di dalam kelas.
b. Instansi kelas merupakan bentuk penyajian dari fakta entitas. (Haidar Dzacko ; 2007 : 21).

II.9. Kamus Data

Kamus data (data *dictionary*) mencakup definisi-definisi dari data yang disimpan di dalam basis data dan dikendalikan oleh sistem manajemen basis data. Figur 6.5 menunjukkan hanya satu tabel dalam basis data jadwal. Struktur basis data yang dimuat dalam kamus data adalah kumpulan dari seluruh definisi *field*, definisi tabel, relasi tabel, dan hal-hal lainnya. Nama *field* data, jenis data (seperti teks atau angka atau tanggal), nilai-nilai yang valid untuk data, dan karakteristik-karakteristik lainnya akan disimpan dalam kamus data. Perubahan-perubahan pada struktur data hanya dilakukan satu kali di dalam kamus data, program-program

palikasi yang mempergunakan data tidak akan ikut terpengaruh (Raymond McLeod ; 2008 : 171).

II.10. Teknik Normalisasi

Salah satu topik yang cukup kompleks dalam dunia manajemen *database* adalah proses untuk menormalisasi tabel-tabel dalam *database relasional.* Dengan normalisasi kita ingin mendesain *database relasional* yang terdiri dari tabel-tabel berikut :

1. Berisi data yang diperlukan.
2. Memiliki sesedikit mungkin redundansi.
3. Mengakomodasi banyak nilai untuk tipe data yang diperlukan.
4. Mengefisienkan update.
5. Menghindari kemungkinan kehilangan data secara tidak disengaja/tidak diketahui.

Alasan utama dari normalisasi *database* minimal sampai dengan bentuk normal ketiga adalah menghilangkan kemungkinan adanya "*insertion anomalies"*, "*deletion anomalies",* dan "*update anomalies".* Tipe-tipe kesalahan tersebut sangat mungkin terjadi pada *database* yang tidak normal.

1. Bentuk tidak normal

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti format tertentu, dapat saja tidak lengkap dan terduplikasi. Data dikumpulkan apa adanya sesuai keadaanya.

2. Bentuk normal tahap pertama (1" *Normal Form*)

Sebuah table disebut 1NF jika :

- Tidak ada baris yang duplikat dalam tabel tersebut.
- Masing-masing *cell* bernilai tunggal

Catatan: Permintaan yang menyatakan tidak ada baris yang duplikat dalam sebuah tabel berarti tabel tersebut memiliki sebuah kunci, meskipun kunci tersebut dibuat dari kombinasi lebih dari satu kolom atau bahkan kunci tersebut merupakan kombinasi dari semua kolom.

3. Bentuk normal tahap kedua (2nd *normal form*)

Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam primary key memiliki ketergantungan fungsional pada primary key secara utuh.

4. Bentuk normal tahap ketiga (3rd *normal form*)

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X -> A, dimana A mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

- X haruslah superkey pada tabel tersebut.
- Atau A merupakan bagian dari primary key pada tabel tersebut.

5. Bentuk Normal Tahap Keempat dan Kelima

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk Normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun

bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

6. Boyce Code Normal Form (BCNF)

a. Memenuhi 1st NF

b. Relasi harus bergantung fungsi pada atribut superkey (Kusrini ; 2007 : 39-43).

II.11. UML (*Unified Modeling Language*)

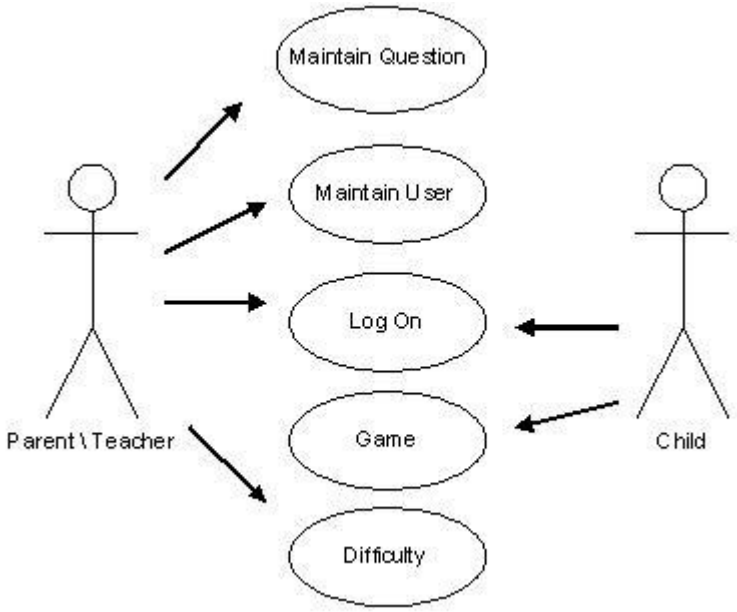
Menurut Sri Dharwiyanti (2013) *Unified Modelling Language* (UML) adalah sebuah "bahasa" yg telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

1. *Use Case* Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan system untuk melakukan pekerjaan-pekerjaan tertentu. *Use case diagram* dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal. Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

Sebuah *use case* juga dapat meng-*extend use case* lain dengan *behaviour*-nya sendiri. Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.



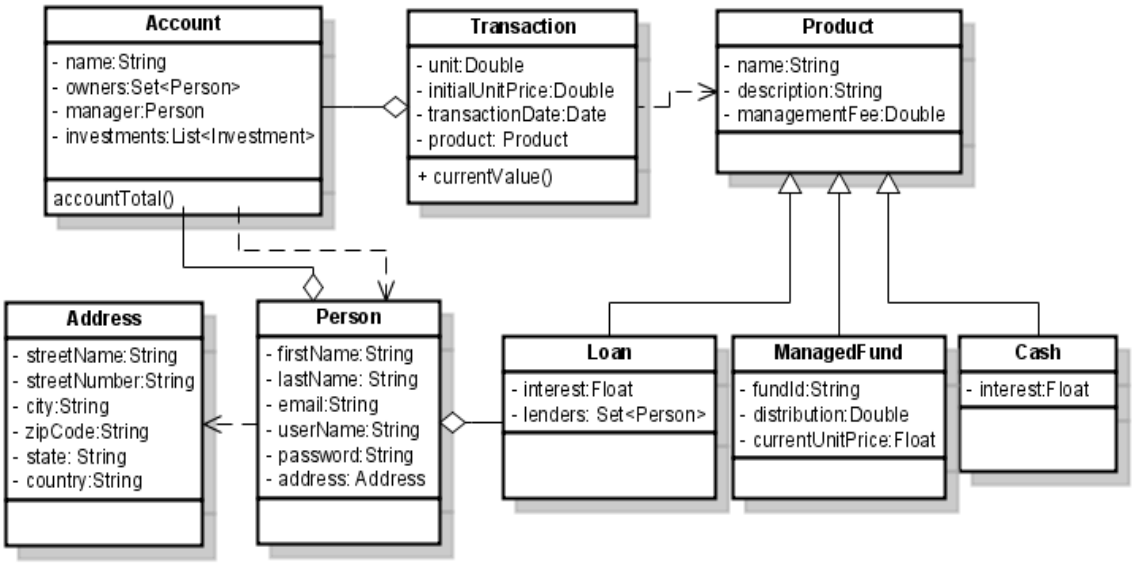
Gambar II.3. *Usecase* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 5)

2. *Class* Diagram

Class adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus

menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain.



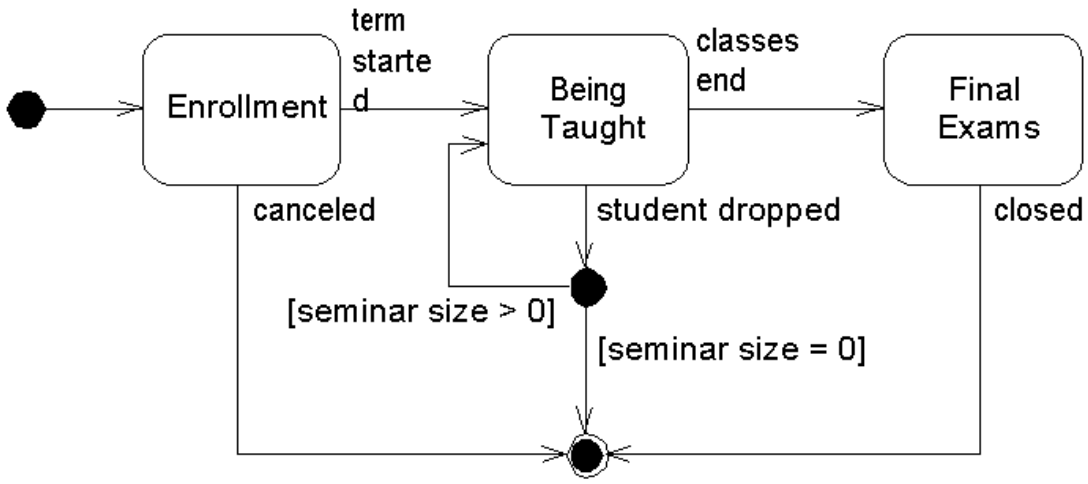
Gambar II.4. *Class* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 6)

3. *Statechart* Diagram

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).

Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu. Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku. *Action* yang dilakukan sebagai akibat

dari *event* tertentu dituliskan dengan diawali garis miring. Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.



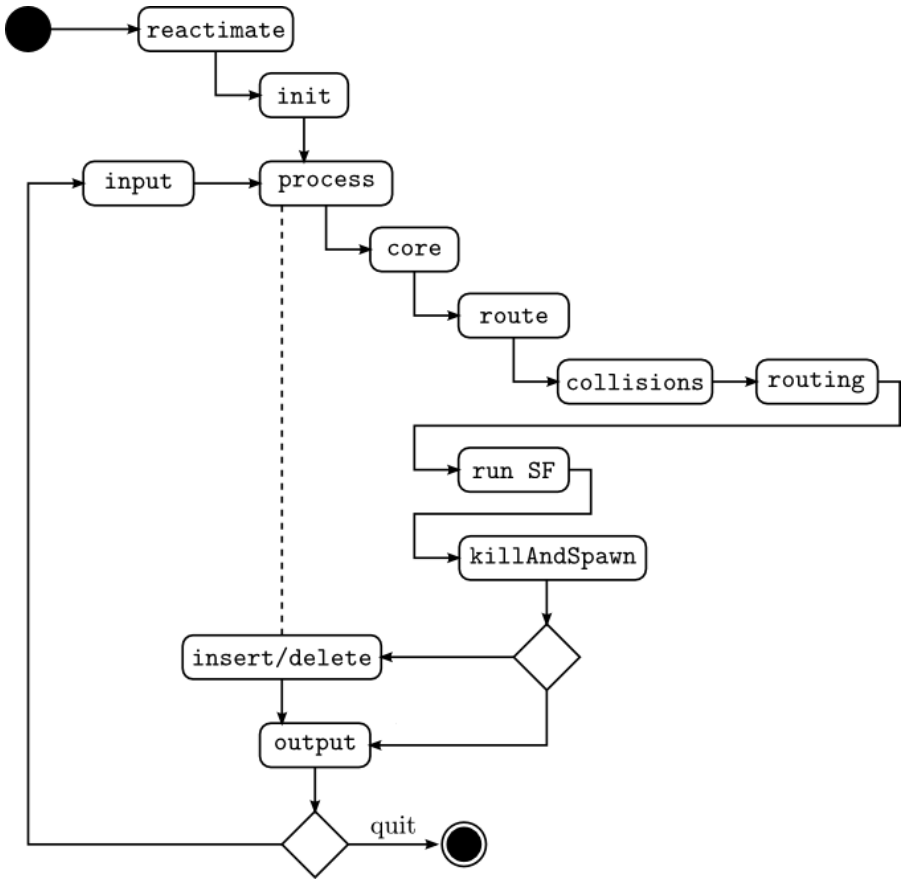
Gambar II.5. *Statechart* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 7)

4. *Activity* Diagram

Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. *Activity diagram* merupakan *state diagram* khusus, di mana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu *activity diagram* tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan

bagaimana aktor menggunakan sistem untuk melakukan aktivitas

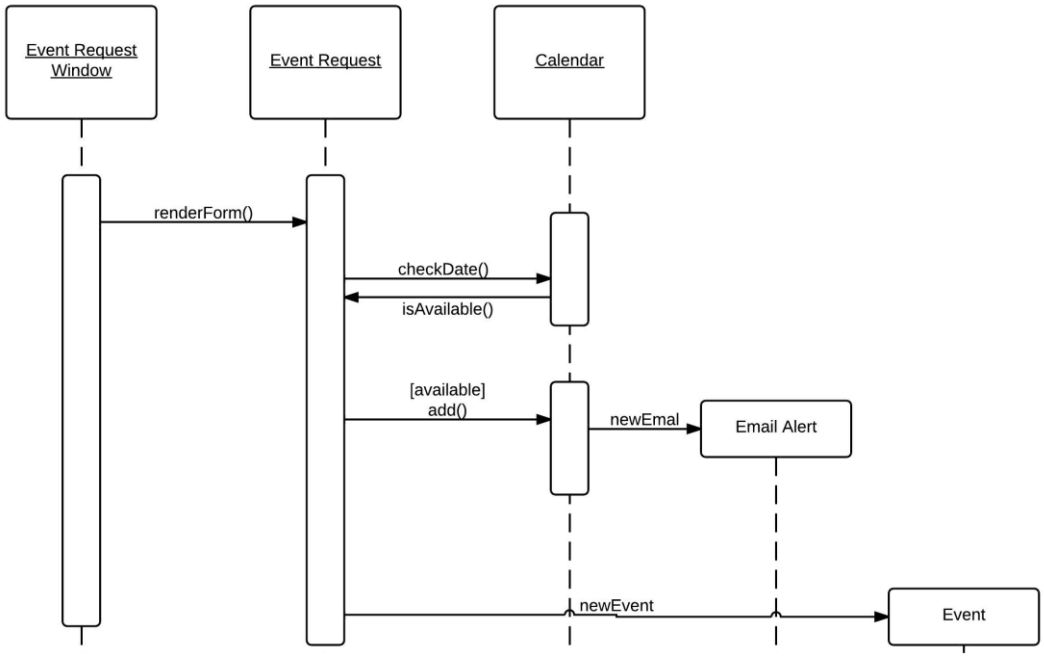


Gambar II.6. *Activity* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 8)

5. *Sequence* Diagram

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

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

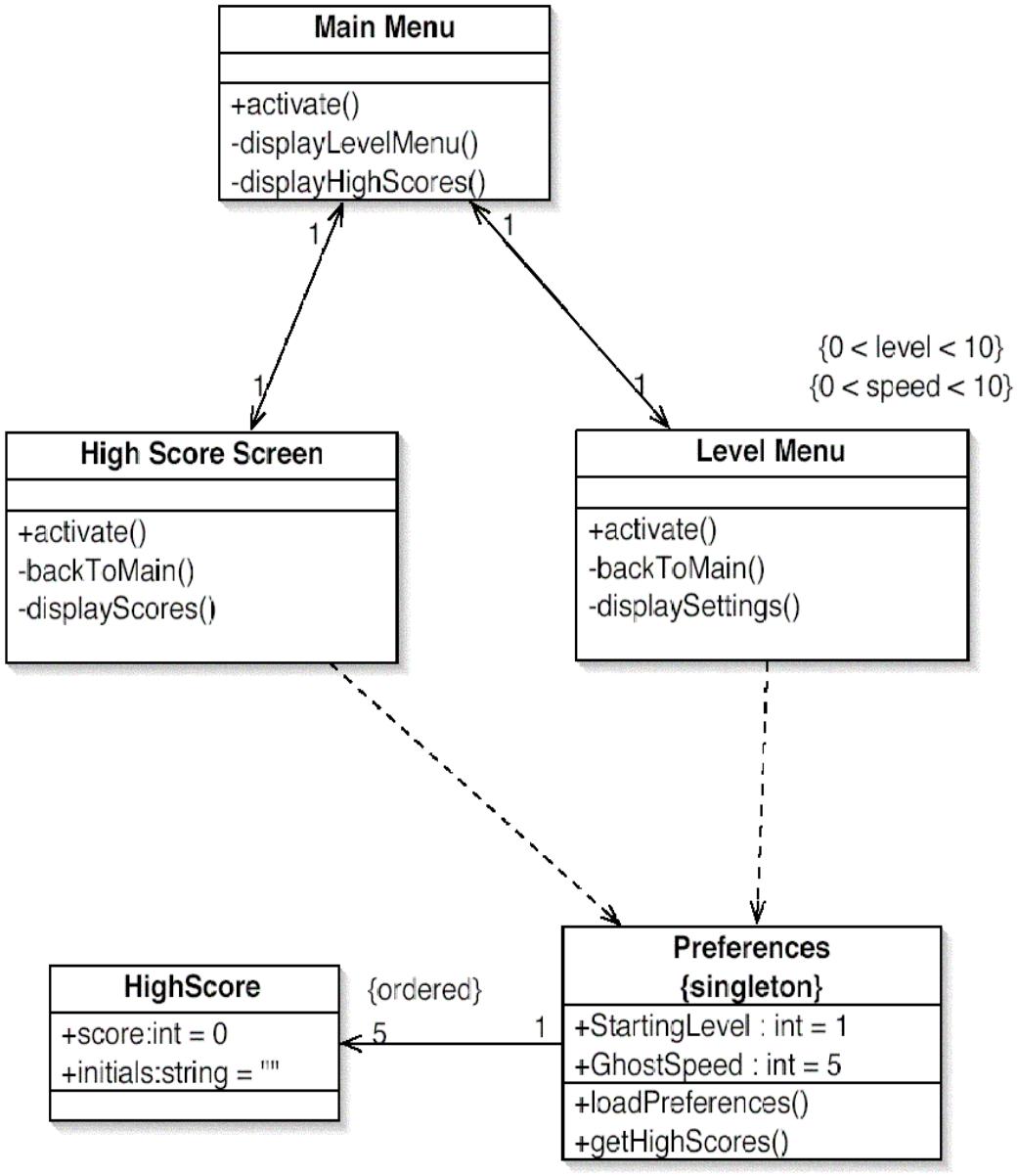


Gambar II.7. *Sequence* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 9)

6. *Collaboration* Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*.

Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

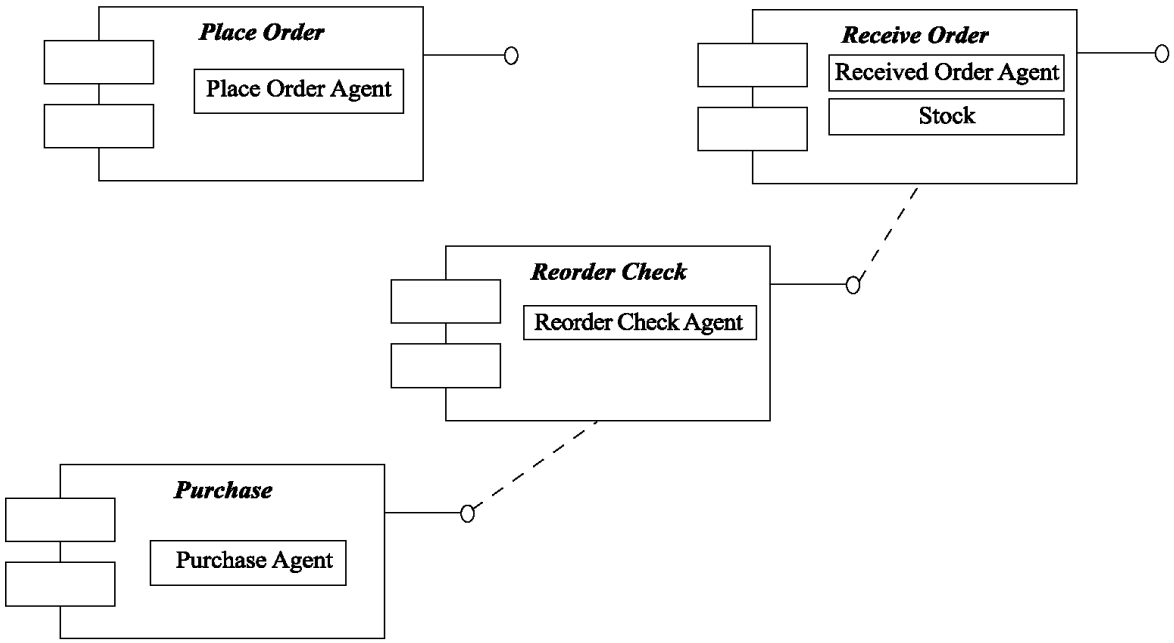


Gambar II.8. *Collaboration* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 9)

7. *Component* Diagram

Component diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik *library* maupun *executable*, baik yang muncul pada *compile time*, *link*

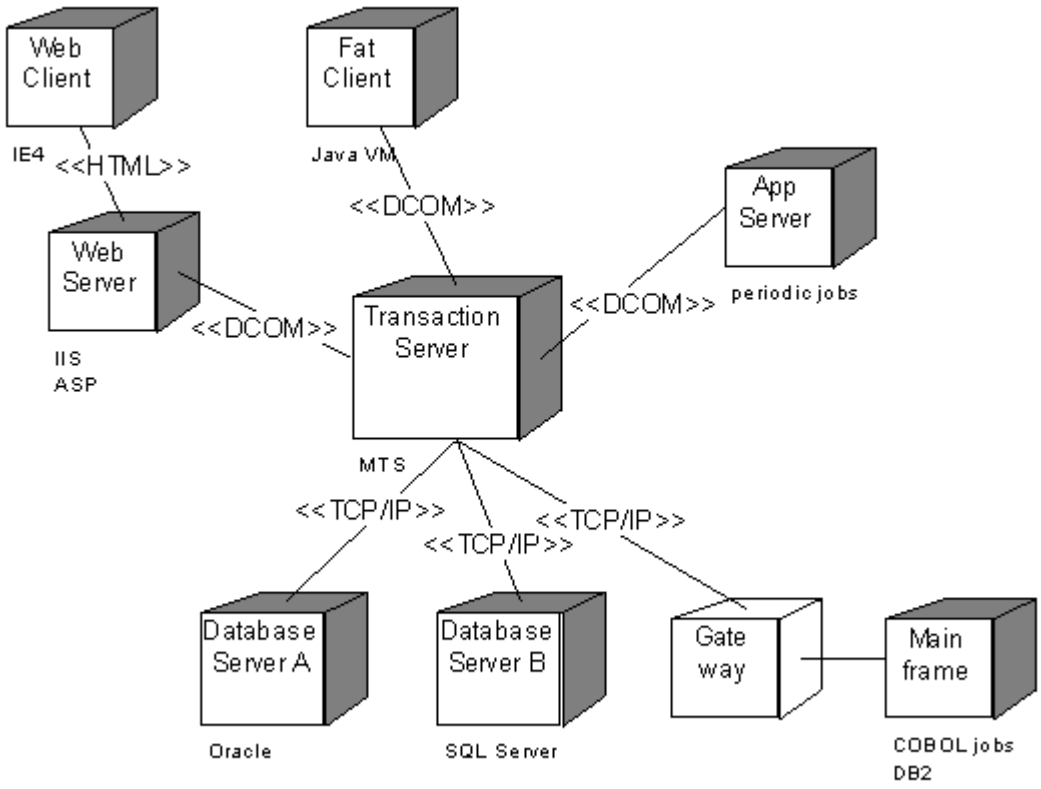
time, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil. Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.



Gambar II.9. *Component* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 10)

8. *Deployment* Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal Sebuah *node* adalah server, *workstation*, atau piranti keras lain yang digunakan untuk men-*deploy* komponen dalam lingkungan sebenarnya.



Gambar II.10. *Deployment* Diagram
(Sumber : Sri Dharwiyanti ; 2013 : 11)