# BAB II

# TINJAUAN PUSTAKA

## II.1. Sistem Informasi

Sistem informasi bukan merupakan hal yang baru, yang baru adalah komputerisasinya. Sebelum ada komputer, teknik penyaluran informasi yang memungkinkan manajer merencanakan serta mengendalikan operasi telah ada. Komputer menambahkan satu atau dua dimensi, seperti kecepatan, ketelitian dan penyediaan data dengan volume yang lebih besar yang memberikan bahan pertimbangan yang lebih banyak untuk mengambil keputusan.

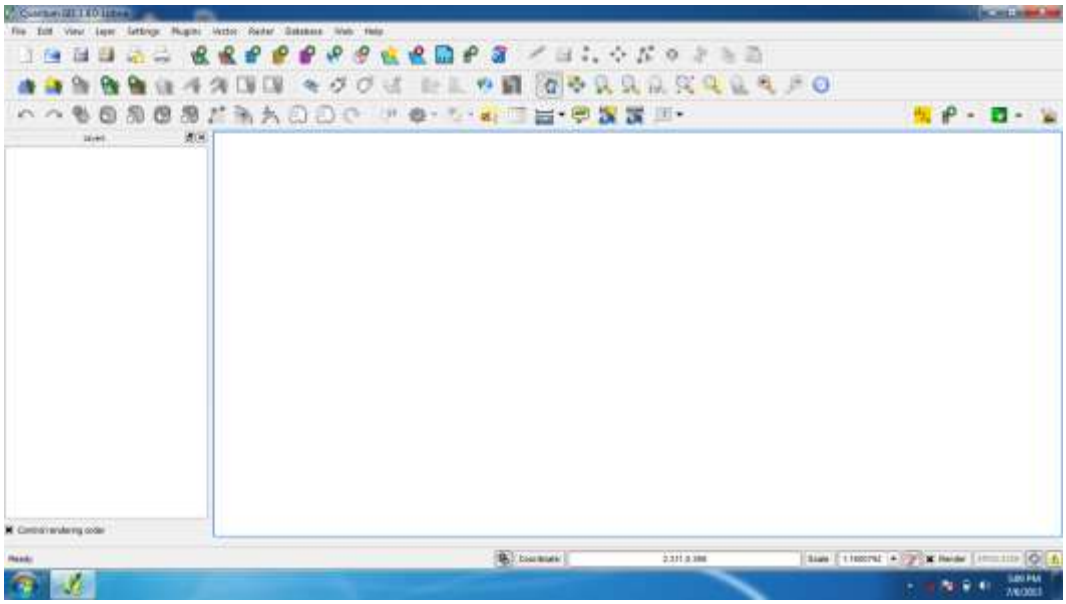
Sistem informasi adalah suatu sistem di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi untuk dapat menyediakan laporan-laporan yang diperlukan oleh pihak luar tertentu (Abdul Kadir ; 2012).

## II.2. Sistem Informasi Geografis

Sistem Informasi Geografis (SIG) atau *Geographic Information System* (GIS) adalah sebuah sistem yang didesain untuk menangkap, menyimpan, memanipulasi, menganalisa, mengatur dan menampilkan seluruh jenis data geografis. Sistem Informasi Geografis dirancang untuk mengumpulkan, menyimpan, dan menganalisis objek dan fenomena dimana daerah geografi merupakan karakteristik yang penting atau kritis untuk dianalisis (Saddam Hussein ; 2012).

## II.3. *Quantum GIS*

*Quantum GIS* merupakan salah satu perangkat lunak *open source* di bawah proyek resmi dari *Open Source Geospatial Foundation* (OSGeo) yang dapat dijalankan dalam sistem operasi *Windows*, *Mac OSX*, *Linux* dan *Unix*. Aplikasi ini menawarkan pengolahan data geospasial dengan berbagai format dan fungsionalitas *vektor*, *raster* dan *database*. Untuk keperluan analisis spasial, aplikasi ini telah cukup lengkap karena telah terintegrasi dengan perangkat lunak *GRASS*. Pemanfaatan perangkat lunak Quantum GIS ini dapat digunakan sebagai pilihan alternatif dari *software* SIG komersial seperti *ArcView* maupun *ArcGIS*. *Quantum GIS* dapat diakses melalui situs resmi yang beralamatkan www.qgis.org (Retno Astrini ; 2012).



**Gambar II.1. Tampilan *Quantum GIS***
**(Sumber : Retno Astrini ; 2012)**

## II.4. *Macromedia Dreamweaver*

*Macromedia Dreamweaver* adalah sebuah HTML editor profesional untuk mendesain secara visual dan mengelola situs *web* maupun halaman *web*. Bilamana kita menyukai untuk berurusan dengan kode-kode HTML secara manual atau lebih menyukai bekerja dengan lingkungan secara visual dalam melakukan editing, *Dreamweaver* mambuatnya menjadi lebih mudah dengan menyediakan tool-tool yang sangat berguna dalam peningkatan kemampuan dan pengalaman kita dalam mendesain *web*.

*Dreamweaver MX* dalam hal ini digunakan untuk *web* desain. *Dreamweaver MX* mengikutsertakan banyak tool untuk kode-kode dalam halaman *web* beserta fasilitas-fasilitasnya, antara lain : Referensi HTML, *CSS* dan *Javascript*, Javascript *debugger*, dan editor kode ( tampilan kode dan Code inspector) yang mengizinkan kita mengedit kode *Javascript*, *XML*, dan dokumen teks lain secara langsung dalam *Dreamweaver*. Teknologi *Dreamweaver Roundtrip* HTML mampu mengimpor dokumen HTML tanpa perlu memformat ulang kode tersebut dan kita dapat menggunakan *Dreamweaver* pula untuk membersihkan dan memformat ulang HTML bila kita menginginkannya.

Selain itu *Dreamweaver* juga dilengkapi kemampuan manajemen situs, yang memudahkan kita mengelola keseluruhan elemen yang ada dalam situs. Kita juga dapat melakukan evaluasi situs dengan melakukan pengecekan *broken link*, kompatibilitas *browser*, maupun perkiraan waktu *download* halaman *web* (Risdiyanto ; 2013).


**Gambar II.2. Tampilan *Dreamweaver***
**(Sumber : Risdiyanto ; 2013)**

## II.5. PHP

PHP singkatan dari PHP : *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*). 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 diesekusi pada *server* di mana *script* tersebut dijalankan. (Anhar ; 2010).

## II.6. *Database*

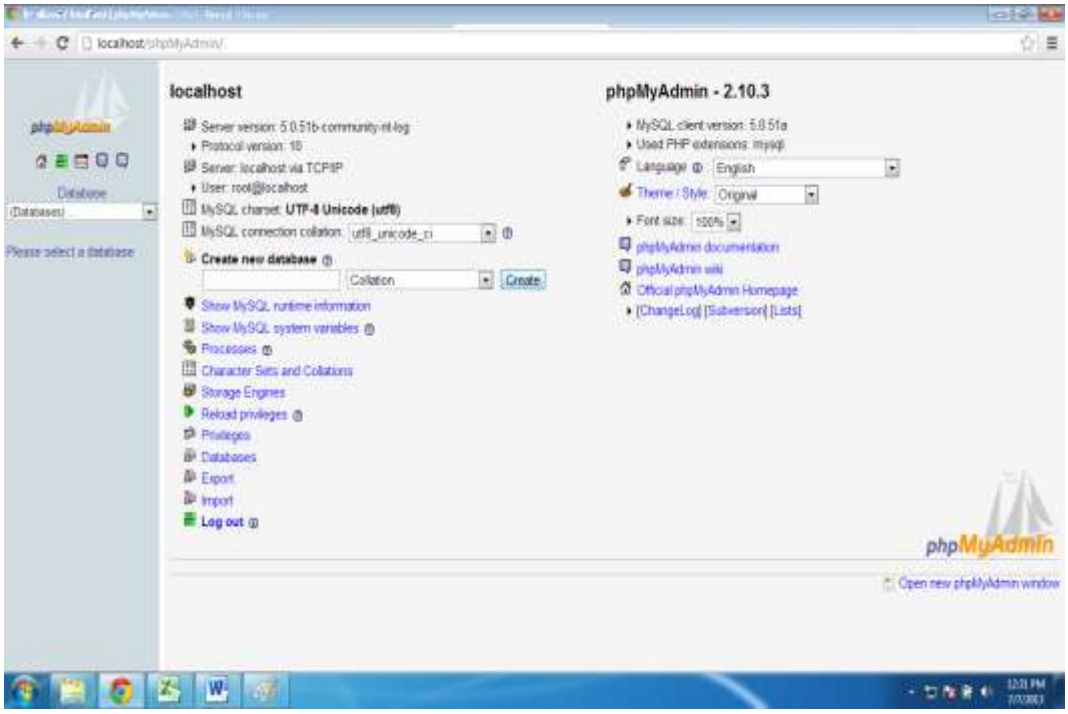
*Database* adalah sekumpulan *file* data yang saling berhubungan dan diorganisasi sedemikian rupa sehingga memudahkan untuk mendapat dan memproses data. Lingkungan sistem *database* menekankan data yang tidak tergantung (*idenpendent data*) pada aplikasi yang akan menggunakan data. Data adalah kumpulan fakta dasar (mentah) yang terpisah.

Sebuah *database* harus dibuat dengan rapi agar data yang dimasukkan sesuai dengan tempatnya. Sebagai contoh, di sebuah perpustakaan, penyimpanan buku dikelompokkan berdasar jenis atau kategori-kategori tertentu, misalnya kategori buku komputer, buku pertanian, dan lain-lain. Kemudian dikelompokkan lagi berdasarkan abjad judul buku, ini dilakukan agar setiap pengunjung dapat dengan mudah mencari dan mendapatkan buku yang dimaksud (Wahana Komputer ; 2010).

## II.7. 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.3. Tampilan MySQL**
**(Sumber : Antonius Nugraha Widhi Pratama ; 2010)**

## II.8. Tehnik Normalisasi

### II.8.1. Bentuk-bentuk Normalisasi

**1. Bentuk normal tahap pertama (1” Normal Form)**

Contoh yang kita gunakan di sini adalah sebuah perusahaan yang mendapatkan barang dari sejumlah pemasok. Masing-masing pemasok berada pada satu kota. Sebuah kota dapat mempunyai lebih dari satu pemasok dan masing-masing kota mempunyai kode status tersendiri.

Supaya bisa menggabungkan jumlah barang yang di pasok (qty) secara unik dengan barang (b#) dan pemasok (#p), kita menggunakan kunci utama gabungan yang tersusun dari b# dan p#. Sebuah tabel relasional secara definisi selalu berada dalam bentuk normal pertama. Semua nilai pada kolom-kolom nya adalah atomik. Ini berarti kolom-kolom tidak mempunyai nilai berulang . Gambar 2.3 menunjukkan tabel pemasok dalam 1NF.( Janner Simarmata, 2010).

| P# | Status | kota | B# | Qty |
|---|---|---|---|---|
| P1 | 20 | Yogkarta | B1 | 300 |
| P1 | 20 | Yogkarta | B2 | 200 |
| P1 | 20 | Yogkarta | B3 | 400 |
| P1 | 20 | Yogkarta | B4 | 200 |
| P1 | 20 | Yogkarta | B5 | 100 |
| P1 | 20 | Yogkarta | B6 | 100 |
| P2 | 10 | Medan | B1 | 300 |
| P2 | 10 | Medan | B2 | 400 |
| P3 | 10 | Medan | B2 | 200 |
| P4 | 20 | Yogkarta | B2 | 200 |
| P4 | 20 | Yogkarta | B2 | 300 |
| P4 | 20 | Yogkarta | B5 | 400 |

**Gambar II.4 Tabel bentuk normal pertama (1NF)**

**(Sumber : Janner Simarmata ; 2010)**

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

Definisi bentuk normal kedua menyatakan bahwa tabel dengan kunci utama gabungan hanya dapat berada pada 1NF, tetapi tidak pada 2NF. Sebuah tabel relasional berada pada bentuk normal kedua jika dia berada pada bentuk normal kedua jika dia berada pada 1NF dan setiap kolom bukan kunci yang sepenuhnya tergantung pada seluruh kolom yang membentuk kunci utama.

Tabel pemasok berada pada INF, tetapi tidak pada 2NF karena status dan kota tergantung secara fungsional hanya pada kolom p# dari kunci gabungan (p#, b#). Ini dapat digambarkan dengan membuat daftar ketergantungan fungsional;

p# : kota, status

kota : status

(p#, b#) : qty

Pada contoh, kita memindahkan kolom p#, status,dan kota ke tabel baru yang disebut PEMASOK2. kolom p# menjadi kunci utama tabel ini. Gambar 2.4 menunjukkan hasilnya. (Janner Simarmata ; 2011).

Pemasok2

| P# | Status | Kota |
|---|---|---|
| P1 | 20 | Yogkarta |
| P2 | 10 | Medan |
| P3 | 10 | Medan |
| P4 | 20 | Yogkarta |
| P5 | 30 | Bandung |

Barang

| P# | B# | Qty |
|---|---|---|
| P1 | B1 | 300 |
| P1 | B2 | 200 |
| P1 | B3 | 400 |
| P1 | B4 | 200 |
| P1 | B5 | 100 |
| P2 | B1 | 300 |
| P2 | B2 | 400 |
| P3 | B2 | 200 |
| P4 | B2 | 200 |
| P4 | B4 | 300 |
| P4 | B5 | 400 |

**Gambar II.5 Tabel bentuk normal kedua (1NF)**

**(Sumber : Janner Simarmata ; 2010)**

### 3. Bentuk normal tahap ketiga (3rd normal form)

Bentuk normal ketiga mengharuskan semua kolom pada tabel relasional tergantung hanya pada kunci utama. Secara definisi, sebuah tabel berada pada bentuk normal ketiga (3NF) jika tabel sudah berada pada 2NF dan setiap kolom yang bukan kunci tidak tergantung secara transitif pada kunci utamanya.

Untuk mengubah PEMASOK2 menjadi 3NF, kita membuat tabel baru yang disebut KOTA_STATUS dan memindahkan kolom kota dan status ke tabel baru. Status dihapus dari tabel asal, kota tetap dibiarkan karena akan berfungsi sebagai kunci asing bagi KOTA_STATUS, dan tabel asal diberi nama baru PEMASOK_KOTA. Gambar 2.5 menunjukkan hasilnya. (Janner Simarmata ; 2010).

PEMASOK_KOTA

| P# | Kota |
|---|---|
| P1 | Yogyakarta |
| P2 | Medan |
| P3 | Medan |
| P4 | Yogyakarta |
| P5 | Bandung |

KOTA_STATUS

| Kota | Status |
|---|---|
| Yogyakarta | 20 |
| Medan | 10 |
| Bandung | 30 |
| Semarang | 40 |

**Gambar II.6 Tabel bentuk normal ketiga (3NF)**

**(Sumber : Janner Simarmata ; 2010)**

### 4. *Boyce Code Normal Form* (BCNF)

Setelah 3NF, semua masalah normalisasi hanya melibatkan tabel yang mempunyai tiga kolom atau lebih dan semua kolom adalah kunci. Banyak praktisi berpendapat bahwa menempatkan entitas pada 3NF sudah cukup karena sangat jarang entitas yang berada pada 3NF bukan merupakan 4NF dan 5NF.( Janner Simarmata ; 2010 ).

**5. Bentuk Normal Tahap Keempat dan Kelima**

Sebuah tabel relasional berada pada bentuk normal keempat (4NF) jika dia dalam BCNF dan semua ketergantungan multivalue merupakan ketergantungan fungsional. Bentuk normal keempat (4NF) didasarkan pada konsep ketergantungan multivalue (MVD).

Misalnya, ada sebuah tabel relasional R dengan kolam A, B dan C, maka R.A $\rightarrow$ R.B (kolom A menentukkan kolom B). Adalah benar jika dan hanya jika himpunan nilai B yang cocok dengan pasangan nilai A dan nilai C pada R hanya tergantung pada nilai A dan tidak tergantung pada nilai C.

Misalnya, pegawai ditugaskan kebanyak proyek dan ia mempunyai banyak keahlian. Jika kita mencatat informasi ini pada satu tabel, ketiga atribut harus digunakan sebagai kunci karena tidak ada satu atribut pun yang dapat secara unik mengidentifikasi sebauh *record*. untuk mengubah sebuah tabel dengan ketergantungan *multivalue* ke dalam 4NF, pindahkan masing-masing pasangan MVD ke tabel baru. Gambar 2.6 menunjukkan hasilnya.

PEGAWAI_PROYEK

| Peg# | Pry# |
|---|---|
| 1211 | P1 |
| 1211 | P3 |

PEGAWAI_AHLI

| Peg# | Ahli |
|---|---|
| 1211 | Analisis |
| 1211 | Perancangan |
| 1211 | Pemrograman |

**Gambar II.7 Tabel bentuk normal keempat (4NF)**

**(Sumber : Janner Simarmata ; 2010)**

Sebuah tabel berada pada bentuk normal kelima (5NF) jika ia tidak dapat mempunyai dekomposisi lossless menjadi sejumlah tabel lebih kecil. Empat bentuk normal pertama berdasarkan pada konsep ketergantungan fungsional, sedangkan bentuk normal kelima berdasarkan pada konsep ketergantungan gabungan (*join dependence*).

contoh, misalkan kita mempunyai pegawai yang menggunakan keahlian perancangan pada suatu proyek lainnya. (Janner Simarmata ; 2010).

## II.9. *Unified Modeling Language* (UML)

Menurut Windu Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.

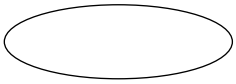
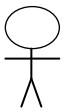
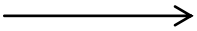
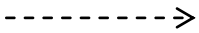
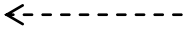
UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem.

- *Use case* Diagram

*Use case* diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem

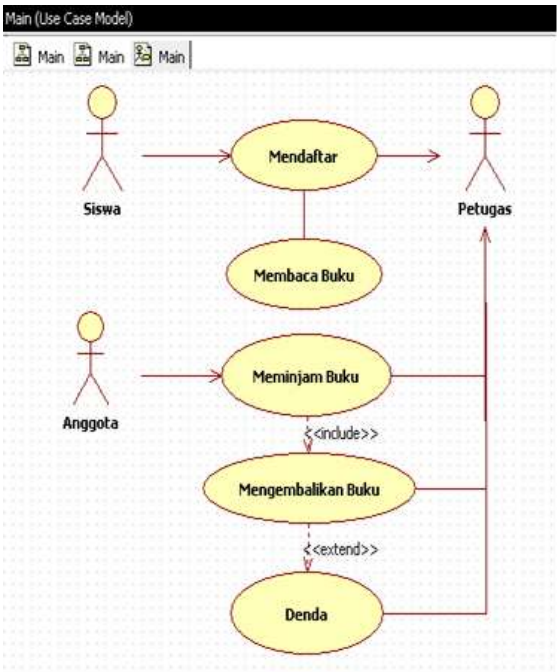
informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

**Tabel II.1. Simbol *Use Case***

| Gambar | Keterangan |
|---|---|
|  | *Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*. |
|  | Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*. |
|  | Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data. |
|  | Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem. |
|  | *Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program. |
|  | *Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi. |

**(Sumber : Windu Gata ; 2013)**

Contoh dari pembuatan *use case diagram* dapat dilihat pada gambar II.4 berikut :




**Gambar. II.8. *Use Case Diagaram***
**(Sumber : Windu Gata ; 2013)**

- *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem.

*Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/Method*), *Visibility*, tingkat akses objek eksternal kepada suatu

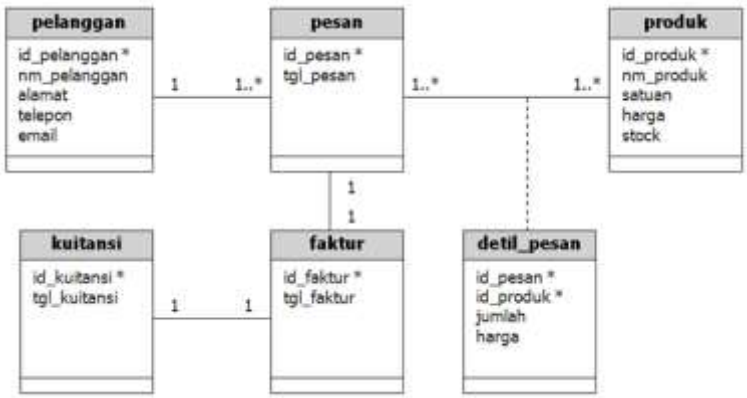
operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

**Tabel II.2. *Multiplicity Class Diagram***

| Multiplicity | Penjelasan |
|---|---|
| 1 | Satu dan hanya satu |
| 0..* | Boleh tidak ada atau 1 atau lebih |
| 1..* | 1 atau lebih |
| 0..1 | Boleh tidak ada, maksimal 1 |
| n..n | Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4 |

**(Sumber : Windu Gata ; 2013)**

Contoh dari pembuatan *use case diagram* dapat dilihat pada gambar II.7 berikut :

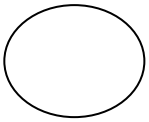
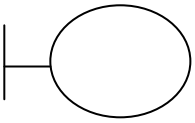
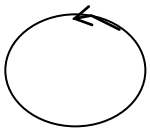
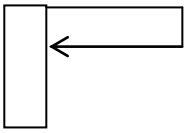



**Gambar. II.9. *Class Diagaram***
**(Sumber : Windu Gata ; 2013)**

- Diagram Urutan (*Sequence Diagram*)

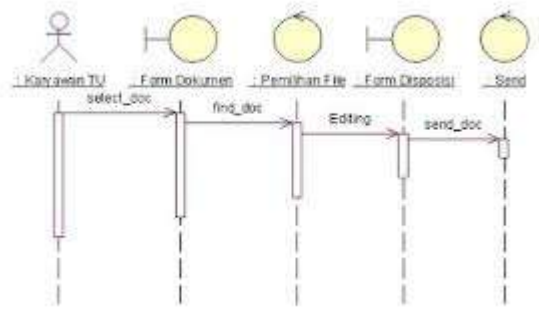
*Sequence diagram* menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

**Tabel II.3. Simbol *Sequence Diagram***

| Gambar | Keterangan |
|---|---|
|  | *Entity Class,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data. |
|  | *Boundary Class,* berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan formentry dan *form* cetak. |
|  | *Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek. |
|  | *Message*, simbol mengirim pesan antar *class*. |
|  | *Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri. |
|  | *Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi. |
|  | *Lifeline,* garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*. |

**(Sumber : Windu Gata ; 2013)**

Contoh dari pembuatan *sequence diagram* dapat dilihat pada gambar II.6 berikut :

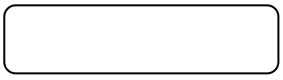
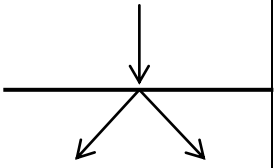


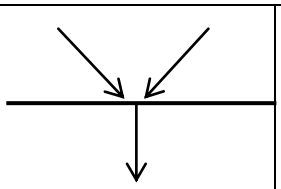
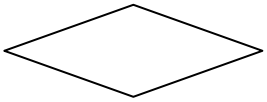

**Gambar. II.10. *Sequence Diagaram***
**(Sumber : Windu Gata ; 2013)**

- Diagram Aktivitas (*Activity Diagram*)

*Activity Diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram*, yaitu :

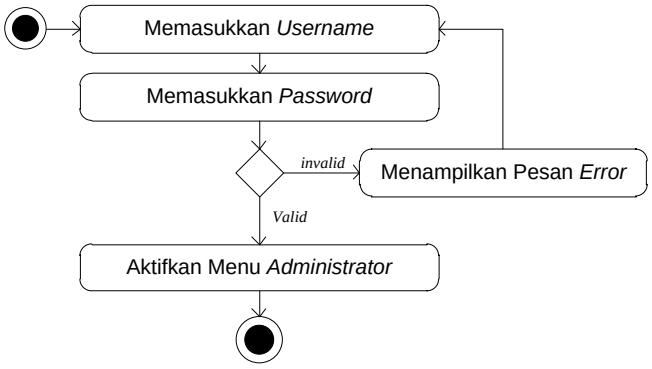
**Tabel II.4. Simbol *Activity Diagram***

| Gambar | Keterangan |
|---|---|
|  | *Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas. |
|  | *End point*, akhir aktifitas. |
|  | *Activites*, menggambarkan suatu proses/kegiatan bisnis. |
|  | *Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu. |

| | |
|---|---|
|  | *Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi. |
|  | *Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true*, *false.* |
|  | *Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa. |

**(Sumber : Windu Gata ; 2013)**

Contoh dari pembuatan *activity diagram* dapat dilihat pada gambar II.5 berikut :




**Gambar. II.11. *Activity Diagaram***
**(Sumber : Windu Gata ; 2013)**