BAB II

LANDASAN TEORI

II.1. Sistem

II.1.1. Konsep Dasar Sistem

Terdapat dua kelompok pendekataan di dalam pendefinisian sistem, yaitu sekelompok yang menekankan pada prosedur dan kelompok yang menekankan pada elemen atau komponennya. Pendekatan yang menekankan pada prosedur dan mendefenisikan sistem sebagai suatu jaringan kerja prosedur-prosedur yang saling berhubungan, berkumpul bersama-sama untuk melakukan suatu kegiatan atau untuk menyelesaikan suatu sasaran tertentu. Sedangkan pendekatan sistem yang lebih menekankan pada elemen atau komponen mendefinisikan sistem sebagai kumpulan elemen yang berinteraksi untuk mencapai suatu tujuan tertentu. kedua kelompok definisi ini adalah benar dan tidak bertentangan. Yang berbeda adalah cara pendekatannya (Tata Sutabri, 2012 : 2).

Teori sistem melahirkan konsep-konsep futuristik, antara lain yang terkenal adalah konsep sibernetika *(cyberbetics).* Konsep atau bidang kajian ilmiah ini terutama berkaitan dengan upaya-upaya untuk menerapkan berbagai disiplin ilmu, yaitu ilmu perilaku, fisikan, biologi, dan teknik. Oleh karena itu sibernetika biasanya berkaitan dengan usaha-usaha otomasi tugas-tugas yang dilakukan oleh manusia, sehingga melahirkan studi-studi tentang robotika, kecerdasan buatan *(artificial*

intelegence) dan lain sebagainya. Unsur-unsur yang mewakili suatu sistem secara umum adalah masukan *(input)*, pengolahan *(processing)* dan keluaran *(output).* Di samping itu suatu sistem senantiasa tidak terlepas dari lingkungan sekitarnya. Maka umpan balik *(feed-back)* selain dapat berasal dari *output*, juga dapat berasal dari lingkungan sistem tersebut. Organisasi dipandang sebagai suatu sistem yang juga memiliki semua unsur (Tata Sutabri, 2012 : 3).

II.1.2. Karakteristik Sistem

Model umum sebuah sistem terdiri dari *input*, proses dan *output*. Hal ini merupakan konsep sebuah sistem yang sangat sederhana mengingat sebuah sistem dapat mempunyai beberapa masukan dan keluaran sekaligus. Selain itu sebuah sistem juga memiliki karakteristik atau sifat-sifat tertentu, yang mencirikan bahwa hal tersebut bisa dikatakan sebagai suatu sistem.

Adapun karakteristik yang dimaksud adalah sebagai berikut :

1. Komponen Sistem *(Components)*

   Suatu sistem terdiri dari sejumlah komponen yang berinteraksi, yang bekerja sama membentuk satu kesatuan. Komponen-komponen sistem tersebut dapat berupa suatu bentuk subsistem. Setiap subsistem memiliki sifat-sifat sistem yang menjalankan suatu fungsi tertentu dan mempengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai sistem yang lebih besar yang disebut dengan supra sistem.

2. Batasan Sistem *(Boundary)*

Ruang lingkup sistem merupakan daerah yang membatasi antara sistem dengan sistem lainnya atau sistem dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang sebagai satu kesatuan yang tidak dapat dipisah-pisahkan.

3. Lingkungan Luar Sistem *(Evirontment)*

Bentuk apapun yang ada di luar lingkup atau batasan sistem yang mempengaruhi operasi sistem tersebut disebut dengan lingkungan luar sistem. Lingkungan luar sistem ini dapat menguntungkan dan dapat juga merugikan sistem tersebut. Lingkungan luar yang menguntungkan merupakan energi bagi sistem tersebut, yang dengan demikian lingkungan luar tersebut harus selalu dijaga dan dipelihara. Sedangkan lingkungan luar yang merugikan harus dikendalikan. Kalau tidak maka akan mengganggu kelangsungan hidup sistem tersebut.

4. Penghubung Sistem *(Interface)*

Media yang menghubungkan sistem dengan subsistem yang lain disebut dengan penghubung sistem *interface*. Penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem yang lain. Keluaran suatu subsistem akan menjadi masukan untuk subsistem yang lain dengan melewati penghubung. Dengan demikian terjadi suatu integrasi sistem yang membentuk satu kesatuan.

5. Masukan Sistem *(Input)*

Energi yang dimasukkan ke dalam sistem tersebut masukan sistem, yang dapat berupa pemeliharaan *(maintenance input)* dan sinyal *(signal input)*. Sebagai contoh di dalam suatu unit sistem komputer, "program" adalah *maintenance input* yang digunakan untuk mengoperasikan *computer*. Sementara "data" adalah *signal* yang akan diolah menjadi informasi.

6. Keluaran Sistem *(Output)*

Hasil dari energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna. Keluaran ini merupakan bagi subsistem yang lain. Seperti contoh sistem informasi, keluaran yang dihasilkan adalah informasi, di mana informasi ini dapat digunakan sebagai masukan untuk pengambilan keputusan atau hal-hal lain yang merupakan *input* bagi subsistem lainnya.

7. Pengolahan Sistem *(Procces)*

Suatu sistem dapat mempunyai suatu proses yang akan mengubah masukan menjadi keluaran. Sebagai contoh, sistem akuntansi, sistem ini akan mengolah data transaksi menjadi laporan-laporan yang dibutuhkan oleh pihak manajemen.

8. Sasaran Sistem *(Objective)*

Suatu sistem memiliki tujuan dan sasaran yang pasti dan bersifat deterministik. Kalau suatu sistem tidak memiliki sasaran, maka operasi sistem tidak ada gunanya. Suatu sistem dikatakan berhasil bila mengenai sasaran atau tujuan yang telah direncanakan (Tata Sutabri, 2012 : 13-14).

II.1.3. Pengertian Sistem

Secara sederhana sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau variabel yang terorganisasi, saling berinteraksi, saling bergantung satu sama lain dan terpadu (Tata Sutabri, 2012 : 3).

Gordon B. Davis dalam bukunya menyatakan bahwa sistem bisa berupa abstrak atau fisik. Sistem yang abstrak adalah susunan gagasan-gagasan atau konsepsi yang teratur yang saling bergantung. Sedangkan sistem yang bersifat fisik adalah serangkaian unsur yang bekerja sama untuk mencapai suatu tujuan (Tata Sutabri, 2012 : 6).

Sedangkan Norman L. Enger menyatakan bahwa suatu sistem dapat terdiri atas kegiatan-kegiatan yang berhubungan guna mencapai tujuan-tujuan perusahaan seperti pengendalian inventaris atau penjadwalan produksi (Tata Sutabri, 2012 : 7).

II.2. Sistem Pendukung Keputusan (*Decision Support System*)

II.2.1. Defenisi Sistem Pendukung Keputusan

Sistem Pendukung Keputusan merupakan penggabungan sumber-sumber kecerdasan individu dengan kemampuan komponen untuk memperbaiki kualitas keputusan. Sistem Pendukung Keputusan juga merupakan sistem informasi berbasis computer untuk manajemen pengambilan keputusan yang menangani masalah-masalah semi struktur.

Dengan pengertian diatas dapat dijelaskan bahwa sistem pendukung keputusan bukan merupakan alat pengambilan keputusan, melainkan merupakan

sistem yang membantu pengambil keputusan dengan melengkapi mereka dengan informasi dari data yang telah diolah dengan relevan dan diperlukan untuk membuat keputusan tentang suatu masalah dengan lebih cepat dan akurat. Sehingga sistem ini tidak dimaksudkan untuk menggantikan pengambilan keputusan dalam proses pembuatan keputusan (Rika Yunitarini, 2013 : 45-46).

Sistem Pendukung Keputusan (SPK) atau Decision Support System (DSS) adalah sebuah sistem yang mampu memberikan kemampuan pemecahan masalah maupun kemampuan pengkomunikasian untuk masalah dengan kondisi semi terstruktur dan tak terstruktur. Sistem ini digunakan untuk membantu pengambilan keputusan dalam situasi semi terstruktur dan situasi yang tidak terstruktur, di mana tak seorangpun tahu secara pasti bagaimana keputusan seharusnya dibuat (Anton Setiawan Honggowibowo, 2015 : 34).

II.2.2. Karakteristik Sistem Pendukung Keputusan

Karakteristik dalam sistem pendukung keputusan, antara lain (Erliza Septia Nagara dan Rini Nurhayati, 2015: 4) :

1. Sistem Pendukung Keputusan dirancang untuk membantu pengambil keputusan dalam memecahkan masalah yang sifatnya semi terstruktur ataupun tidak terstruktur dengan menambahkan kebijaksanaan manusia dan informasi komputerisasi.
2. Dalam proses pengolahannya, sistem pendukung keputusan mengkombinasikan penggunaan model-model analisis dengan teknik

pemasukan data konvensional serta fungsi-fungsi pencari/interogasi informasi.

3. Sistem Pendukung Keputusan, dirancang sedemikian rupa sehingga dapat digunakan/dioperasikan dengan mudah.
4. Sistem Pendukung Keputusan dirancang dengan menekankan pada aspek fleksibilitas serta kemampuan adaptasi yang tinggi.

II.2.3. Komponen Sistem Pendukung Keputusan

Untuk dapat menerapkan SPK, ada 4 komponen subsistem yang harus disediakan yaitu (Syukron Hidayat dan Imam Mukhlash, 2015 45) :

1. Subsistem manajemen data

   Subsistem ini menyediakan data bagi sistem, termasuk didalamnya basis data. Berisi data yang relevan untuk situasi dan diatur oleh perangkat lunak yang disebut *Database Management System* (DBMS).

2. Subsistem manajemen model

   Subsistem ini berfungsi sebagai pengelola berbagai model, mulai dari model keuangan, statistik, matematik, atau model kuantitatif lainnya yang memiliki kemampuan analisis dan manajemen perangkat lunak yang sesuai. Perangkat lunak ini sering disebut *Model Base Management System* (MBMS).

3. Subsistem manajemen pengetahuan

   Subsistem ini mendukung berbagai subsistem lainnya, atau dapat dikatakan berperan sebagai komponen yang independen. Subsistem ini

menyediakan intelegensi untuk menambah pertimbangan pengambil keputusan.

4. Subsistem manajemen antar muka pengguna

Subsistem ini berupa tampilan yang disediakan yang mampu mengintegrasikan sistem terpasang dengan pengguna secara interaktif. Melalui subsistem ini pengguna dapat berkomunikasi dengan sistem pendukung keputusan serta memerintah sistem pendukung keputusan.

II.3. Metode SMART

II.3.1. Pengertian *Simple Multi-Atribute Rating Technique* (SMART)

Metode SMART merupakan metode pengambilan keputusan multi kriteria yang dikembangkan oleh Edward pada tahun 1977. SMART merupakan teknik pengambilan keputusan multi kriteria ini didasarkan pada teori bahwa setiap alternatif terdiri dari sejumlah kriteria yang memiliki nilai-nilai dan setiap kriteria memiliki bobot yang menggambarkan seberapa penting ia dibandingkan dengan kriteria lain. Pembobotan ini digunakan untuk menilai setiap alternatif agar diperoleh alternatif terbaik (Suryanto, 2015 : 26).

Pada hakekatnya Simple Multy Attribute Rating (SMART) merupakan suatu model pengambil keputusan yang komprehensif dengan memperhitungkan hal-hal yang bersifat kualitatif dan kuantitatif. Dalam model pengambilan keputusan dengan SMART pada dasarnya berusaha menutupi setiap kekurangan dari model-model tanpa komputerisasi sebelumnya. SMART juga memungkinkan ke struktur suatu

sistem dan lingkungan kedalam komponen saling berinteraksi dan kemudian menyatukan mereka dengan mengukur dan mengatur dampak dari komponen kesalahan sistem (Eva Yulianti, 2015 : 56).

SMART mengunakan *linier adaptif model* untuk meramal nilai setiap alternatif. SMART lebih banyak digunakan karena kesederhanaannya dalam merespon kebutuhan pembuat keputusan dan caranya menganalisa respon. Analisis yang terbaik adalah transparan sehingga metode ini memberikan pemahaman masalah yang tinggi dan dapat diterima oleh pembuat keputusan. Pembobotan pada SMART mengunakan skala 0 sampai 1, sehingga mempermudah perhitungan dan perbandingan nilai pada masing-masing alternatif (Rika Yunitarini, 2013 : 46).

Model fungsi utility linier yang digunakan dalam SMART yaitu :

$$Maximize \sum_{j}^{k} = 1\, w_j . u_{ij} , \forall i = 1, \dots , n$$

Di mana :

- w_j adalah nilai pembobotan kriteria ke-*j* dari *k* kriteria.
- u_{ij} adalah nilai *utility* alternatif *i* pada kriteria *j*.
- Pemilihan keputusan adalah mengidentifikasi mana dari *n* alternatif yang mempunyai nilai fungsi terbesar.
- Nilai fungsi ini juga dapat digunakan untuk meranking *n* alternatif (Anton Setiawan Honggowibowo, 2015 : 32).

II.3.2. Teknik SMART

Adapun langkah-langkah dalam proses perhitungan metode *SMART* dapat ditunjukkan sebagai berikut (Kustiyahningsih, dkk, 2012 : 22-23).

1. Langkah 1 : menentukan jumlah kriteria.
2. Langkah 2 : sistem secara *default* memberikan skala 0-100 berdasarkan prioritas yang telah diinputkan kemudian dilakukan normalisasi.

$$\text{Normalisasi} = \frac{w_j}{\sum w_j} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots . (\text{II}.1)$$

Keterangan :

- w_j : bobot suatu kriteria
- $\sum w_j$: total bobot semua kriteria

3. Langkah 3 : memberikan nilai kriteria untuk setiap alternatif.
4. Langkah 4 : hitung nilai *utility* untuk setiap kriteria masing-masing.

$$u_i(a_i)=100\ \frac{(C_{out\ i}-C_{min})}{(C_{max}-C_{min})}\% \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (\text{II}.2)$$

Keterangan :

- $u_i(a_i)$: nilai *utility* kriteria ke-1 untuk kriteria ke-i
- C_{max} : nilai kriteria maksimal
- C_{min} : nilai kriteria minimal
- $C_{out\ i}$: nilai kriteria ke-i

5. Langkah 5 : hitung nilai akhir masing-masing.

$$u\ (a_i) = \sum_{J=1}^{m} w_j u_i(a_i) \ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (\text{II}.3)$$

II.3.3. Kelebihan Metode SMART

SMART memiliki beberapa kelebihan dibandingkan dengan metode pengambilan keputusan yang lain yaitu (Rika Yunitarini, 2013 : 46-47) :

1. Mungkin melakukan penambahan / pengurangan alternatif.

   Pada metode SMART penambahan atau pengurangan alternatif tidak akan mempengaruhi perhitungan pembobotan karena setiap penilaian alternatif tidak saling bergantung.

2. Sederhana

   Perhitungan pada metode SMART lebih sederhana sehingga tidak diperlukan perhitungan matematis yang rumit dengan pemahaman matematika yang kuat.

3. Transparan

   Proses dalam menganalisa alternatif dan kriteria dalam SMART dapat dilihat oleh *user*, sehingga *user* dapat memahami bagaimana alternatif tertentu dapat dipilih. Alasan-alasan bagaimana alternatif itu dipilih dapat dilihat dari prosedur-prosedur yang dilakukan dalam SMART mulai dari penentuan kriteria, pembobotan, dan pemberian nilai pada setiap alternatif.

4. Fleksibelitas Pembobotan

   Pembobotan yang dipakai di dalam metode SMART ada 3 jenis, yaitu pembobotan secara langsung (*direct weighting*), pembobotan swing (*swing weighting*) dan pembobotan centroid (*centroid weighting*).

II.4. Basis Data (Database)

II.4.1. Defenisi Basis Data

Basis data dapat didefinisikan sebagai himpunan kelompok data yang saling berhubungan yang diorganisasikan sedemikian rupa agar kelak dapat dimanfaatkan kembali dengan cepat dan mudah. Prinsip utamanya adalah pengaturan data. Tujuan utamanya kemudahan dan kecepatan dalam pengambilan kembali data (Priyanto Hidayatullah, 2012 : 137).

Sebuah basis data adalah sebuah kumpulan data yang saling berhubungan secara logis, dan merupakan sebuah penjelasan dari data tersebut, yang didesain untuk menemukan data yang dibutuhkan oleh sebuah organisasi. Di dalam basis data, semua data diintegrasikan dengan departemen dan pemakai. Basis data tidak hanya memegang data operasional organisasi, tetapi juga penjelasan mengenai data tersebut. Karena alasan tersebut basis data dapat juga dideskripsikan sebagai kumpulan data yang saling terintegrasi. Basis data juga merupakan sekumpulan elemen data terintegrasi yang secara logika saling berhubungan. Basis data mengonsolidasikan berbagai catatan yang terlebih dahulu disimpan dalam file-file terpisah ke dalam satu gabungan umum elemen data yang menyediakan data untuk banyak aplikasi. Elemen data mendeskripsikan entitas-entitas dan hubungan antara entitas-entitas tersebut (Indrajani, 2015 : 70).

II.4.2. Tujuan Basis Data

Secara lebih lengkap pemanfaatan basis data dilakukan untuk memenuhi tujuan berikut ini (Priyanto Hidayatullah, 2012 : 138) :

1. Kecepatan dan kemudahan (*Speed*).
2. Efisiensi ruang penyimpanan (*Space*).
3. Keakuratan (*Accuracy*).
4. Ketersediaan (*Availability*).
5. Kelengkapan (*Completeness*).
6. Keamanan (*Security*).
7. Pemakaian bersama (*Share ability*).

II.5. Kamus Data (Data Dictionary)

Kamus data adalah katalog fakta tentang data dan kebutuhan informasi suatu sistem informasi. Kamus data terdapat pada tahapan analisis dan perancangan. Pada tahap analisis, kamus data berfungsi untuk mendefinisikan data yang mengalir pada sistem. Sedangkan pada tahap perancangan, kamus data ini digunakan untuk merancang masukan dan keluaran seperti laporan serta basis data (Indrajani, 2015 : 30-31).

Sumber kamus data, yaitu :

1. Data Store (File-File)
2. Data Flow (Aliran Data)

3. Data element yang dinyatakan dalam spesifikasi data dan berasal dari file.

Berikut notasi-notasi yang digunakan dalam kamus data terlihat pada Tabel II.1.

Tabel II.1 Notasi Kamus Data

Notasi	**Keterangan**
=	*Is composed of*
+	*And*
()	*Optional (may be present or absent)*
{}	*Iteration*
[]	*Select one of several alternative choices*
**	*Comment*
@	*Identifier (key field) for a store*
|	*Separates alternative choices in the [] construct*

(Sumber: Indrajani, 2015 : 31)

Contoh kamus data, antara lain :

name= courtesy-title + first-name + (*middle-name*) *+ last-name*

courtesy-title=[*Mr.* | *Miss* | *Mrs.* | *Ms.* | *Dr.* | *Professor*]

first-name={*legal-character*}

middle-name={*legal-character*}

last-name= {*legal-character*}

legal-character=[A-Z | a-z | 0-9 | ` | - | |]

II.6. Normalisasi

II.6.1. Pengertian Normalisasi

Normalisasi adalah suatu teknik dengan pendekatan *bottom-up* yang digunakan untuk membantu mengidentifikasikan hubungan. Dimulai dari menguji hubungan, yaitu *functional dependencies* antara atribut. Pengertian lainnya adalah suatu teknik yang menghasilkan sekumpulan hubungan dengan sifat-sifat yang diinginkan dan memenuhi kebutuhan pada perusahaan (Indrajani, 2015 : 7).

Proses normalisasi merupakan proses pengelompokkan elemen data menjadi tabel-tabel yang menunjukkan entitas dan relasinya. Proses ini selalu diuji pada beberapa kondisi. Apakah ada kesulitan pada saat menambah (*insert*), menghapus (*delete*), mengubah (*update*), atau membaca (*retrieve*) pada satu *database*. Bila ada kesulitan pada pengujian tersebut maka relasi dapat dipecah dalam beberapa tabel lagi (Tata Sutabri, 2012 : 138).

II.6.2. Tujuan Normalisasi

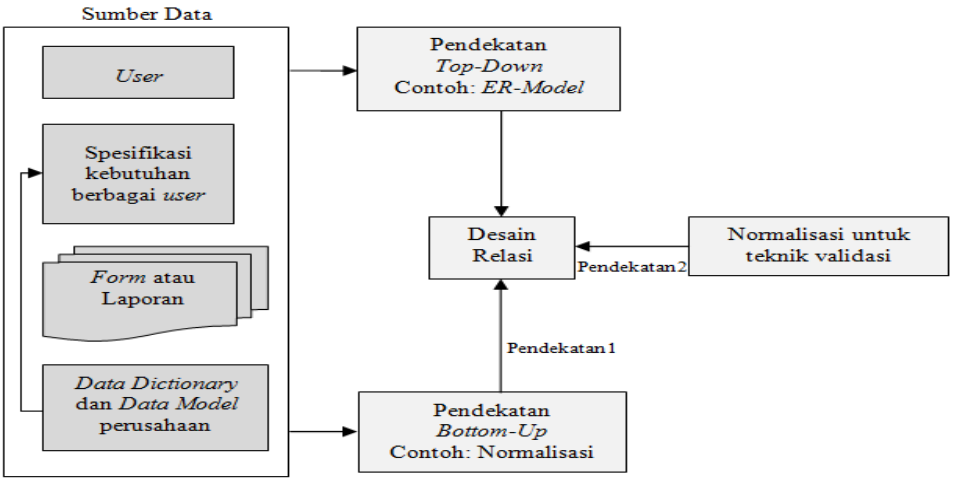
Tujuan utama normalisasi adalah mengidentifikasikan kesesuaian hubungan yang mendukung data untuk memenuhi kebutuhan perusahaan. Adapun karakteristik hubungan tersebut mencakup (Indrajani, 2015 : 7) :

1. Minimal jumlah atribut yang diperlukan untuk mendukung kebutuhan perusahaan.
2. Atribut dengan hubungan logika yang menjelaskan mengenai *functional dependencies*.

3. Minimal duplikasi untuk tiap atribut.

II.6.3. Perananan Normalisasi Dalam Perancangan Basis Data

Normalisasi adalah suatu teknik formal yang dapat digunakan dalam perancangan basis data. Peranan normalisasi dalam hal ini adalah dalam penggunaan pendekatan *bottom-up* dan teknik validasi. Teknik validasi digunakan untuk memeriksa, apakah struktur relasi yang dihasilkan oleh *ER-modeling* itu baik atau tidak baik (Indrajani, 2015 : 7). Peranan normalisasi dalam perancangan basis data dapat lebih jelas terlihat pada Gambar II.1.



Gambar II.1. Peranan Normalisasi Dalam Perancangan Basis Data

(Sumber : Indrajani, 2015 : 7)

Dari Gambar II.1. dapat dilihat sumber data terdiri dari *user*, spesifikasi kebutuhan berbagai *user*, berbagai *form* atau laporan, *data dictionary*, dan *data model* perusahaan. Kemudian terdapat pendekatan *top-down* dan *bottom-up*, di mana

pendekatan tersebut nantinya menghasilkan desain relasi. Lalu peranan normalisasi pada *bottom-up* dan teknik validasi.

Adapun beberapa bentuk normalisasi yang biasa digunakan yaitu (Indrajani, 2015 : 8-10) :

1. *Un-normalized Form* (UNF)

   Merupakan suatu tabel yang berisikan satu atau lebih grup yang berulang. Membuat tabel yang *un-normalized*, yaitu dengan memindahkan data dari sumber informasi. Contoh: nota penjualan yang disimpan ke dalam format tabel dengan baris dan kolom.

2. *First Normal Form* (1NF) atau Normalisasi Tingkat 1

   Merupakan sebuah relasi di mana setiap baris dan kolom berisikan satu dan hanya satu nilai. Adapun proses UNF ke 1NF yaitu :

   a. Tentukan satu atau kumpulan atribut sebagai kunci untuk tabel *un-normalized*.
   b. Identifikasikan grup yang berulang dalam tabel *un-normalized* yang berulang untuk kunci atribut.
   c. Hapus grup yang berulang dengan cara:
      1) Masukkan data yang semestinya ke dalam kolom yang kosong pada baris yang berisikan data yang berulang (*flattening the table*).
      2) Menggantikan data yang ada dengan menulis ulang dari kunci atribut yang sesungguhnya ke dalam relasi terpisah.

3. *Second Normal Form* (2NF) atau Normalisasi Tingkat 2

Berdasarkan pada konsep *full functionaldependency*, yaitu A dan B merupakan atribut sebuah relasi. B dikatakan *fully dependent* terhadap A jika B *functionally dependent* pada A tetapi tidak pada *proper subset* dari A. 2NF merupakan sebuah relasi dalam 1NF dan setiap atribut *non primary key* bersifat *fully functionally dependent* pada *primary key.* Adapun proses 1NF ke 2NF yaitu :

a. Identifikasi *primary key* untuk relasi 1NF.
b. Identifikasi *functional dependencies* dalam relasi.
c. Jika terdapat *partial dependencies* terhadap *primary key*, maka hapus dengan menempatkan dalam relasi yang beru bersama dengan salinan determinannya.

4. *Third Normal Form* (3NF) atau Normalisasi Tingkat 3

Berdasarkan pada konsep *transitive dependency*, yaitu suatu kondisi di mana A, B, dan C merupakan atribut sebuah relasi, maka A → B dan B → C, maka *transitively dependent* pada A melalui B (jika A tidak *functionally dependent* pada B atau C). 3NF adalah sebuat relasi dalam 1NF dan 2NF, di mana tidak terdapat atribut *non primary key* yang bersifat *transitively dependent* pada *primary key*. Adapun proses 2NF ke 3NF yaitu :

a. Identifikasikan *primary key* dalam relasi 2NF.
b. Identifikasikan *functional dependencies* dalam relasi.

c. Jika terdapat *transitive dependencies* terhadap *primary key*, hapus dengan menempatkannya dalam relasi yang baru bersama dengan salinan determinannya.

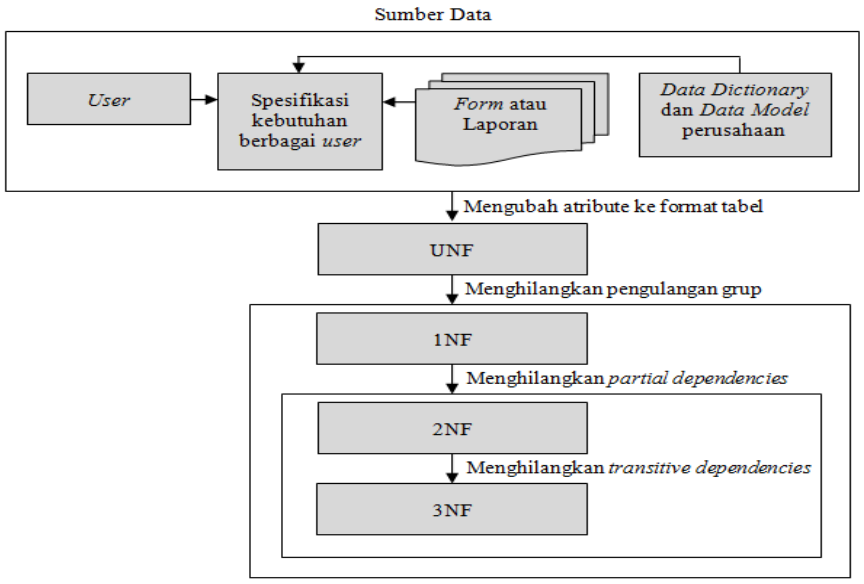
5. *Boyce-Code Normal Form* (BCNF)

Berdasarkan pada *functional dependencies* yang dimasukkan ke dalam hitungan seluruh *candidate key* dalam suatu relasi. Bagaimana pun BCNF juga memiliki batasan-batasan tambahan disamakan dengan definisi umum dari 3NF. Suatu relasi dikatakan BCNF, jika dan hanya jika setiap determinan merupakan *candidate key*. Perbedaan antara 3NF dan BCNF yaitu untuk *functional dependency* A → B, 3NF memungkinkan *dependency* ini dalam satu relasi jika B adalah atribut *primary key* dan A bukan merupakan *candidate key*. Sedangkan BCNF menetapkan dengan jelas bahwa untuk *dependency* ini agar ditetapkan dalam relasi A, maka A harus merupakan *candidate key*. Setiap relasi dalam BCNF juga merupakan 3NF, tetapi relasi dalam 3NF belum tentu termasuk ke dalam BCNF. Dalam BCNF kesalahan jarang sekali terjadi, kesalahan dapat terjadi pada relasi yang :

a. Terdiri atas 2 atau lebih *composite candidate key*.

b. *Candidate key overlap*, sedikitnya satu atribut.

Adapun bentuk diagram proses normalisasi dapat lebih jelas dilihat pada Gambar II.2.



Gambar II.2. Diagram Proses Normalisai

(Sumber : Indrajani, 2015 : 8)

II.7. *Entity Relationship Diagram* (ERD)

Entity Relational (ER) *Modeling* adalah sebuah pendekatan *top-bottom* dalam perancangan basis data yang dimulai dengan mengidentifikasikan data-data terpenting yang disebut dengan entitas dan hubungan antara entitas-entitas tersebut yang digambarkan dalam suatu model. Karena terdapat keterbatasan pada ER Model, maka terdapat pengembangan penambahan konsep semantik pada ER yang disebut *Enhanced Entity Relational* (EER) Model (Indrajani, 2015 : 17).

Entity Relationship Diagram (ERD) memiliki dua komponen utama yaitu Entitas (*Entity*) dan Relasi (*Relation*). Kedua komponen ini masing-masing dilengkapi dengan sejumlah atribut yang mempresentasikan seluruh fakta yang ada di dunia nyata (Eka, 2014 : 30). Entitas adalah sesuatu atau objek dalam dunia nyata

yang dibedakan dari objek lain, misalnya: mahasiswa dan matakuliah. Entitas digambarkan dalam basis data dengan kumpulan atribut, misalnya: NIM, nama, alamat, dan kota. Relasi adalah hubungan antara beberapa entitas, misalnya: relasi menghubungkan mahasiswa dengan matakuliah yang diambilnya (D. Tri Octafian, 2011 : 150).

Adapun simbol daripada *Entity Relationship Diagram* dapat dilihat pada Tabel II.2.

Tabel II.2 *Entity Relationship Diagram* (ERD)

No.	Simbol	Keterangan Fungsi
1.		Persegi panjang menyatakan himpunan entitas adalah orang, kejadian, atau berada di mana data akan dikumpulkan.
2.		Atribut merupakan informasi yang diambil tentang sebuah entitas.
3.		Belah ketupat menyatakan himpunan relasi merupakan hubungan antar entitas.
4.		Garis sebagai penghubung antara himpunan, relasi, dan himpunan entitas dengan atributnya.

(Sumber: Ibnu Aqil, 2010 : 6)

Pemetaan kardinalitas menyatakan jumlah entitas di mana entitas lain dapat dihubungkan ke entitas tersebut melalui sebuah himpunan relasi. Kardinalitas relasi yang terjadi di antara dua himpunan entitas di antaranya adalah (D. Tri Octafian, 2011 : 151-152) :

1. *One to One*

Sebuah entitas pada A berhubungan dengan paling banyak satu entitas pada B dan sebuah entitas pada B berhubungan paling banyak satu entitas pada A. Contoh: pada pengajaran privat, satu guru satu siswa. Seorang guru mengajar seorang siswa, seorang siswa diajar oleh seorang guru. Hubungan *One to One* dapat dilihat pada Gambar II.3.

Gambar II.3. Hubungan *One to One*

(Sumber: D. Tri Octafian, 2011 : 151)

2. *One to Many* / *Many to One*

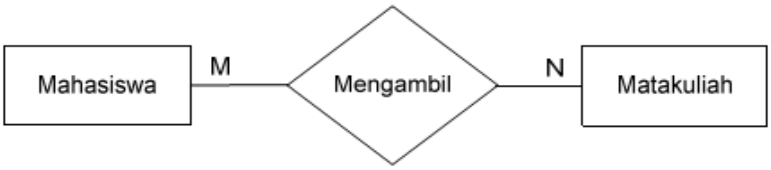
Sebuah entitas pada A berhubungan dengan lebih dari satu entitas pada B dan sebuah entitas pada B berhubungan dengan paling banyak satu entitas pada A, atau sebaliknya (*Many to One*). Contoh: dalam satu perusahaan, satu bagian mempekerjakan banyak pegawai. Satu bagian mempekerjakan banyak pegawai, satu pegawai kerja dalam satu bagian. Hubungan *One to Many* dapat dilihat pada Gambar II.4.

Gambar II.4. Hubungan *One to Many*

(Sumber: D. Tri Octafian, 2011 :152)

3. *Many to Many*

Sebuah entitas pada A berhubungan dengan lebih dari satu entitas pada B dan sebuah entitas pada B berhubungan dengan lebih dari satu entitas pada A. Contoh: dalam satu universitas, seorang mahasiswa dapat mengambil banyak matakuliah. Satu mahasiswa mengambil banyak matakuliah dan satu matakuliah diambil banyak mahasiswa. Hubungan *Many to Many* dapat dilihat pada Gambar II.5.



Gambar II.5. Hubungan *Many to Many*
(Sumber: D. Tri Octafian, 2011 : 152)

II.8. *Unified Modeling Language* (UML)

II.8.1. Pengertian UML

Unified Modelling Language (UML) merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek. UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam sistem melalui jumlah

elemen grafis yang bisa dikombinasikan menjadi diagram (Rosana Junita Sirait, et al., 2015).

UML dianggap sebagai industri bahasa pemodelan standar dengan notasi grafis yang kaya dengan seperangkat diagram dan elemen. Hal ini digunakan untuk menentukan, memvisualisasikan, memodifikasi, membangun, dan mendokumentasikan bentuk dari perangkat lunak-intensif berorientasi objek dalam membangun sistem (Lee, Sunguk, 2012 :157).

II.8.2. *Fungsi Unified Modeling Language* (UML)

Unified Modeling Language (UML) biasa digunakan untuk (Aris, et al., 2015).

1. Menggambarkan batasan sitem dan fungsi -fungsi sistem secara umum, dibuat dengan *use case* dan *actor*.
2. Menggambarkan kegiatan atau proses bisnis yang dilaksanakan secara umum, dibuat dengan *interaction diagrams*.
3. Menggambarkan representasi struktur *static* sebuah sistem dalam betuk *class diagrams*.
4. Membuat model behavior “yang menggambarkan kebiasaan atau sifat sebuah sistem” dengan *state transition diagrams*.
5. Menyatakan arsitektur implementasi fisik menggunakan *component and development.*
6. Menyampaikan atau memperluas *fungsionality* dengan *stereotypes*.

UML merupakan salah satu alat bantu yang sangat handal dalam bidang pengembangan sistem berorientasi objek. Karena UML menyediakan bahasa pemodelan visual yang memungkinkan pengembang sistem membuat *blue print* atas visinya dalam bentuk yang baku. UML berfungsi sebagai jembatan dalam mengkomunikasikan beberapa aspek dalam sistem melalui sejumlah elemen grafis yang bisa dikombinasikan menjadi diagram. UML mempunyai banyak diagram yang dapat mengakomodasikan berbagai sudut pandang dari suatu perangkat lunak yang akan dibangun. Diagram-diagram tersebut digunakan untuk :

1. Mengkomunikasikan ide.
2. Melahirkan ide-ide baru dan peluang-peluang baru.
3. Menguji ide dan membuat prediksi.
4. Memahami struktur dan relasi-relasinya.

II.8.3. Diagram-Diagram *Unified Modeling Language* (UML)

UML memiliki beberapa jenis diagram, namun dalam penulisan skripsi ini penulis hanya menggunakan 4 jenis diagram UML yaitu: *Use Case Diagram*, *Activity Diagram*, *Class Diagram* dan *Sequence Diagram*. Adapun penjelasan dari diagram-diagram tersebut adalah sebagai berikut :

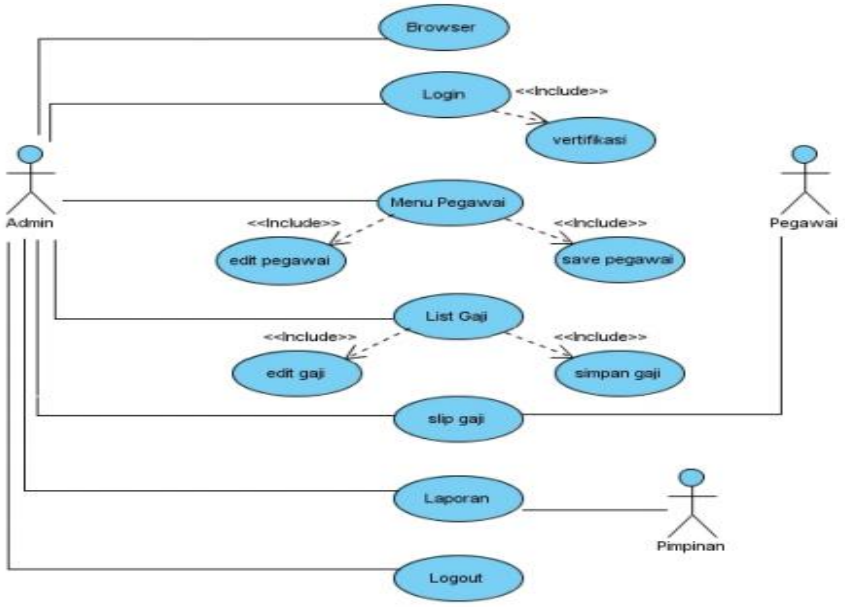
1. *Use Case Diagram*

   Use case diagram merupakan suatu diagram yang berisi *use case*, actor, serta *relationship* di antaranya. Use case diagram merupakan titik awal yang baik dalam memahami dan menganalisis kebutuhan apa saja yang diperlukan

dari suatu sistem. Jadi, dapat digambarkan dengan detail bagaimana suatu sistem memproses atau melakukan sesuatu, bagaimana cara actor akan menggunakan sistem, serta apa saja yang dapat dilakukan terhadap suatu sistem. Notasi yang digunakan dalam Use Case adalah persegi panjang yang merupakan system boundary, oval yang merupakan suatu proses, dan gambar orang yang berinteraksi dalam proses tersebut (Indrajani, 2015 : 45). *Use case* diagram digunakan untuk penggambaran *use case* statik dari suatu sistem. *Use case* diagram penting dalam mengatur dan memodelkan kelakuan dari suatu sistem. *Use case* menjelaskan apa yang dilakukan sistem (atau subsistem) tetapi tidak menspesifikasi cara kerjanya. *Flow of event* digunakan untuk menspesifikasi cara kerjanya kelakuan dari *use case*. *Flow of event* menjelaskan *use case* dalam bentuk tulisan dengan sejelas-jelasnya, dintaranya bagaimana, kapan *use case* dimulai dan berakhir, ketika *use case* berinteraksi dengan aktor, objek apa yang digunakan, alur dasar dan alur alternatif. Terdapat beberapa simbol dalam menggambarkan diagram *use case*, yaitu *use cases*, aktor dan relasi (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015).

Use Case memiliki dua istilah, yaitu :

1. *System use case*; interaksi dengan sistem.
2. *Business use case*; interaksi bisnis dengan konsumen atau kejadian nyata. Adapun contoh penggunaan use case diagram dapat dilihat pada Gambar II.6.



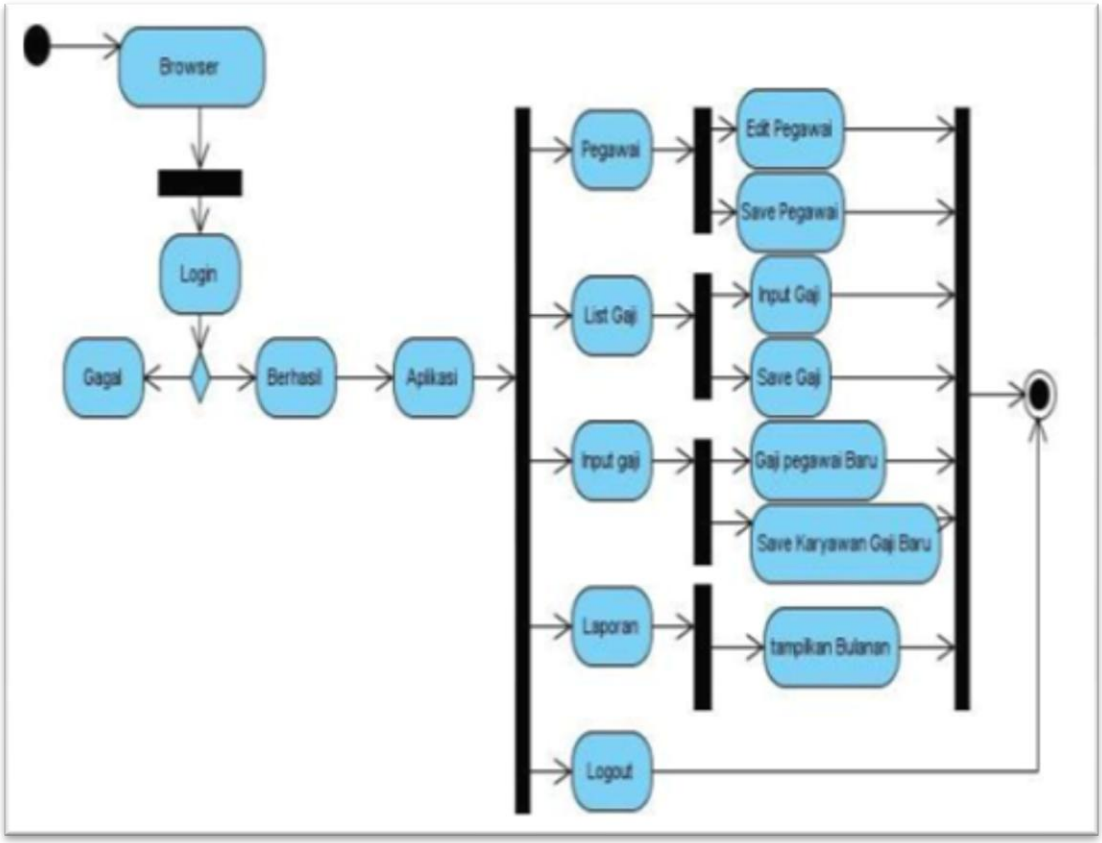
Gambar II.6. ***Use Case Diagram***

(Sumber : Aris, et al., 2015)

2. Activity Diagram

Diagram ini digunakan untuk menganalisis behaviour dengan use case yang lebih kompleks dan menunjukkan interaksi-interaksi di antara mereka satu sama lain. Activity diagram sebenarnya memiliki kesamaan dengan *statechart activity diagram* banyak digunakan pada saat peralihan antara analisis dan fase desain. Umumnya digunakan untuk menggambarkan sistem interaktif yang "real time". Diagram ini bersifat optional dalam suatu perancangan sistem. Kumpulan sub-state yang dikelompokkan ke dalam sebuah state disebut sebaga composite state (Indrajani, 2015: 46). *Activity* diagram memperlihatkan alur langkah demi langkah dalam suatu proses.

Suatu aktivitas menunjukkan sekumpulan aksi (secara sekuensial atau bercabang dari satu aksi ke aksi lain), dan nilai yang dihasilkan atau digunakan oleh aksi-aksi yang terjadi. Activity diagram ditunjukkan untuk memodelkan fungsi dari suatu sistem dan menekankan pada alur dari kontrol didalam pelaksanaan dari suatu tindakan (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015). Adapun contoh penggunaan *activity diagram* dapat dilihat pada Gambar II.7.

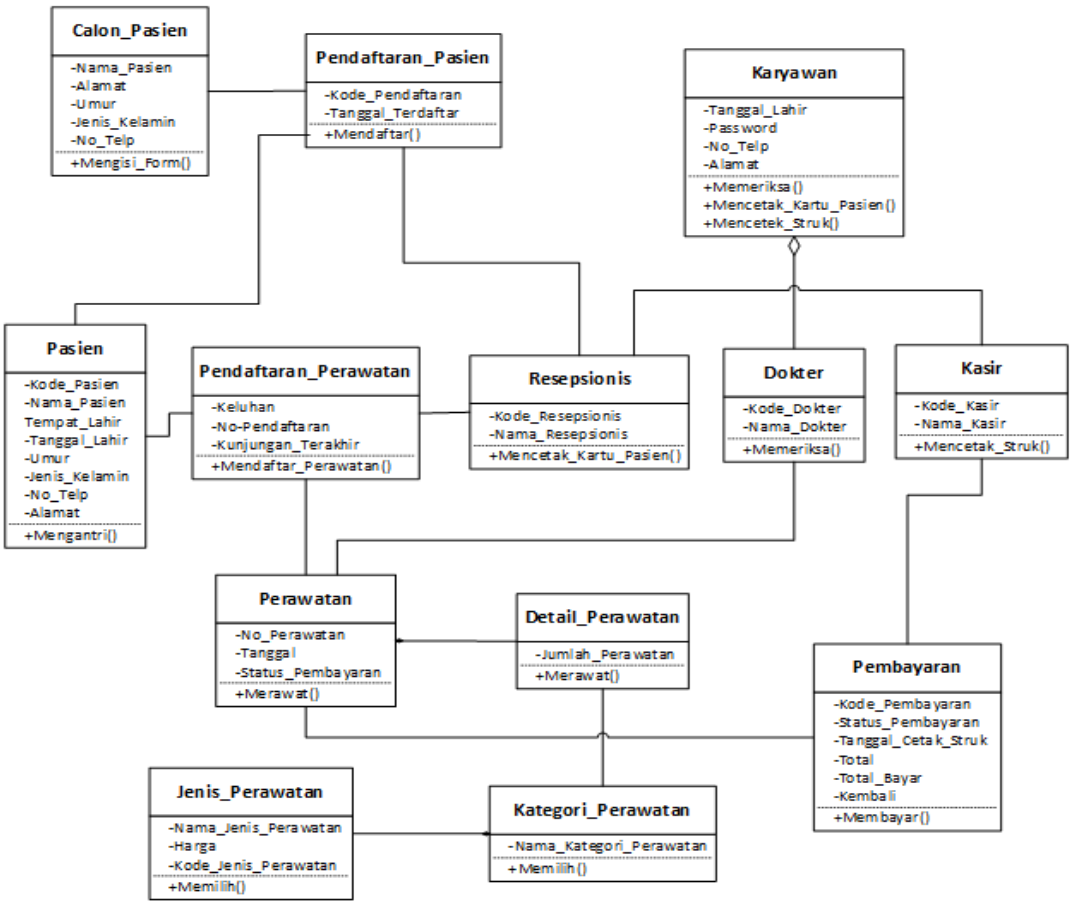


Gambar II.7. *Activity Diagram*

(Sumber : Aris, et al., 2015)

3. Class Diagram

Diagram ini digunakan untuk menggambarkan perbedaan yang mendasar antara *class-class*, hubungan antar *class*, dan di mana sub-sistem *class* tersebut. Pada *class diagram* terdapat nama *class*, *attributes*, *operations*, serta *association* (hubungan antar *class*) (Indrajani, 2015 : 49). *Class* diagram menunjukan sekumpulan kelas, antarmuka, dan kerjasama serta hubungannya. *Class* diagram digunakan untuk memodelkan perancangan statik dari gambaran sistem. Biasanya meliputi pemodelan *vocabulary* dari sistem, pemodelan kerjasama, atau pemodelan skema. *Class* diagram dapat digunakan untuk membangun sistem yang dapat dieksekusi melalui teknik *forward and reverse*, selain untuk penggambaran, menspesifikasikan, dan pendokumentasian struktur model (Achmad Hamzah Nasrullah dan Dadang Sudrajat, 2015). Adapun contoh pengguna *class diagram* dapat dilihat pada Gambar II.8.



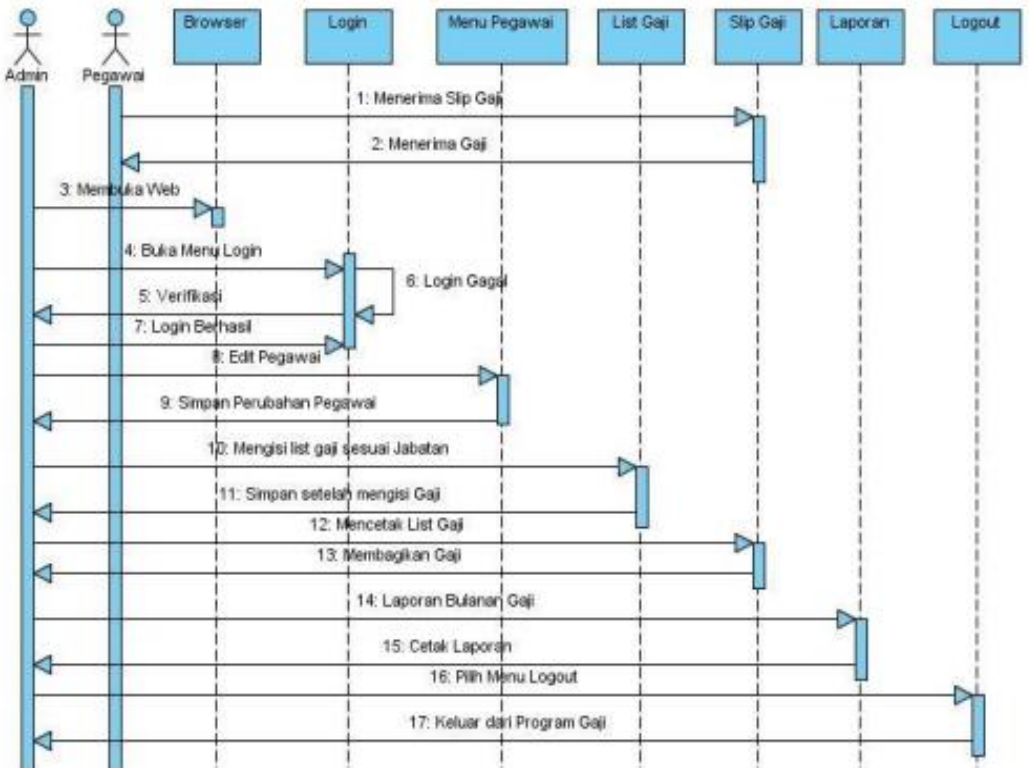
Gambar II.8. Class Diagram

(Sumber : Indrajani, 2015 : 50)

4. Sequence Diagram

Diagram ini merupakan suatu diagram interaksi yang menggambarkan bagaimana objek-objek berpartisipasi dalam bagian interaksi (*particular interaction*) dan pesan yang ditukar dalam urutan waktu. *Sequence* diagram dapat digambarkan dalam beberapa level secara detail dan untuk tujuan yang berbeda pada beberapa langkah yang dikembangkan secara *lifecycle*

(Indrajani, 2015: 50). *Sequence* Diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan jumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini di dalam *use case* (Oktafiansyah, 2012). Adapun contoh pengguna *sequence diagram* dapat dilihat pada Gambar II.9.



Gambar II.9. Sequence Diagram

(Sumber : Aris, et al., 2015)

II.9. *Microsoft Visual Studio* 2010

Microsoft Visual Studio 2010 merupakan sebuah perangkat lunak yang dapat digunakan untuk melakukan pengembangan aplikasi, yang di dalamnya terdapat

beberapa kumpulan *tools* pemrograman seperti *Visual Basic .NET, Visual C++ .NET, Visual C# .NET,* dan *Visual J# .NET*. Namun yang paling umum digunakan yaitu : *Visual Basic .NET*. *Visual Basic .NET* adalah *Visual Basic* yang direkayasa kembali untuk digunakan pada *platform .NET* sehingga aplikasi yang dibuat menggunakan *Visual Basic .NET* dapat berjalan pada sistem komputer apapun, dan dapat mengambil data dari *server* dengan tipe apapun asalkan terinstal *.NET Framework* (Priyanto Hidayatullah, 2012 : 5-8).

Microsoft Visual Basic.NET (VB.NET) adalah suatu pengembangan aplikasi bahasa pemrograman berbasis Visual Basic dan merupakan bahasa pemrograman terbaru buatan Microsoft setelah Microsoft Visual Basic 6.0. Pengembangan yang signifikan dari VB.NET ialah kemampuannya memanfaatkan platform NET, sehingga pengguna dapat membuat aplikasi Windows, aplikasi konsol, pustaka kelas, layanan NT, aplikasi web form, dan XML Web Service, yang secara keseluruhan memungkinkan integrasi tanpa batas dengan bahasa pemrograman lain sehingga berpeluang untuk berintegrasi dengan web.

Beberapa keunggulan lainnya yang dimiliki VB.NET, seperti memiliki penanganan debug yang baik sehingga pembangun aplikasi dapat mengetahui kesalahan kode yang terjadi secara cepat dan memiliki Windows form design yang memungkinkan pembangun/developer memperoleh aplikasi desktop dalam waktu singkat. VB.NET memiliki Interface Development Environment (IDE) yang lebih lengkap dan mudah bagi user pemula untuk mencari komponen atau objek yang kita inginkan, seperti menempelkan kontrol-kontrol yang terdapat pada toolbox, mampu

memformat secara otomatis ukuran textbox, serta mengatur property dari masing-masing kontrol. VB.NET juga memiliki .NET Framework. Microsoft .NET ialah sebuah platform untuk membangun, menjalankan, dan meningkatkan generasi lanjut dari aplikasi terdistribusi, memperluas klien, server dan service (Widiana Mulyani, 2015 : 16).

II.10. *SQL Server* 2008

Microsoft SQL Server merupakan produk *Relational Database Management System* (RDBMS) yang dibuat oleh *Microsoft*. Pada tahun 2008 *Microsoft* mengeluarkan *SQL Server 2008 R2* yang merupakan versi yang banyak digunakan. *SQL* (*Structured Query Language*) adalah sebuah bahasa yang dipergunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara *de facto* merupakan bahasa standar yang digunakan dalam manajemen basis data relasional. Saat ini hampir semua *server* basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya. *SQL* terdiri dari dua bahasa, yaitu *Data Definition Language* (DDL) dan *Data Manipulation Language* (DML). Implementasi DDL dan DML berbeda untuk tiap Sistem Manajemen Basis Data (SMBD), namun secara umum implementasi setiap bahasa ini memiliki bentuk standar yang ditetapkan oleh ANSI (Adelia dan Jimmy Setiawan, 2011 : 115).

SQL Server adalah sebuah *database* relasional yang dirancang untuk mendukung aplikasi dengan arsitektur *client/server* dimana *database* terdapat pada komputer pusat yang disebut *server*, dan informasi digunakan bersama–sama oleh

beberapa *user* yang menjalankan aplikasi didalam komputer lokalnya yang disebut dengan *client.* Arsitektur semacam ini memberikan integritas data yang tinggi karena semua *user* bekerja dengan informasi yang sama. Melalui aturan-aturan bisnis, kendali diterapkan kepada semua *user* mengenai informasi yang ditambahkan ke dalam *database*. *Database SQL Server* dibagi kedalam beberapa komponen logikal, seperti misalnya tabel, view, dan elemen–elemen lain yang terlihat oleh *user* (Rika Yunitarini, 2013).