BAB II

TINJAUAN PUSTAKA

2.1 Plagiarisme

2.1.1 Pengertian Plagiarisme

Plagiarisme adalah tindakan menjiplak karya orang lain dan dianggap sebagai bentuk kriminalitas karena mencuri hak cipta orang lain. Plagiarisme atau sering disebut plagiat adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri [1].

Plagiarisme adalah tindakan penyalahgunaan, pencurian / perampasan, penerbitan, pernyataan, atau menyatakan sebagai milik sendiri sebuah pikiran, ide, tulisan, atau ciptaan yang sebenarnya milik orang lain [2]. Di dunia pendidikan plagiarisme akan mendapatkan sanksi tegas seperti penguguran karya ilmiah bahkan dikeluarkan dari institusi.

Dalam buku *Bahasa Indonesia: Sebuah Pengantar Penulisan Ilmiah*, Felicia Utorodewo dkk. menggolongkan hal-hal berikut sebagai tindakan plagiarism [3] :

1. Mengakui tulisan orang lain sebagai tulisan sendiri,
2. Mengakui gagasan orang lain sebagai pemikiran sendiri
3. Mengakui temuan orang lain sebagai kepunyaan sendiri
4. Mengakui karya kelompok sebagai kepunyaan atau hasil sendiri

5. Menyajikan tulisan yang sama dalam kesempatan yang berbeda tanpa menyebutkan asal-usulnya
6. Meringkas dan memparafrasekan (mengutip tak langsung) tanpa menyebutkan sumbernya
7. Meringkas dan memparafrasekan dengan menyebut sumbernya, tetapi rangkaian kalimat dan pilihan katanya masih terlalu sama dengan sumbernya.

Yang digolongkan sebagai plagiarisme:

1. Menggunakan tulisan orang lain secara mentah, tanpa memberikan tanda jelas (misalnya dengan menggunakan tanda kutip atau blok alinea yang berbeda) bahwa teks tersebut diambil persis dari tulisan lain
2. Mengambil gagasan orang lain tanpa memberikan anotasi yang cukup tentang sumbernya

Yang tidak tergolong plagiarisme:

1. menggunakan informasi yang berupa fakta umum
2. menuliskan kembali (dengan mengubah kalimat atau parafrase) opini orang lain dengan memberikan sumber jelas
3. mengutip secukupnya tulisan orang lain dengan memberikan tanda batas jelas bagian kutipan dan menuliskan sumbernya

Beberapa tipe plagiarisme yaitu : [3]

1. Word-for-word plagiarism

   Menyalin setiap kata secara langsung tanpa diubah sedikitpun.

2. Plagiarism of authorship

   Mengakui hasil karya orang lain sebagai hasil karya sendiri dengan cara mencantumkan nama sendiri menggantikan nama pengarang yang sebenarnya.

3. Plagiarism of ideas

   Mengakui hasil pemikiran atau ide orang lain.

4. Plagiarism of sources

   Jika seorang penulis menggunakan kutipan dari penulis lainnya tanpa mencantumkan sumbernya.

2.1.2 Metode Pendeteksian Plagiarisme

Metode pendeteksi plagiarisme dibagi menjadi tiga bagian yaitu metode perbandingan teks lengkap, metode dokumen *fingerprinting*, dan metode kesamaan kata kunci. Metode pendeteksi plagiarism dapat dilihat pada gambar 2.1 [4] :



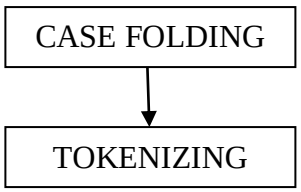
Gambar 2.1 Metode Pendeteksi Plagiarisme

Berikut ini penjelasan dari masing-masing metode dan algoritma pendeteksi plagiarisme :

1. Perbandingan teks lengkap. Metode ini diterapkan dengan membandingkan semua isi dokumen. Dapat diterapkan untuk dokumen yang besar. Pendekatan ini membutuhkan waktu yang lama tetapi cukup efektif, karena kumpulan dokumen yang diperbandingkan adalah dokumen yang disimpan pada penyimpanan lokal. Metode perbandingan teks lengkap tidak dapat diterapkan untuk kumpulan dokumen yang tidak terdapat pada dokumen lokal. Algoritma yang digunakan pada metode ini adalah algoritma *Brute-Force*, algoritma *edit distance,* algoritma *Boyer Moore* dan algoritma *lavenshtein distance.*
2. Dokumen *Fingerprinting*. Dokumen *fingerprinting* merupakan metode yang digunakan untuk mendeteksi keakuratan salinan antar dokumen, baik semua teks yang terdapat di dalam dokumen atau hanya sebagian teks saja. Prinsip kerja dari metode dokumen *fingerprinting* ini adalah dengan menggunakan teknik *hashing*. Teknik *hashing* adalah sebuah fungsi yang mengkonversi setiap string menjadi bilangan. Misalnya Rabin-Karp, Winnowing dan Manber.
3. Kesamaan Kata Kunci. Prinsip dari metode ini adalah mengekstrak kata kunci dari dokumen dan kemudian dibandingkan dengan kata kunci pada dokumen yang lain. Pendekatan yang digunakan pada metode ini adalah teknik dot.

2.2 Ekstraksi Dokumen

Teks yang akan dilakukan proses teks *mining*, pada umumnya memiliki beberapa karakteristik diantaranya adalah memiliki dimensi yang tinggi, terdapat *noise* pada data, dan terdapat struktur teks yang tidak baik. Cara yang digunakan dalam mempelajari suatu data teks, adalah dengan terlebih dahulu menentukan fitur-fitur yang mewakili setiap kata untuk setiap fitur yang ada pada dokumen. Sebelum menentukan fitur-fitur yang mewakili, diperlukan tahap *preprocessing* yang dilakukan secara umum dalam teks *mining* pada dokumen, yaitu *case folding* dan *tokenizing*. Gambar 2.2 adalah tahap dari *preprocessing* [5] *:*



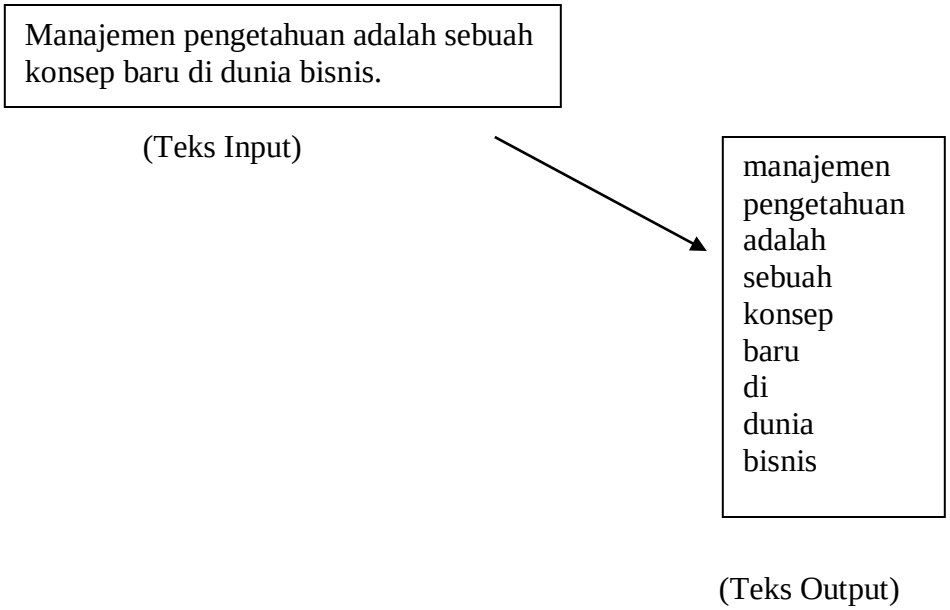
Gambar 2.2 Tahap *Preprocessing*

2.2.1 Case Folding

Case folding adalah mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf ‘a‘ sampai dengan ‘z‘ yang diterima.

2.2.2 Tokenizing

Tahap *tokenizing / parsing* adalah tahap pemotongan string input berdasarkan tiap kata yang menyusunnya. Karakter selain huruf dihilangkan dan dianggap *delimiter*. Gambar 2.3 adalah proses *tokenizing* [5] :



Gambar 2.3 *Tokenizing*

2.3 Algoritma Rabin-Karp

Algoritma Karp-Rabin diciptakan oleh Michael O. Rabin dan Richard M. Karp pada tahun 1987 yang menggunakan fungsi *hashing* untuk menemukan *pattern* di dalam string teks.

Fungsi *hashing* menyediakan metode sederhana untuk menghindari perbandingan jumlah karakter yang quadratik di dalam banyak kasus atau situasi. Daripada melakukan pemeriksaan terhadap setiap posisi dari teks ketika terjadi pencocokan pola, akan lebih baik efisien untuk melakukan pemeriksaan hanya jika teks yang sedang proses memiliki kemiripan seperti pada *pattern*. Untuk melakukan pengecekan kemiripan antara dua kata ini digunakan fungsi *hash* [6].

Algoritma Rabin-Karp ini banyak digunakan dalam pendeteksian pencontekan atau kecurangan.Contohnya pada makalah atau pada *paper*. [6]

Kelebihan algoritma rabin-karp diantaranya sebagai berikut:

1. Rabin-Karp menelusuri karakter satu persatu pada deret karakter (kontigu), tetapi proses perbandingannya (penghitungan *hash key* nya) relatif mudah.
2. Kasus pencarian string dengan pola yang panjang.

Kekurangan algoritma rabin-karp diantaranya sebagai berikut:

1. Membutuhkan waktu yang lama dalam membandingkan kata.
2. Tidak bisa menentukan persamaan makna *sinonim* kata.

2.3.1 Hashing

Hashing adalah suatu cara untuk mentransformasi sebuah string menjadi suatu nilai yang unik dengan panjang tertentu (*fixed-length*) yang berfungsi sebagai penanda string tersebut. Fungsi untuk menghasilkan nilai ini disebut fungsi *hash*, sedangkan nilai yang dihasilkan disebut nilai *hash*. Contoh sederhana *hashing* adalah:

Firdaus, Hari

Munir, Rinaldi

Rabin, Michael

Karp, Richard

menjadi :

7864 = Firdaus, Hari

9802 = Munir, Rinaldi

1990 = Rabin, Michael

8822 = Karp, Richard

Contoh di atas adalah pengunaan *hashing* dalam pencarian pada *database*. Apabila tidak di-*hash*, pencarian akan dilakukan karakter per karakter pada nama-nama yang panjangnya bervariasi dan ada 26 kemungkinan pada setiap karakter. Namun pencarian akan menjadi lebih efisien setelah di-*hash* karena hanya akan membandingkan empat digit angka dengan cuma 10 kemungkinan setiap angka. Nilai *hash* pada umumnya digambarkan sebagai *fingerprint* yaitu suatu string pendek yang terdiri atas huruf dan angka yang terlihat acak (data biner yang ditulis dalam heksadesimal) [8].

Rolling hashing adalah suatu cara menyimpan dan mengambil target elemen tanpa *searching*, yaitu dengan cara menghitung lokasi target. Fungsi *hash* dengan basis disebut dengan *Rolling Hash.* Basis biasanya adalah bilangan prima yang cukup besar. Persamaannya adalah sebagai berikut :

$$H = C_1 * a^{(k-1)} + C_2 * a^{(k-2)} + C_3 * a^{(k-3)} \ldots + C_k * a^0 \quad \ldots.(II.1)$$

Keterangan :

H adalah nilai *Hash*

C adalah nilai ASCII suatu karakter

a adalah basis (tidak boleh 1 dan 0)

k adalah banyaknya karakter

Algoritma Rabin-Karp didasarkan pada fakta jika dua buah string sama maka harga *hash value*-nya pasti sama. Akan tetapi ada dua masalah yang timbul dari hal ini, masalah pertama yaitu ada begitu banyak string yang berbeda, permasalahan ini dapat dipecahkan dengan meng-*assign* beberapa string dengan *hash value* yang sama. Masalah yang kedua belum tentu string yang mempunyai *hash value* yang sama cocok untuk mengatasinya maka untuk setiap string yang di-*assign* dilakukan pencocokan string secara *Brute-Force*. Kunci agar algoritma Rabin-Karp efisien, terdapat pada pemilihan *hash value*-nya. Salah satu cara yang terkenal dan efektif adalah memperlakukan setiap *substring* sebagai suatu bilangan dengan basis tertentu [8].

2.3.2 *K*-grams

Kgrams adalah rangkaian *terms* dengan panjang K. Kebanyakan yang digunakan sebagai *terms* adalah kata. ***K-gram*** merupakan sebuah metode yang diaplikasikan untuk pembangkitan kata atau karakter. Metode *k-grams* ini digunakan untuk mengambil potongan-potongan karakter huruf sejumlah k dari sebuah kata yang secara kontinuitas dibaca dari teks sumber hingga akhir dari dokumen. Berikut ini adalah contoh *k-grams* dengan k=5 :

- *Text*: A do run run run, a do run run
- Kemudian dilakukan penghilangan spasi : adorunrunrunadorunrun
- Sehingga dihasilkan rangkaian 5-*grams* yang diturunkan dari *text*:

- adoru dorun orunr runru unrun nrunr runru unrun nruna runad unado nador adoru dorun orunr runru unrun [9].

2.3.3 Konsep Algoritma Rabin-Karp

Algoritma Rabin-Karp adalah algoritma pencocokan *string* yang menggunakan fungsi *hash* sebagai pembanding antara *string* yang dicari (m) dengan *substring* pada teks (n). Apabila *hash value* keduanya sama maka akan dilakukan perbandingan sekali lagi terhadap karakter-karakternya. Apabila hasil keduanya tidak sama, maka substring akan bergeser ke kanan. Pergeseran dilakukan sebanyak (n-m) kali. Perhitungan nilai hash yang efisien pada saat pergeseran akan mempengaruhi performa dari algoritma ini. Cara kerja dari algoritma Rabin-Karp dapat dilihat pada gambar 2.4 [8].

```
function RabinKarp (input s:
string[1..m], teks: string[1..n])
boolean
{ Melakukan pencarian string s
pada
string teks dengan algoritma
Rabin-K}
Deklarasi
i : integer
ketemu = boolean

Algoritma:
ketemu ←false
hs ←hash(s[1..m])
for i ← 0 to n-m do
      hsub ←hash(teks[1..i+m-1])
      if hsub = hs then
          if teks[i..i+m-1] = s
then
          ketemu ←true
      else
      hsub ←hash(teks[i+1..i+m])
endfor
return ketemu
```

Gambar 2.4 Algoritma Rabin-Karp

2.3.4 Pengukuran Nilai Similaritas

Inti dari pendekatan *k-grams* dibagi menjadi dua tahap. Tahap pertama, membagi kata menjadi *k-grams*. Kedua, mengelompokkan hasil *terms* dari *k-grams* yang sama. Kemudian untuk menghitung *similarity* dari kumpulan kata tersebut maka digunakan rumus pengukuran nilai similaritas untuk pasangan kata yang digunakan.

$$S = \frac{N_t}{N_x + N_y} \qquad \text{....(II.2)}$$

Keterangan:

S = Similaritas

N_t = Jumlah Hashing yang sama

N_x = Total substring asli

N_y = Total substring uji

2.3.5 Persentase Nilai Similaritas

Untuk menentukan jenis dokumen yang diuji ada 5 jenis penilaian persentase similaritas:

1. 0% : Hasil uji 0% berarti kedua dokumen tersebut benar-benar berbeda baik dari segi isi dan kalimat secara keseluruhan
2. < 15%: Hasil uji 15% berarti kedua dokumen tersebut hanya mempunyai sedikit kesamaan
3. 15-50%: Hasil uji 15-50% berarti menandakan dokumen tersebut termasuk plagiat tingkat sedang
4. >50%: Hasil uji lebih dari 50% berarti dapat dikatakan bahwa dokumen tersebut mendekati plagiarism
5. 100%: Hasil uji 100% menandakan bahwa dokumen tersebut adalah plagiat karena dari awal sampai akhir mempunyai isi yg sama persis [11].

2.4 Basis Data

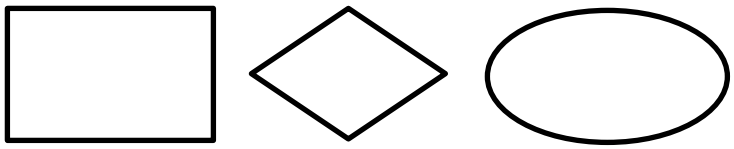
Data merupakan fakta mengenai suatu objek seperti manusia, benda, peristiwa, konsep, keadaan dan sebagainya yang dapat dicatat dan mempunyai arti secara implisit. Data dapat dinyatakan dalam bentuk angka, karakter atau simbol, sehingga bila data dikumpulkan dan saling berhubungan maka dikenal dengan istilah basis data (database) [16]. Sedangkan menurut George Tsu-der Chou basis data merupakan kumpulan informasi bermanfaat yang diorganisasikan ke dalam aturan yang khusus. Informasi ini adalah data yang telah diorganisasikan ke dalam bentuk yang sesuai dengan kebutuhan seseorang [17]. Menurut *Encyclopedia of Computer Science and Engineer*, para ilmuwan di bidang informasi menerima definisi standar informasi yaitu data yang digunakan dalam pengambilan keputusan.

2.4.1 ERD (*ENTITY RELATIONSHIP MODEL)*

Model entity-relationship pertama kali diperkenalkan oleh Peter Chen pada tahun 1976. Dalam pemodelan ini dilakukan dengan tahapan sebagai berikut [18]:

1. Memilih entitas-entitas yang akan disusun dalam basis data dan menentukan hubungan antar entitas yang telah dipilih.
2. 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:



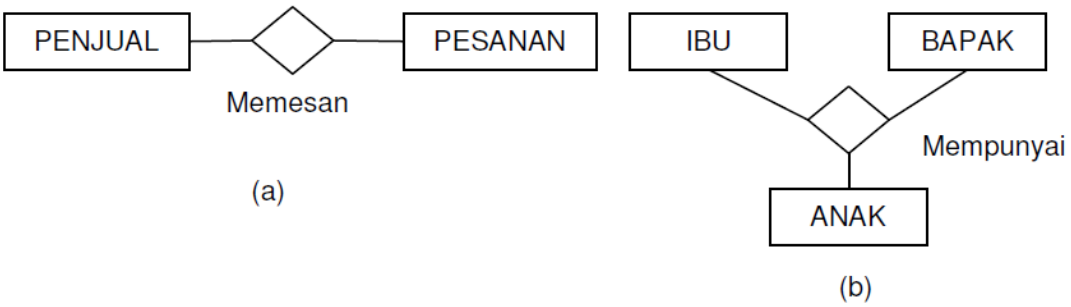
Gambar 2.5 Elemen-elemen ER Model

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 [18]:

1. Kelas entitas adalah kumpulan entitas dan dijelaskan oleh struktur atau format entitas di dalam kelas.
2. Instansi kelas merupakan bentuk penyajian dari fakta entitas.

Umumnya terdapat banyak instansi entitas di dalam setiap entitas kelas. Setiap entitas kelas memiliki atribut yang menjelaskan karakteristik dari entitas tersebut, sedangkan setiap instansi entitas mempunyai identifikasi yang dapat bernilai unik (mempunyai nilai yang berbeda untuk setiap identifikasinya) atau non-unik (dapat bernilai sama untuk setiap identifikasinya).

Antara entitas diasosiakan dalam suatu hubungan (*relationship*). Suatu relationship dapat memiliki beberapa atribut. Jumlah kelas entitas dalam suatu relationship disebut **derajat relationship**. Gambar di bawah ini merupakan contoh dari relationship berderajat dua dan relationship berderajat tiga.

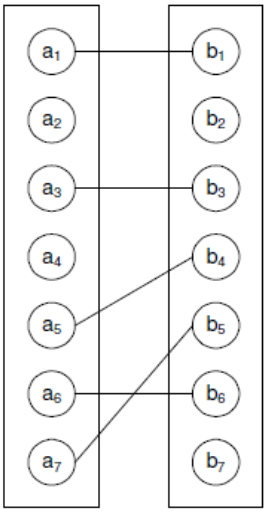


Gambar 2.6 (a) Relationship berderajat dua (b) Relationship berderajat tiga

2.4.2 Tipe Binary Relationship

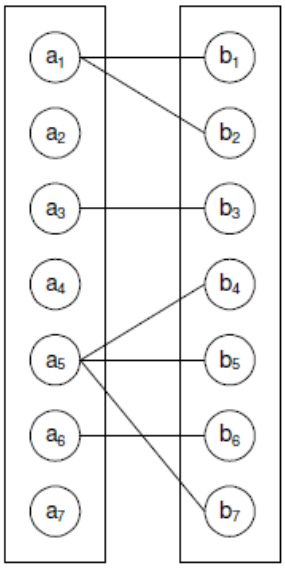
Relationship memiliki tiga tipe biner yaitu:

1. One-to-one (1:1). Hubungan terjadi bila setiap instansi entitas hanya memiliki satu hubungan dengan instansi entitas lain.



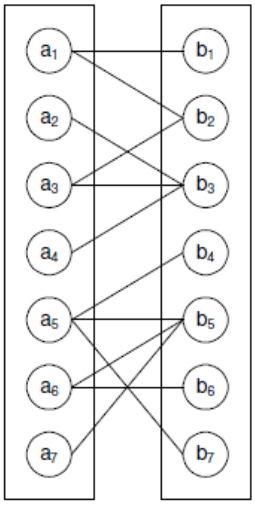
Gambar 2.7 Hubungan 1:1 (one-to-one)

2. One-to-many (1:M). Relationship ini terjadi bila setiap instansi entitas dapat memiliki lebih dari satu hubungan terhadap instansi entitas lain tetapi tidak kebalikannya.



Gambar 2.8 Hubungan 1:M (one-to-many)

3. Many-to-many (M:N). Hubungan saling memiliki lebih dari satu dari setiap instansi entitas terhadap instansi entitas lainnya.



Gambar 2.9 Hubungan M:N (many-to-many)

2.4.3 Normalisasi

Proses normalisasi menyediakan cara sistematis untuk meminimalkan terjadinya kerangkapan data diantara relationship dalam perancangan logikal basis data.

2.4.3.1 Bentuk Normal Pertama (1NF)

Bentuk normal pertama (1NF) mensyaratkan bahwa data dalam tabel menjadi dua dimensi, bahwa tidak ada kelompok yang berulang pada baris. Contoh tabel tidak dalam 1NF adalah di mana ada karyawan "record" seperti [19]:

```
Employee(name, address, {dependent name})
```

di mana {nama tergantung} menyimpulkan bahwa atribut itu diulangi. contoh data untuk catatan ini mungkin:

Smith, 123 4th St., {John, Mary, Paul, Sally}

Jones, 4 Moose Lane., {Edgar, Frank, Bob}

Adams, 88 Tiger Circle., {Kaitlyn, Alicia, Allison}

Masalah dengan menempatkan data dalam tabel dengan kelompok mengulangi adalah bahwa tabel tidak dapat dengan mudah diindeks atau diatur sedemikian rupa sehingga informasi dalam mengulangi kelompok dapat ditemukan tanpa mencari setiap record secara individual. Orang relasional biasanya memanggil sekelompok mengulang "nonatomic" (memiliki lebih dari satu nilai dan dapat dipisah-pisahkan).

2.4.3.2 Bentuk Normal Kedua (2NF)

Bentuk normal kedua (2NF) mensyaratkan bahwa data dalam tabel tergantung pada seluruh kunci dari meja. Dependensi parsial tidak diperbolehkan. Contoh:

Employee (name, job, salary, address)

di mana ia membawa nama + pekerjaan kombinasi (kunci concatenated) untuk mengidentifikasi gaji, tapi alamat hanya bergantung pada nama. Beberapa data sampel:

Name	Job	Salary	Address
Smith	Welder	14.75	123 4th St
Smith	Programmer	24.50	123 4th St
Smith	Waiter	7.50	123 4th St
Jones	Programmer	26.50	4 Moose Lane
Jones	Bricklayer	34.50	4 Moose Lane
Adams	Analyst	28.50	88 Tiger Circle

Tabel 2.1 Bentuk Normal Kedua (2NF)

Dapatkah Anda melihat masalah berkembang di sini? Alamat akan diulang untuk setiap terjadinya nama. Ini mengulangi disebut redundansi dan menyebabkan anomali. Suatu anomali berarti bahwa ada pembatasan untuk melakukan karena susunan data sesuatu. Ada penyisipan anomali, anomali penghapusan, dan anomali update. Kunci dari tabel ini adalah Nama + Job - ini jelas karena tidak satu pun adalah unik dan benar-benar mengambil keduanya nama dan pekerjaan untuk mengidentifikasi gaji. Namun, alamat tergantung hanya pada nama, bukan pekerjaan, ini adalah contoh dari ketergantungan parsial. Alamat tergantung pada hanya sebagian dari kunci. Contoh penyisipan sebuah anomali akan di mana orang akan ingin memasukkan seseorang ke dalam tabel di atas, tetapi orang yang akan dimasukkan belum ditentukan pekerjaan. Ini tidak bisa dilakukan karena nilai harus

dikenal untuk atribut kerja. Nilai null tidak dapat nilai yang valid untuk kunci dalam database relasional (ini dikenal sebagai kendala entitas-integritas). Sebuah anomali pembaruan akan di mana salah satu karyawan berubah nya atau alamatnya. Tiga baris harus menjadi diubah untuk mengakomodasi perubahan ini salah alamat. Contoh menghapus anomali adalah bahwa Adams berhenti, sehingga Adams hilang, tapi kemudian informasi bahwa analis yang dibayar $ 28,50 juga hilang. Oleh karena itu, lebih informasi terkait dari yang diantisipasi sebelumnya hilang.

2.4.3.3 Bentuk Normal Ketiga (3NF)

Bentuk normal ketiga (3NF) mensyaratkan bahwa data dalam tabel tergantung pada primary key dari tabel. Sebuah contoh klasik non-3NF adalah:

Employee (name, address, project#, project-location)

Misalkan project-location berarti lokasi dari mana proyek dikendalikan, dan didefinisikan oleh project#. Beberapa data sampel akan menunjukkan masalah dengan tabel ini:

Name	Address	Project#	Project-location
Smith	123 4th St	101	Memphis
Smith	123 4th St	102	Mobile
Jones	4 Moose Lane	101	Memphis

Tabel 2.2 Bentuk Normal Ketiga (3NF)

Perhatikan redundansi dalam tabel ini. Proyek 101 terletak di Memphis, tetapi setiap kali seseorang tercatat bekerja pada proyek 101, fakta bahwa mereka bekerja pada sebuah proyek yang dikendalikan dari Memphis dicatat lagi. itu sama anomali - anomali insert, anomali update, dan menghapus anomali - juga hadir dalam tabel ini.

Untuk menghapus database anomali dan redudansi, database harus normalisasi. Proses normalisasi melibatkan membelah meja menjadi dua atau lebih tabel (dekomposisi). Setelah tabel terpecah (proses yang disebut dekomposisi), mereka dapat bersatu kembali dengan sebuah operasi yang disebut "bergabung."

2.5 XAMPP

XAMPP adalah perangkat lunak bebas, yang mendukung banyak sistem operasi, merupakan kompilasi dari beberapa program. Fungsinya adalah sebagai server yang berdiri sendiri (localhost), yang terdiri atas program *Apache HTTP Server, MySQL database,* dan penerjemah bahasa yang ditulis dengan bahasa pemrograman *PHP* dan *Perl*. Nama XAMPP merupakan singkatan dari X (empat sistem operasi apapun), *Apache, MySQL, PHP* dan *Perl*. Program ini tersedia dalam *GNU General Public License* dan bebas, merupakan web server yang mudah digunakan yang dapat melayani tampilan halaman web yang dinamis. Untuk mendapatkanya dapat mendownload langsung dari web resminya [20].

2.6 Java

Java adalah bahasa pemrograman yang dapat dijalankan di berbagai komputer termasuk telepon genggam. Bahasa ini awalnya dibuat oleh James Gosling saat masih bergabung di Sun Microsystems saat ini merupakan bagian dari Oracle dan dirilis tahun 1995. Bahasa ini banyak mengadopsi sintaksis yang terdapat pada C dan C++ namun dengan sintaksis model objek yang lebih sederhana serta dukungan rutin-rutin aras bawah yang minimal. Aplikasi-aplikasi berbasis java umumnya dikompilasi ke dalam p-code (*bytecode*) dan dapat dijalankan pada berbagai Mesin Virtual Java (JVM). Java merupakan bahasa pemrograman yang bersifat umum/non-spesifik (*general purpose*), dan secara khusus didisain untuk memanfaatkan dependensi implementasi seminimal mungkin. Karena fungsionalitasnya yang memungkinkan aplikasi java mampu berjalan di beberapa platform sistem operasi yang berbeda, java dikenal pula dengan slogannya, "*Tulis sekali, jalankan di mana pun*".Saat ini java merupakan bahasa pemrograman yang paling *actor* digunakan, dan secara luas dimanfaatkan dalam pengembangan berbagai jenis perangkat lunak aplikasi desktop ataupun aplikasi berbasis web [12].

2.6.1 Sejarah Singkat Perkembangan Java

Bahasa pemrograman Java terlahir dari The Green Project, yang berjalan selama 18 bulan, dari awal tahun 1991 hingga musim panas 1992. Proyek tersebut belum menggunakan versi yang dinamakan Oak. Proyek ini dimotori oleh Patrick Naughton, Mike Sheridan, James Gosling dan Bill Joy, beserta *actor* pemrogram

lainnya dari Sun Microsystems. Salah satu hasil proyek ini adalah *actor Duke*y ang dibuat oleh Joe Palrang.

Pertemuan proyek berlangsung di sebuah gedung perkantoran *Sand Hill Road* di Menlo Park. Sekitar musim panas 1992 proyek ini ditutup dengan menghasilkan sebuah program *Java Oak* pertama, yang ditujukan sebagai pengendali sebuah peralatan dengan teknologi layar sentuh (*touch screen*), seperti pada PDA sekarang ini. Teknologi baru ini dinamai "*7" (*Star Seven*).

Setelah era *Star Seven* selesai, sebuah anak perusahaan Tv kabel tertarik ditambah beberapa orang dari proyek The Green Project. Mereka memusatkan kegiatannya pada sebuah ruangan kantor di 100 Hamilton Avenue, Palo Alto.

Perusahaan baru ini bertambah maju: jumlah karyawan meningkat dalam waktu singkat dari 13 menjadi 70 orang. Pada rentang waktu ini juga ditetapkan pemakaian Internet sebagai medium yang menjembatani kerja dan ide di antara mereka.Pada awal tahun 1990-an, Internet masih merupakan rintisan, yang dipakai hanya di kalangan akademisi dan militer.

Mereka menjadikan perambah (*browser*) Mosaic sebagai landasan awal untuk membuat perambah Java pertama yang dinamai Web Runner, terinsipirasi dari film 1980-an, Blade Runner. Pada perkembangan rilis pertama, Web Runner berganti nama menjadi Hot Java.

Pada sekitar bulan Maret1995, untuk pertama kali kode sumber Java versi 1.0a2 dibuka. Kesuksesan mereka diikuti dengan untuk pemberitaan pertama kali pada surat kabar *San Jose Mercury News* pada tanggal 23 Mei 1995.

Sayang terjadi perpecahan di antara mereka suatu hari pada pukul 04.00 di sebuah ruangan hotel Sheraton Palace. Tiga dari pimpinan utama proyek, Eric Schmidt dan George Paolini dari Sun Microsystems bersama Marc Andreessen, membentuk Netscape.

Nama Oak, diambil dari pohon oak yang tumbuh di depan jendela ruangan kerja "Bapak Java", James Gosling. Nama Oak ini tidak dipakai untuk versi release Java karena sebuah perangkat lunak lain sudah terdaftar dengan merek dagang tersebut, sehingga diambil nama penggantinya menjadi "Java". Nama ini diambil dari kopi murni yang digiling langsung dari biji (kopi tubruk) kesukaan Gosling.Konon kopi ini berasal dari Pulau Jawa. Jadi nama bahasa pemrograman Java tidak lain berasal dari kata Jawa (bahasa Inggris untuk Jawa adalah Java).

2.7 *Unified Modelling Language (*UML)

Unified Modelling Language (UML) adalah sebuah bahasa untuk menentukan, visualisasi, kontruksi, dan mendokumentasikan *artifacts* dari sistem perangkat lunak, untuk memodelkan bisnis, dan sistem *non-software* lainnya. UML merupakan suatu kumpulan teknik terbaik yang telah terbukti sukses dalam memodelkan sistem yang besar dan kompleks [13].

UML adalah keluarga notasi grafis yang didukung oleh meta-model tunggal, yang membantu pendeskripsian dan desain sistem perangkat lunak, khususnya sistem yang dibangun menggunakan pemrograman berorientasi objek [15]. UML merupakan standar yang terbuka yang dikontrol oleh *Object Management Group* (OMG), yang

terdiri dari banyak perusahaan. UML lahir dari penggabungan banyak bahasa pemodelan grafis berorientasi objek antara lain metode *Booch* oleh Graddy Booch, metode *Object Modelling Technique* (OMT) oleh DR. James Rumbaugh, dan metode *Object Oriented Software Engineering* (OOSE) oleh Ivar Jacobson. UML cocok digunakan untuk pemodelan perangkat lunak yang sudah berorientasi objek seperti Java, C++, PHP, C#, VB.NET dan AS3.

2.7.1 Komponen *Unified Modelling Language* (UML)

Komponen Diagram UML yang digunakan dalam membuat sistem pengukuran tingkat similaritas dokumen ini terdiri dari beberapa bagian, yaitu:

1. *Use Case Diagram*
2. *Activity Diagram*
3. *Sequence Diagram*
4. *Class Diagram*

UML dibagi menjadi 3 desain, yaitu:

1. *Desain Structure*

   Desain Structure dibagi menjadi :

   a. *Class Diagram*

   b. *Object Diagram*

   c. *Component Diagram*

   d. *Composite Diagram*

   e. *Package Diagram*

f. *Deploymen Diagram*

2. *Desain Behavior*

Desain Behavior dibagi menjadi :

a. *Use Case Diagram*

b. *Activity Diagram*

c. *State Machine Diagram*

3. *Desain Interaction*

Desain Interaction dibagi menjadi :

a. *Sequence Diagram*

b. *Communication Diagram*

c. *Timing Diagram*

d. *Interaction Overview Diagram*

2.7.2 *Desain Structure*

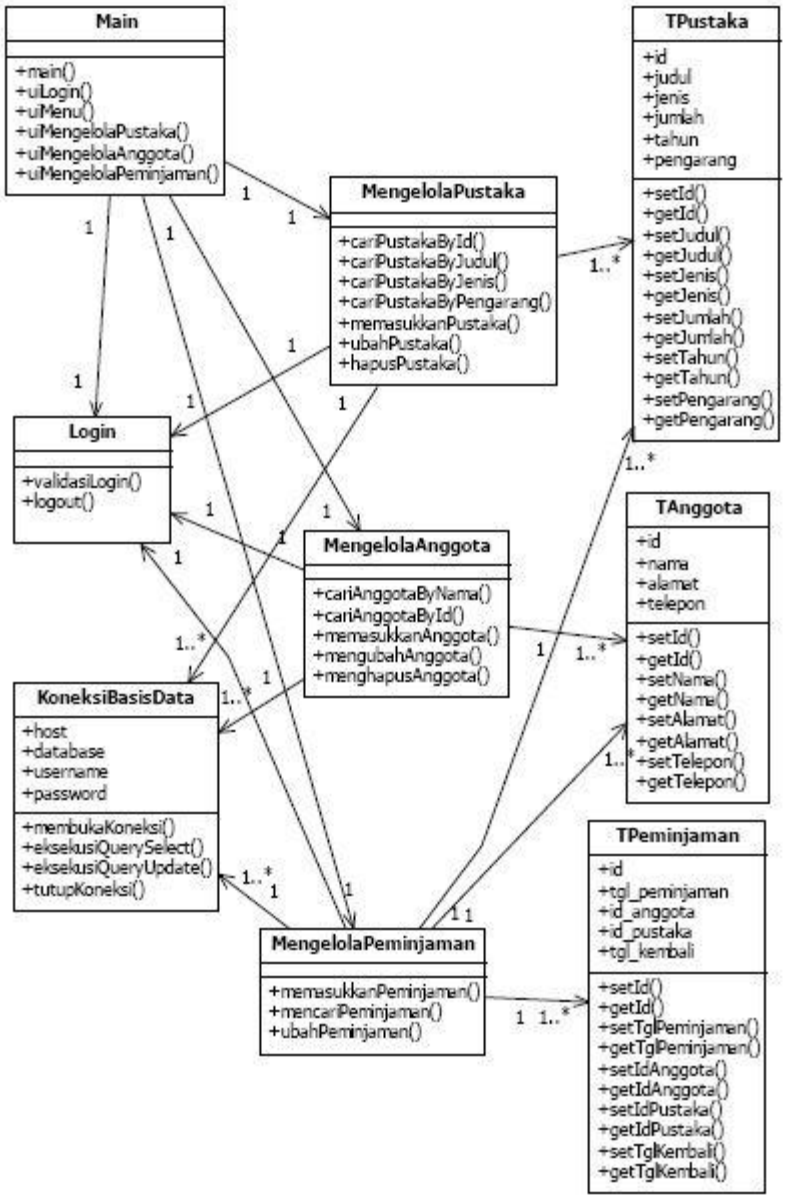
Desain Structure yaitu kumpulan diagram yang digunakan untuk menggambarkan suatu struktur statis dari sistem yang dimodelkan.

a. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi [14].

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

Berikut adalah contoh *class diagram*:

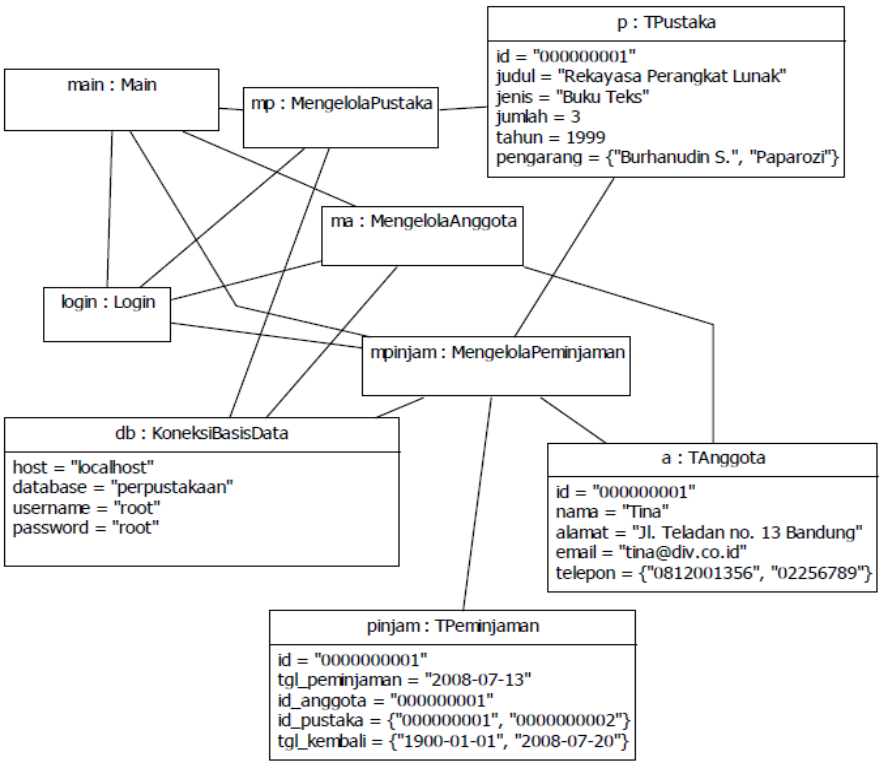


Gambar 2.10 *Class Diagram*

b. *Object Diagram*

Diagram objek menggambarkan struktur sistem dari segi penamaan objek dan jalannya objek dalam sistem. Pada diagram objek harus dipastikan semua kelas yang sudah didefinisikan pada diagram kelas harus dipakai objeknya, karena jika tidak, pendefinisian kelas itu tidak dapat dipertanggungjawabkan. Untuk apa mendefinisikan sebuah kelas sedangkan pada jalannya sistem, objeknya tidak pernah dipakai. Hubungan *link* pada diagram objek merupakan hubungan memakai dan dipakai dimana dua buah objek akan dihubungkan oleh *link* jika ada objek yang dipakai oleh objek lainnya [14].

Berikut adalah contoh *Object Diagram :*

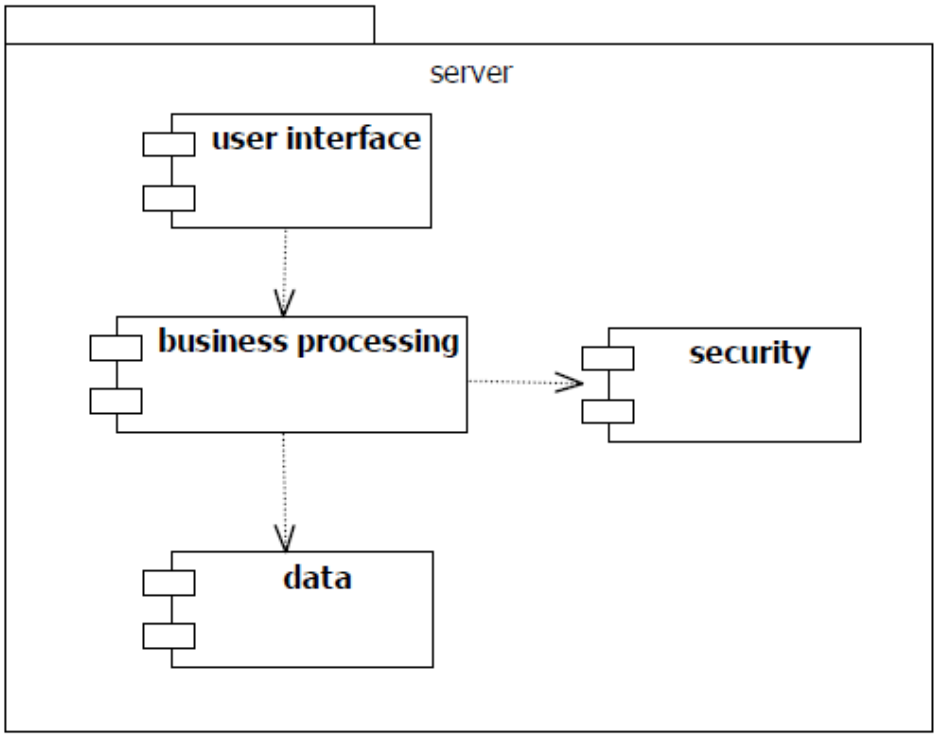


Gambar 2.11 *Object Diagram*

c. *Component Diagram*

Diagram komponen atau *component diagram* dibuat untuk menunjukkan organisasi dan ketergantungan diantara kumpulan komponen dalam sebuah sistem. Diagram komponen fokus pada komponen sistem yang dibutuhkan dan ada di dalam sistem [14].

Berikut adalah contoh *component diagram:*



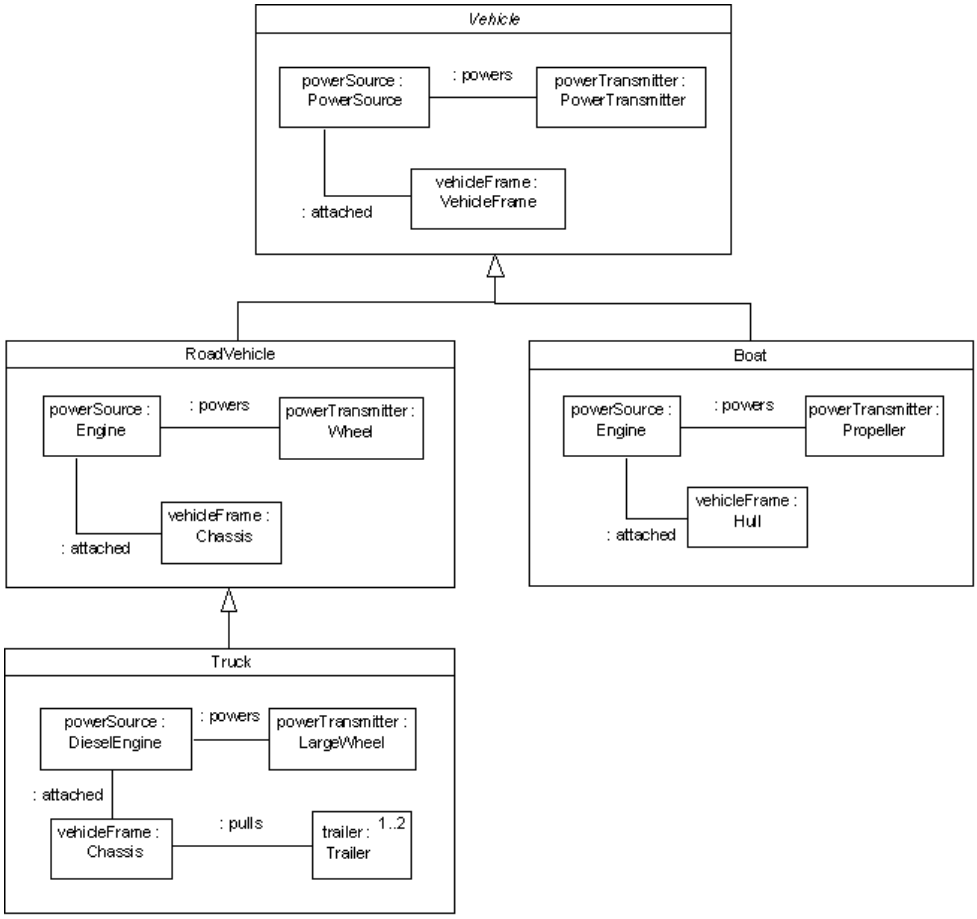
Gambar 2.12 *Component Diagram*

d. *Composite Structure Diagram*

Composite Structure Diagram adalah diagram untuk menunjukkan dekomposisi secara hierarkis sebuah *class* ke sebuah struktur internal. Hal

ini memungkinkan untuk memecah objek yang kompleks menjadi bagian-bagian yang kecil.

Berikut adalah contoh *Composite Structure* Diagram :

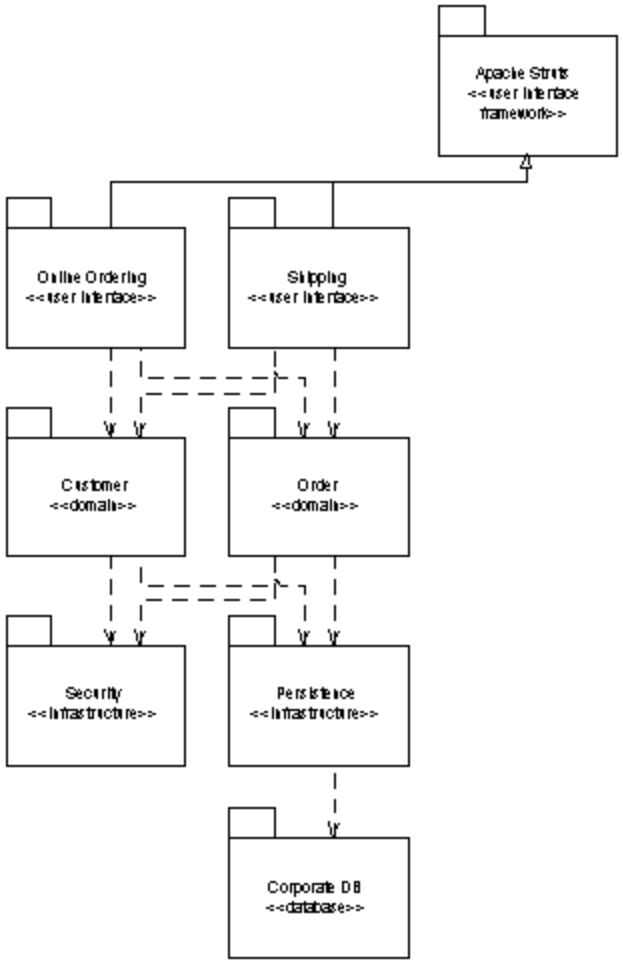


Gambar 2.13 *Composite Structure Diagram*

e. *Package Diagram*

Package diagram adalah sebuah bentuk pengelompokan yang memungkinkan untuk mengambil setiap bentuk di UML dan mengelompokkan elemen-elemen dalam tingkatan unit yang lebih tinggi.

Berikut adalah contoh *Package Diagram* :

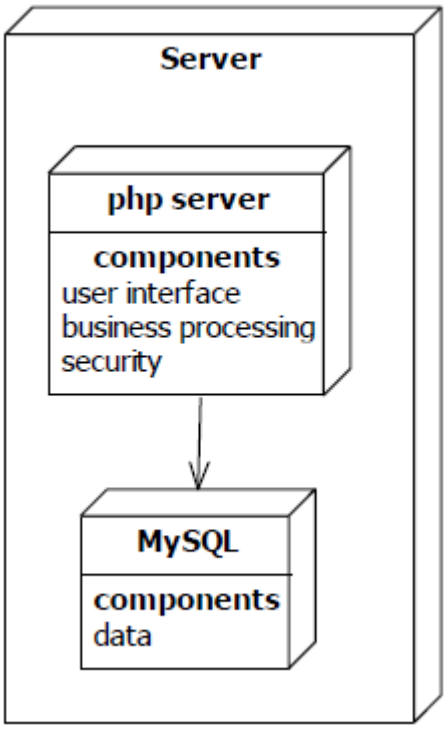


Gambar 2.14 *Package Diagram*

f. *Deployment Diagram*

Diagram *deployment* atau *deployment diagram* menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi.

Berikut adalah contoh *deployment diagram:*



Gambar 2.15 *Deployment Diagram*

2.7.3 *Desain Behavior*

Desain Behavior yaitu kumpulan diagram yang digunakan untuk menggambarkan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

a. *Use Case Diagram*

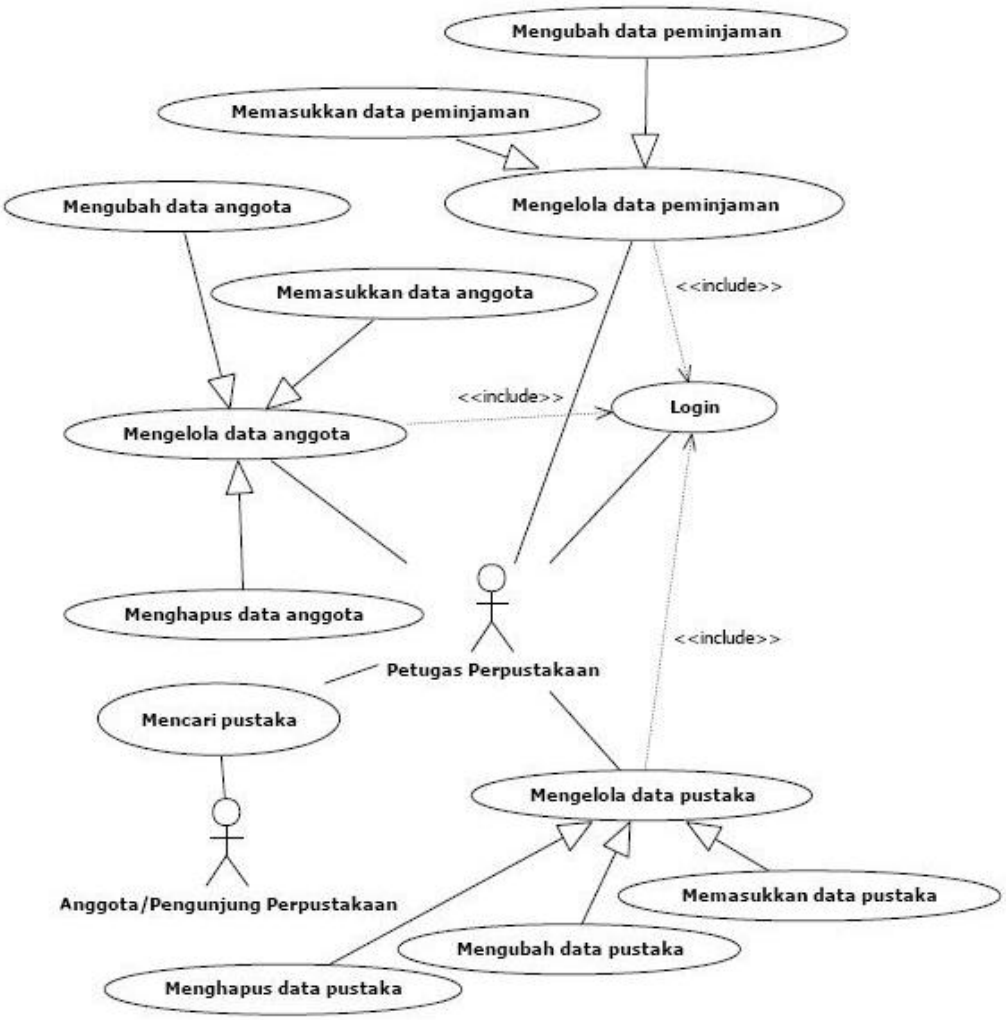
Use case atau diagram *use case* merupakan pemodelan untuk kelakukan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih actor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang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

dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu [14].

Syarat penamaan pada *use case* adalah nama didefinisikan sesimpel mungkin dan dapat dipahami. Ada dua hal utama pada *use case* yaitu pendefinisian apa yang disebut actor dan *use case*:

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

   Berikut adalah contoh *use case diagram:*



Gambar 2.16 *Use Case Diagram*

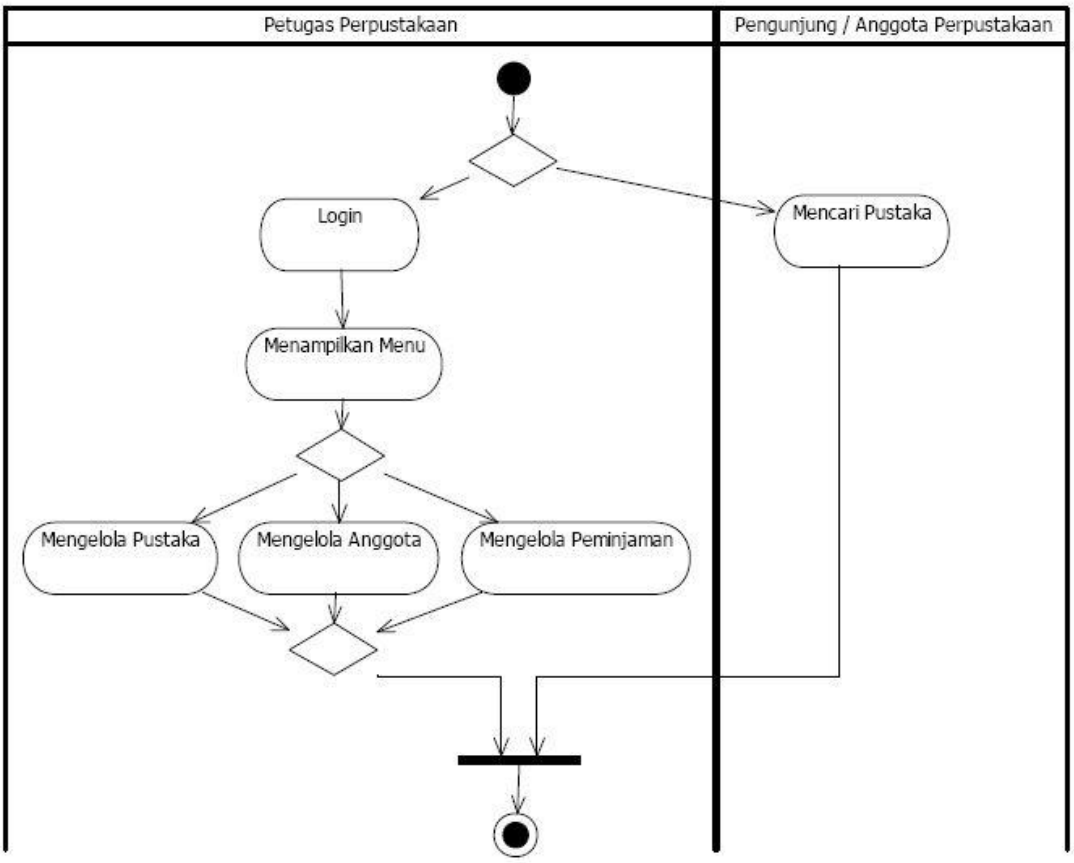
b. *Activity Diagram*

Diagram aktivitas atau *activity diagram* menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem [14].

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal berikut:

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

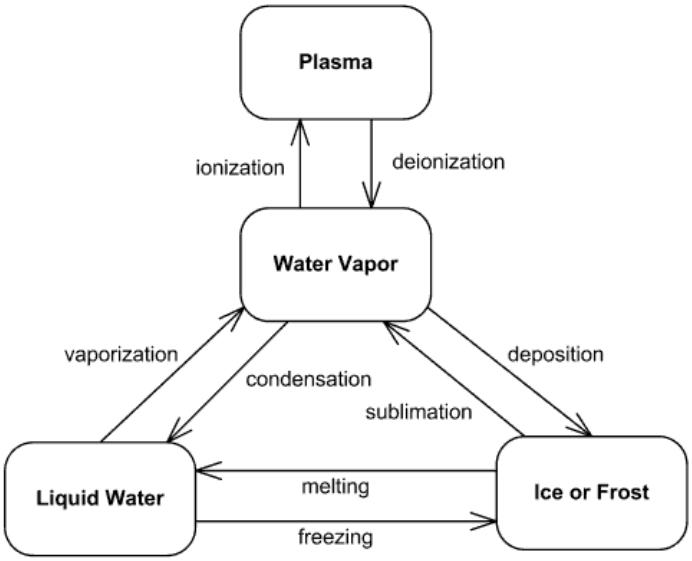
Berikut adalah contoh *Activity Diagram:*



Gambar 2.17 *Activity Diagram*

c. *State Machine Diagram*

State Machine Diagram menelusuri individu-individu objek melalui keseluruhan daur hidupnya, menspesifikasi semua urutan yang mungkin dari pesan-pesan yang akan diterima objek tersebut bersama-sama dengan tanggapan atas pesan-pesan tersebut. Diagram *State* menggambarkan transisi dan perubahan keadaan suatu objek dalam sistem sebagai akibat dari stimuli yang diterima. Pada umumnya diagram ini menggambarkan *class* tertentu. *State* diagram membantu analis, perancang dan pengembang untuk memahami perilaku objek dalam sistem. Berikut adalah contoh *State Machine Diagram :*



Gambar 2.18 *State Machine Diagram*

2.7.4 *Desain Interaction*

Desain Interaction yaitu kumpulan diagram yang digunakan untuk

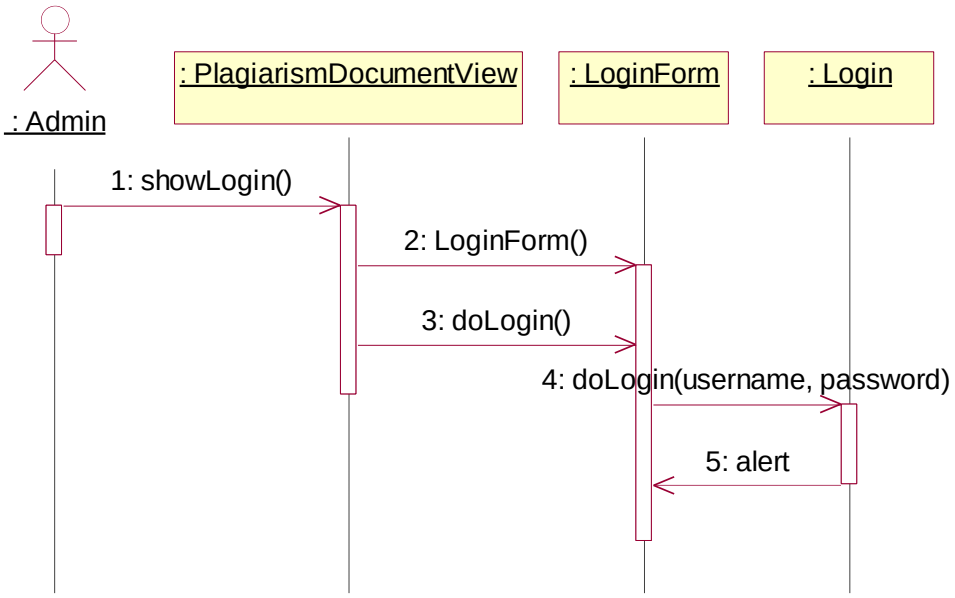
menggambarkan interaksi sistem dengan sistem lain maupun interaksi antar subsistem pada suatu sistem.

a. *Sequence Diagram*

Diagram sekuen menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuen yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefinisikan maka diagram sekuen yang harus dibuat juga semakin banyak [14].

Berikut adalah contoh *sequence diagram:*

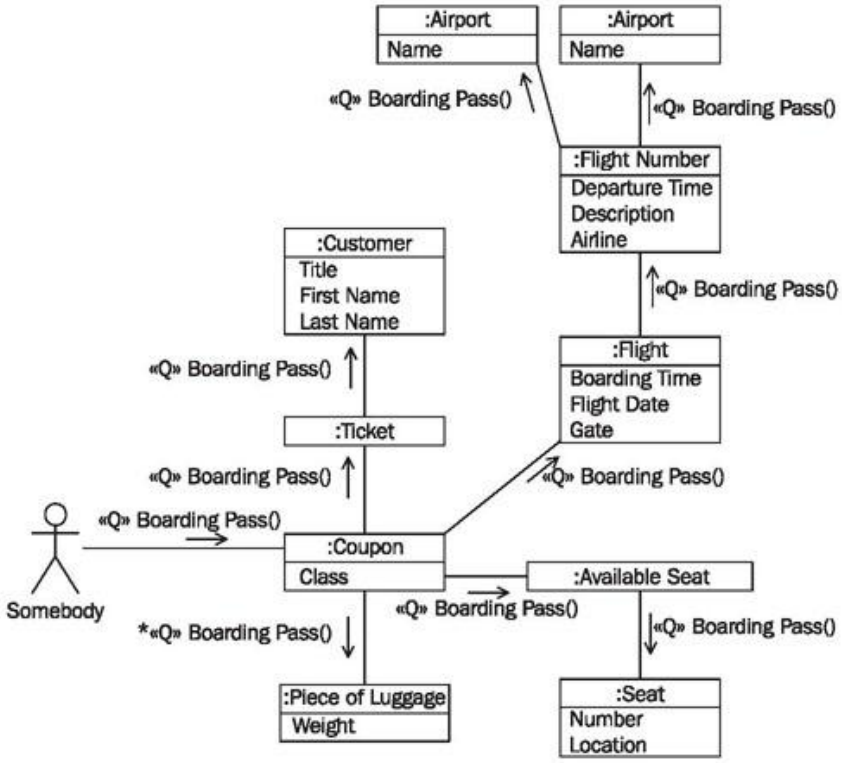


Gambar 2.19 *Sequence Diagram*

b. *Communication Diagram*

Communication diagram sejenis dengan diagram interaksi, yang lebih menekankan pada link data diantara bermacam-macam partisipan pada interaksi tersebut.

Berikut adalah contoh *Communication Diagram :*

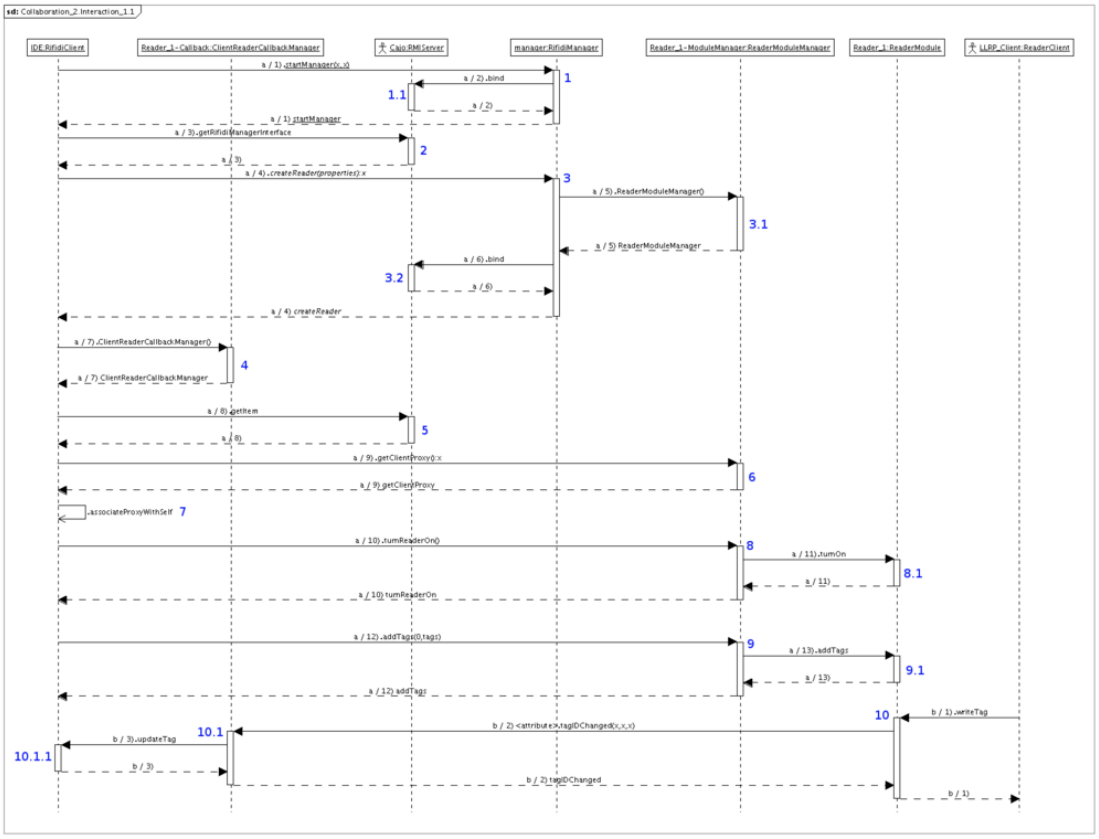


Gambar 2.20 *Communication Diagram*

c. *Timing Diagram*

Timing Diagram adalah bentuk lain dari *interaction* diagram, dimana fokus utamanya lebih ke waktu. Timing diagram sangat berdaya guna dalam menunjukkan faktor pembatas waktu diantara perubahan *state* pada objek yang berbeda.

Berikut adalah contoh *Timing Diagram:*

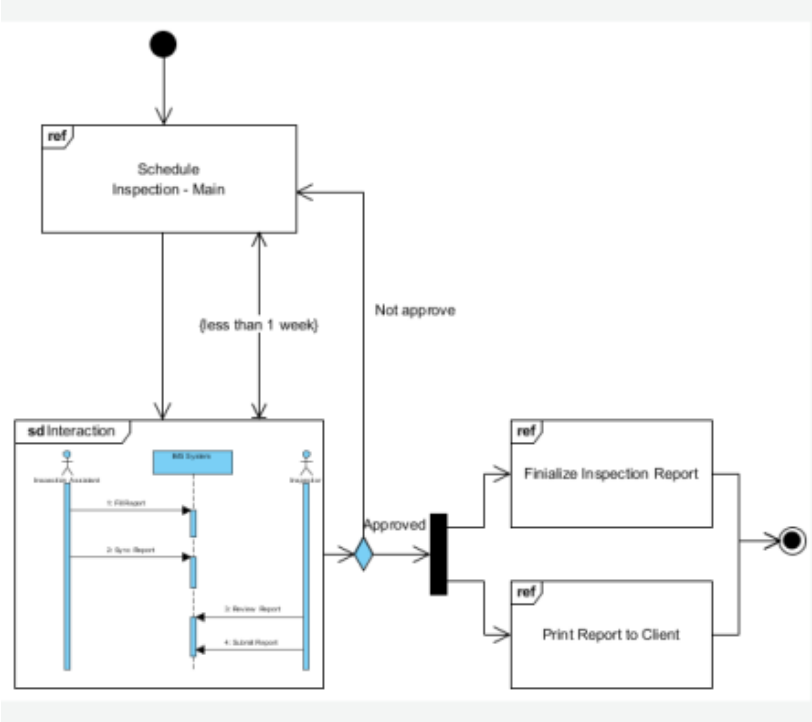


Gambar 2.21 *Timing Diagram*

d. *Interaction Overview Diagram*

Interaction Overview Diagram adalah pencangkokan secara bersama antara activity diagram dengan sequence diagram. *Interaction Overview* Diagram dapat dianggap sebagai *activity* diagram dimana semua aktivitas diganti dengan sedikit *sequence* diagram, atau bisa juga dianggap sebagai *sequence* diagram yang dirincikan dengan notasi *activity* diagram yang digunakan untuk menunjukkan aliran pengawasan.

Berikut adalah contoh *Interaction Overview* Diagram :



Gambar 2.22 *Interaction Overview Diagram*

2.7.5 Analisis Persyaratan Dengan UML

Analisis persyaratan meliputi usaha untuk mengetahui apa kemampuan sebuah sistem yang diinginkan pengguna dan pelanggan dari sebuah pembuat perangkat lunak [15]. Analisis ini dilakukan untuk mendapatkan informasi atau persyaratan cukup untuk mempersiapkan model yang menggambarkan apa yang diperlukan dari perspektif pengguna.

Diagram yang digunakan dalam analisis persyaratan yaitu:

1. *Use case diagram* yang digunakan untuk menunjukkan fungsionalitas suatu sistem dan bagaimana sistem berinterakasi dengan dunia luar.

2. *Activity diagram* yang menunjukkan alur kerja (*work flow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses.
3. *Class diagram* yang membantu dalam visualisasi struktur sistem yang mendeskripsikan jenis-jenis objek dalam suatu sistem dan hubungan yang terdapat diantara objek tersebut.

2.8 *Software* Pendukung Pemodelan UML (Rational Rose)

Rational Rose merupakan salah satu software yang paling banyak digunakan untuk melakukan design software melalui pendekatan UML (Unified Modelling Language). Rational Rose merupakan software yang menyediakan banyak fungsi - fungsi seperti : design proses, generate code, reverse engineering, serta banyak fungsi-fungsi yang lain. Rational Rose merupakan tool yang sangat mudah karena sudah menyediakan contoh-contoh design dari beberapa software [21].

2.9 Pengertian ASCII

Kode Standar Amerika untuk Pertukaran Informasi atau ASCII (*American Standard Code for Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti Hex dan Unicode tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Ia selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan biner sebanyak 8 bit. Dimulai dari 0000 0000 hingga

1111 1111. Total kombinasi yang dihasilkan sebanyak 256, dimulai dari kode 0 hingga 255 dalam sistem bilangan Desimal [10].

2.9.1 Tabel Karakter ASCII

Tabel berikut berisi karakter-karakter ASCII . Dalam sistem operasi Windows dan MS-DOS, pengguna dapat menggunakan karakter ASCII dengan menekan tombol Alt+[nomor nilai ANSI (desimal)]. Sebagai contoh, tekan kombinasi tombol **Alt+87** untuk karakter huruf latin "W" kapital.

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Tabel 2.3 Karakter ASCII