BAB 2
TINJAUAN PUSTAKA

2.1 Fatwa

2.1.1 Definisi fatwa

Dalam terminologi usul fiqh, kata fatwa diberi definisi oleh sebagian kalangan ulama usul fiqh sebagai "penjelasan hukum suatu masalah yang merupakan jawaban atas suatu pertanyaan". Definisi ini merupakan pemahaman dari apa yang didefinisikan oleh Ibn Hamdan al-Hanbali saat mengemukakan makna kata "mufti". Beliau menyatakan bahwa mufti adalah orang yang memberitahukan hukum Allah SWT karena pengetahuannya tentang dalil-dalil hukum tersebut.

Di dalam Kitab *Mafaahim Islaamiyyah* diterangkan sebagai berikut, *Secara literal, kata "al-fatwa"* mempunyai arti jawaban atas persoalan-persoalan syariat atau perundang-perundangan yang sulit. Bentuk jamaknya adalah *fataawin* dan *fataaway*. Jika dinyatakan *"aftay fi al-mas`alah* : menerangkan hukum dalam permasalahan tersebut. Sedangkan *al-iftaa`* adalah penjelasan hukum-hukum dalam persoalan-persoalan syariat, undang-undang, dan semua hal yang berkaitan dengan pertanyaan-pertanyaan orang yang bertanya (*ibaanat al-ahkaam fi al-mas`alah al-syar'iyyah, au qanuuniyyah, au ghairihaa mimmaa yata'allaqu bisu`aal al-saail*). *Al-Muftiy* adalah orang yang menyampaikan penjelasan hukum atau menyampaikan fatwa di tengah-tengah masyarakat. *Mufti* adalah seorang faqih yang diangkat oleh negara untuk menjawab persoalan-persoalan. Sedangkan menurut pengertian syariat, tidak ada perselisihan pendapat mengenai makna syariat dari kata *al-fatwa* dan *al-iftaa'* berdasarkan makna bahasanya. Oleh karena itu, fatwa secara syariat bermakna, penjelasan hukum syariat atas suatu permasalahan dari permasalahan-permasalah yang ada, yang didukung oleh dalil yang berasal dari al-Quran, Sunnah Nabawiyyah, dan ijtihad. Fatwa merupakan perkara yang

sangat urgen bagi manusia, dikarenakan tidak semua orang mampu menggali hukum-hukum syariat.

2.1.2 Kaedah Menggali Hukum atau Fatwa

Jika fatwa adalah penjelasan hukum syariat atas persoalan tertentu, maka, kaedah pengambilan fatwa tidak ubahnya dengan kaedah menggali hukum-hukum syariat dari dalil-dalil syariat (*ijtihad*). Pasalnya, satu-satunya cara untuk mengetahui hukum syariat dari dalil-dalil syariat adalah dengan ijtihad, tidak ada yang lain. Oleh karena itu, seorang *muftiy* tak ubahnya dengan seorang mujtahid.

2.1.3 Makna Ijtihad

Menurut Imam al-Amidiy, secara literal kata "ijtihad" bermakna ,"*Istafraagh al-wus'iy fi tahqiiq amr min al-umuur mustalzim li al-kalafat wa al-musyaqqaq" (*mencurahkan seluruh kemampuan dalam *mentahqiq* (meneliti dan mengkaji) suatu perkara yang meniscayakan adanya kesukaran dan kesulitan).

Imam Syaukaniy berpendapat, bahwa kata "*ijtihad"* diambil dari kata *al-juhd* yang bermakna *al-musyaqqah wa al-thaqah (kesukaran dan kemampuan).* Ijtihad digunakan secara khusus untuk menggambarkan sesuatu yang membawa konsekuensi kesulitan dan kesukaran (kemampuan paling optimal). Sedangkan suatu usaha yang tidak sampai pada taraf "kesukaran dan kesulitan" (musyaqqah) tidak dinamakan dengan ijtihad. Dalam kitab *al-Mahshuul* disebutkan, secara literal ijtihad bermakna "*istafraagh al-wus'iy fi ayy fi'li" (mencurahkan segenap kemampuan pada setiap perbuatan).* Untuk itu, kata *istafraagh al-wus'iy* hanya digunakan pada seseorang yang membawa beban yang sangat berat, tidak bagi orang yang membawa beban yang ringan.

Di kalangan 'ulama ushul, ijtihad diistilahkan dengan mencurahkan seluruh kemampuan untuk menggali hukum-hukum *syara' dari dalil-dalil dzanniy,* hingga batas dirinya merasa tidak mampu melakukan usaha lebih dari apa yang telah dicurahkannya.

Berdasarkan definisi di atas, dapatlah disimpulkan, bahwa *iijtihad* adalah proses menggali hukum *syara'* dari dalil-dalil yang bersifat *dzanniy* dengan

mencurahkan segenap tenaga dan kemampuan, hingga dirinya tidak mungkin lagi melakukan usaha lebih dari itu.

Dengan demikian, suatu aktivitas diakui sebagai ijtihad jika memenuhi tiga syarat berikut :

a. Ijtihad hanya melibatkan dalil-dalil yang bersifat *dzanniy*. Menurut al-Amidiy, hukum-hukum yang sudah qath'iy tidak digali berdasarkan proses ijtihad. Sebab hukum yang terkandung di dalam *nash-nash* yang *qath'iy (dilalahnya)* sudah sangat jelas, dan tidak membutuhkan interpretasi lain. Sebab, tidak ada pertentangan atau multi interpretasi pada *nash-nash yang qath'iy*. Oleh karena itu, ijtihad tidak berhubungan atau melibatkan dalil-dalil yang bersifat *qath'iy*, akan tetapi hanya melibatkan dalil-dalil yang bersifat *dzanniy*. Atas dasar itu, ijtihad tidak berlaku pada perkara-perkara 'aqidah, maupun hukum-hukum syara' yang *dilalahnya qath'iy*; misalnya wajibnya potong tangan bagi pencuri, had bagi pezina, bunuh bagi orang-orang yang murtad, dan lain sebagainya.

b. Ijtihad adalah proses menggali hukum *syara'*, bukan proses untuk menggali hal-hal yang bisa dipahami oleh akal secara langsung (*ma'qulaat*), maupun perkara-perkara yang bisa diindera (*al-mahsuusaat*). Penelitian dan uji coba di dalam laboratorium hingga menghasilkan sebuah teorema maupun hipotesa tidak disebut dengan ijtihad.

c. Ijtihad harus dilakukan dengan sungguh-sungguh dengan mengerahkan puncak tenaga dan kemampuan, hingga taraf ia tidak mungkin melakukan usaha lebih dari apa yang telah dilakukannya. Seseorang tidak disebut sedang melakukan ijtihad, jika ia hanya mencurahkan sebagian kemampuan dan tenaganya, padahal, ia masih mampu melakukan upaya lebih dari yang telah ia lakukan.

2.1.4 Syarat-syarat Mujtahid (Muftiy)

Seseorang layak melakukan ijtihad bila telah memenuhi syarat-syarat berikut ini:

a. Memahami *dalil-dalil sam'iyyah* yang digunakan untuk membangun kaedah-kaedah hukum. Yang dimaksud dengan dalil sam'iyyah adalah al-Quran, Sunnah, dan Ijma'. Seorang mujtahid harus memahami al-Quran, Sunnah, dan Ijma', klasifikasi dan kedudukannya. Seorang mujtahid juga harus memiliki kemampuan untuk memahami, menimbang, mengkompromikan, serta mentarjih dalil-dalil tersebut jika terjadi pertentangan. Kemampuan untuk memahami dalil-dalil sam'iyyah dan menimbang dalil-dalil tersebut merupakan syarat pokok bagi seorang mujtahid.

b. Memahami arah penunjukkan dari suatu lafadz (makna yang ditunjukkan lafadz) yang sejalan dengan lisannya orang Arab dan para *ahli balaghah.* Syarat kedua ini mengharuskan seseorang yang hendak berijtihad memiliki kemampuan dalam memahami seluk beluk bahasa Arab, atau kemampuan untuk memahami arah makna yang ditunjukkan oleh suatu lafadz. Oleh karena itu, seorang mujtahid atau mufti harus memiliki kemampuan bahasa yang mencakup kemampuan untuk memahami makna suatu lafadz, makna balaghahnya, dalalahnya, serta pertentangan makna yang dikandung suatu lafadz serta mana makna yang lebih kuat –setelah dikomparasikan dengan riwayat *tsiqqah* dan perkataan ahli bahasa. Seorang mujtahid tidak cukup hanya mengerti dan menghafal arti sebuah kata berdasarkan pedoman kamus. Akan tetapi, ia harus memahami semua hal yang berkaitan dengan kata tersebut dari sisi kebahasaan.

Berdasarkan penjelasan di atas dapat disimpulkan bahwa, penetapan fatwa harus didasarkan pada prinsip-prinsip ijtihad, yakni *"fahm al-nash" (memahami nash)* dan *fahm al-waaqi' al-haaditsah"* (memahami realitas yang terjadi). *Fahmu al-nash* adalah upaya memahami dalil-dalil syariat hingga diketahui *dilalah al-hukm* (penunjukkan hukum) yang terkandung di dalam dalil

tersebut. Sedangkan *fahmu al-waaqi' al-haaditsah* adalah upaya mengkaji dan meneliti realitas yang hendak dihukumi agar substansi persoalannya bisa diketahui, serta hukum syariat yang paling sesuai dengan realitas tersebut.
Realitas bukanlah dalil hukum (sumber hukum), akan tetapi realitas adalah obyek yang dihukumi. Oleh karena itu, fatwa tidak digali atau dirujuk dari realitas, akan tetapi diambil dan dirujuk dari dalil-dalil syariat (al-Quran, Sunnah, Ijma' Shahabat, dan Qiyas)[5].

2.2 Information Retrieval System

Information Retrieval System (Sistem Temu Kembali Informasi) adalah suatu sistem yang digunakan untuk menemukan kembali (*retrieve*) informasi-informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis.

Salah satu aplikasi dari Sistem Temu Kembali Informasi adalah mesin pencari yang dapat diterapkan di berbagai bidang. Pada mesin pencari dengan Sistem Temu Kembali Informasi pengguna dapat memasukkan *keyword* yang bebas dalam arti kata *keyword* yang sesuai dengan bahasa manusia dan sistem dapat menemukan dokumen yang sesuai dengan *keyword* yang ditulis oleh pengguna.

Model sistem Sistem Temu Kembali Informasi menentukan detail sistem Sistem Temu Kembali Informasi yaitu, meliputi:

1. Representasi dokumen dan *keyword.*
2. Fungsi pencarian.
3. Notasi kesesuaian (*relevance notation*) dokumen terhadap *keyword.*

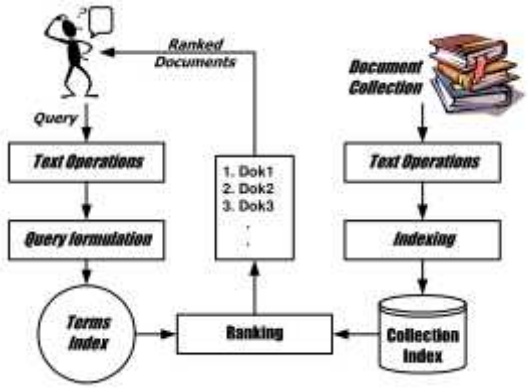
Yang dimaksud dengan representasi dokumen atau *keyword* adalah kumpulan kalimat yang menyatu menjadi paragrap kemudian menjadi bab dan menjadi satu buku, atau disebut juga dengan kumpulan kata yang menyusun menjadi kalimat. Sedangkan yang dimaksud dengan fungsi pencarian adalah

bagaimana mesin mengolah *keyword* untuk dicocokkan dengan dokumen, lalu mengambil dokumen yang relevan.

Bagian ini terbagi menjadi beberapa bagian, yaitu:

1. Cara memilih kata (*term*) untuk indeks.
2. Cara mengindeks kata.
3. Cara membobot kata.

Bagian yang paling signifikan pengaruhnya adalah pembobotan kata. Cara pembobotan kata mencirikan bagaimana sebuah sistem temu kembali informasi di bangun. Notasi kesesuaian adalah hubungan yang terjadi antara *query* dengan hasil pencarian. Sistem temu kembali informasi digunakan untuk menemukan kembali informasi-informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi secara otomatis.



Gambar 2. 1 Bagian-Bagian Sistem Temu Kembali Informasi

Gambar 2.1 memperlihatkan bahwa terdapat dua buah alur operasi pada sistem temu kembali informasi. Alur pertama dimulai dari koleksi dokumen dan alur kedua dimulai dari *query* pengguna. Alur pertama yaitu pemrosesan terhadap koleksi dokumen menjadi pangkalan data indeks dan tidak tergantung pada alur kedua. Sedangkan alur kedua tergantung dari keberadaan pangkalan data indeks yang dihasilkan pada alur pertama.

Bagian-bagian dari sistem Sistem Temu Kembali Informasi menurut gambar 2.1 meliputi:

1. *Text Operation* (operasi terhadap teks) yang meliputi pemilihan kata-kata dalam query maupun dalam dokumen (*term selection*) dalam transformasi dokumen atau query menjadi *term index* (index dari kata-kata).
2. *Query formulation* (formulasi terhadap *query*) yaitu memberi bobot pada kata indeks *query*.
3. *Ranking* (perangkingan), mencari dokumen-dokumen yang relevan terhadap *query* dan mengurutkan dokumen tersebut berdasarkan kesesuaiannya dengan *query*.
4. *Indexing* (indeks), membangun pangkalan data indeks dari koleksi dokumen. Dilakukan terlebih dahulu sebelum pencarian dokumen dilakukan.

Sistem Temu Kembali Informasi menerima *query* dari pengguna, kemudian melakukan perangkingan terhadap dokumen pada koleksi berdasarkan kesesuaiannya dengan *query*. Hasil perangkingan yang diberikan kepada pengguna merupakan dokumen yang menurut sistem relevan dengan *query*. Namun relevansi dokumen terhadap suatu query merupakan penilaian pengguna yang subjektif dan dipengaruhi banyak faktor seperti topik, pewaktuan, sumber informasi maupun tujuan pengguna.

Sistem temu kembali informasi terutama berhubungan dengan pencarian informasi yang isinya tidak memiliki struktur. Demikian pula ekspresi kebutuhan pengguna yang disebut *query*, juga tidak memiliki struktur. Hal ini yang membedakan sistem temu kembali informasi dengan sistem basis data. Dokumen adalah contoh informasi yang tidak terstruktur. Isi dari suatu dokumen sangat tergantung pada pembuat dokumen tersebut.

Sistem Temu Kembali Informasi sebagai sistem yang berfungsi untuk menemukan informasi yang relevan dengan kebutuhan pemakai, merupakan salah satu tipe sistem informasi. Salah satu hal yang perlu diingat adalah bahwa

informasi yang diproses terkandung dalam sebuah dokumen yang bersifat tekstual.

Dalam konteks ini, temu kembali informasi berkaitan dengan representasi, penyimpanan, dan akses terhadap dokumen representasi dokumen. Dokumen yang ditemukan tidak dapat dipastikan apakah relevan dengan kebutuhan informasi pengguna yang dinyatakan dalam *query*. Pengguna Sistem Temu Kembali Informasi sangat bervariasi dengan kebutuhan informasi yang berbeda-beda.

Tujuan dari sistem Sistem Temu Kembali Informasi adalah:

1. Menemukan seluruh dokumen yang relevan terhadap suatu *query*.
2. Hanya menemukan dokumen relevan saja, artinya tidak terdapat dokumen yang tidak relevan pada dokumen hasil pencarian.[6]

2.3 Proses Indexing

Indexing subsystem adalah proses *subsystem* yang merepresentasikan koleksi dokumen kedalam bentuk tertentu untuk memudahkan dan mempercepat proses pencarian dan penemuan kembali dokumen yang relevan.

Pembangunan index dari koleksi dokumen merupakan tugas pokok pada tahapan preprocessing di dalam IR. Kualitas index mempengaruhi efektifitas dan efisiensi sistem IR. Index dokumen adalah himpunan *term* yang menunjukkan isi atau topik yang dikandung oleh dokumen. Index akan membedakan suatu dokumen dari dokumen lain yang berada di dalam koleksi. Ukuran index yang kecil dapat memberikan hasil buruk dan mungkin beberapa item yang relevan terabaikan. Index yang besar memungkinkan ditemukan banyak dokumen yang relevan tetapi sekaligus dapat menaikkan jumlah dokumen yang tidak relevan dan menurunkan kecepatan pencarian (*searching*).

Pembuatan *inverted index* harus melibatkan konsep *linguistic processing* yang bertujuan mengekstrak *term-term* penting dari dokumen yang direpresentasikan sebagai *bag-of-words*. Ekstraksi term biasanya melibatkan dua operasi utama berikut:

1. Penghapusan ***stop-words***. *Stop-word* didefinisikan sebagai *term* yang tidak berhubungan (*irrelevant*) dengan subyek utama dari *database* meskipun kata tersebut sering kali hadir di dalam dokumen. Berikut ini adalah Contoh *stop wrds* dalam bahasa inggris :*a, an, the, this, that, these, those, her, his, its, my, our, their, your, all, few, many, several, some, every, for, and, nor, bit, or, yet, so, also, after, although, if, unless, because, on, beneath, over, of, during, beside,* dan *etc*. Contoh *stop words* dalam bahasa Indonesia : yang, juga, dari, dia, kami, kamu, aku, saya, ini, itu, atau, dan, tersebut, pada, dengan, adalah, yaitu, ke, tak, tidak, di, pada, jika, maka, ada, pun, lain, saja, hanya, namun, seperti, kemudian, dll. *Stop-words* termasuk pula beberapa kata tertentu yang didefinisikan terkait dengan topik *database*, misal pada *database* yang menampung daftar karya tulis (*paper*) penelitian terkait dengan *heart diseases*, maka kata *heart* dan *disease* sebaiknya dihapus.

2. ***Stemming***. Kata-kata yang muncul di dalam dokumen sering mempunyai banyak varian morfologik. Karena itu, setiap kata yang bukan *stop-words* direduksi ke bentuk *stemmed word* (*term*) yang cocok. Kata tersebut distem untuk mendapatkan bentuk akarnya dengan menghilangkan awalan atau akhiran. Dengan cara ini, diperoleh kelompok kata yang mempunyai makna serupa tetapi berbeda wujud sintaktis satu dengan lainnya. Kelompok tersebut dapat direpresentasikan oleh satu kata tertentu. Sebagai contoh, kata menyebutkan, tersebut, disebut dapat dikatakan serupa atau satu kelompok dan dapat diwakili oleh satu kata umum sebut.

Terdapat 5 langkah pembangunan *inverted index*, yaitu:

a. **Penghapusan format dan markup dari dalam dokumen.**

Tahap ini menghapus semua *tag markup* dan format khusus dari dokumen, terutama pada dokumen yang mempunyai banyak *tag* dan format seperti dokumen (X)HTML.

b. **Pemisahan rangkaian kata (*tokenization*).**

Tokenization adalah tugas memisahkan deretan kata di dalam kalimat, paragraf atau halaman menjadi *token* atau potongan kata tunggal atau *termmed word*. Tahapan ini juga menghilangkan karakter-karakter tertentu seperti tanda baca dan mengubah semua *token* ke bentuk huruf kecil (*lower case*).

c. **Penyaringan (*filtration*)**

Pada tahapan ini ditentukan *term* mana yang akan digunakan untuk merepresentasikan dokumen sehingga dapat mendeskripsikan isi dokumen dan membedakan dokumen tersebut dari dokumen lain di dalam koleksi. Term yang sering dipakai tidak dapat digunakan untuk tujuan ini, setidaknya karena dua hal. Pertama, jumlah dokumen yang relevan terhadap suatu *query* kemungkinan besar merupakan bagian kecil dari koleksi. *Term* yang efektif dalam pemisahan dokumen yang relevan dari dokumen tidak relevan kemungkinan besar adalah *term* yang muncul pada sedikit dokumen. Kedua, *term* yang muncul dalam banyak dokumen tidak mencerminkan definisi dari topik atau sub-topik dokumen. Karena itu, term yang sering digunakan dianggap sebagai *stop-word* dan dihapus.

d. **Konversi *term* ke bentuk dasar (*stemming*).**

Stemming adalah proses konversi *term* ke bentuk umumnya, sebagaimana dijelaskan sebelumnya. Dokumen dapat pula diekspansi dengan mencarikan sinonim bagi term-term tertentu di dalamnya. Sinonim adalah kata-kata yang mempunyai pengertian serupa tetapi berbeda dari sudut pandang morfologis. Seperti

stemming, operasi ini bertujuan menemukan suatu kelompok kata terkait. Akan tetapi sinonim bekerja berdasarkan pada thesaurus, tidak berbagi-pakai term stem. Jika pengguna memasukkan *query* "*heart disease*" maka *query* diekspansi untuk mengakomodasi semua sinonim dari *disease* seperti *ailment, complication, condition, disorder, fever, ill, illness, infirmity, malady, sickness*, dan lain-lain.

e. **Pemberian bobot terhadap term (*weighting*).**
Setiap *term* diberikan bobot sesuai dengan skema pembobotan yang dipilih, apakah pembobotan lokal, global atau kombinasi keduanya. Banyak aplikasi menerapkan pembobotan kombinasi berupa perkalian bobot lokal *term frequency* dan *global inverse document frequency*, ditulis tf .idf.

2.3.1 Stemming Bahasa Indonesia dengan Algoritma Nazief dan Andriani

Algoritma stemming untuk bahasa yang satu berbeda dengan algoritma stemming untuk bahasa lainnya. Sebagai contoh bahasa Inggris memiliki morfologi yang berbeda dengan bahasa Indonesia sehingga algoritma stemming untuk kedua bahasa tersebut juga berbeda. Proses stemming pada teks berbahasa Indonesia lebih rumit/kompleks karena terdapat variasi imbuhan yang harus dibuang untuk mendapatkan root word (kata dasar) dari sebuah kata.

Pada umumnya kata dasar pada bahasa Indonesia terdiri dari kombinasi: **Prefiks 1 + Prefiks 2 + Kata dasar + Sufiks 3 + Sufiks 2 + Sufiks 1**

Algoritma Nazief & Adriani yang dibuat oleh Bobby Nazief dan Mirna Adriani ini memiliki tahap-tahap sebagai berikut:

1. Pertama cari kata yang akan diistem dalam kamus kata dasar. Jika ditemukan maka diasumsikan kata adalah root word. Maka algoritma berhenti.
2. Inflection Suffixes ("-lah", "-kah", "-ku", "-mu", atau "-nya") dibuang. Jika berupa particles ("-lah", "-kah", "-tah" atau "-pun")

maka langkah ini diulangi lagi untuk menghapus Possesive Pronouns (“-ku”, “-mu”, atau “-nya”), jika ada.

3. Hapus Derivation Suffixes (“-i”, “-an” atau “-kan”). Jika kata ditemukan di kamus, maka algoritma berhenti. Jika tidak maka ke langkah 3a
   a. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k” juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3b.
   b. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4
4. Hilangkan derivation prefixes DP {“di-”,“ke-”,“se-”,“me-”,“be-”,“pe”, “te-”} dengan iterasi maksimum adalah 3 kali:
   a. Langkah 4 berhenti jika:
      - Terjadi kombinasi awalan dan akhiran yang terlarang seperti pada Tabel 1
      - Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
      - Tiga awalan telah dihilangkan.

Tabel 2. 1 Kombinasi Awalan Akhiran yang tidak diijinkan

Awalan	Akhiran yang tidak diizinkan
be-	-i
di-	-an
ke-	-i, -kan
me-	-an
se-	-i, -kan
te-	-an

   b. Identifikasikan tipe awalan dan hilangkan. Awalan ada tipe:
      - Standar: “di-”, “ke-”, “se-” yang dapat langsung dihilangkan dari kata.

- Kompleks: “me-”, “be-”, “pe”, “te-” adalah tipe-tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya. Oleh karena itu, gunakan aturan pada Tabel 2 untuk mendapatkan pemenggalan yang tepat.

Tabel 2. 2 Aturan Pemenggalan Awalan *Stemmer* Nazief dan Adriani

Aturan	Format Kata	Pemenggalan
1	berV...	ber-V...|be-erV...
2	berCAP...	ber-CAP... dimana C!=’r’ & P!=’er’
3	berCAerV...	ber-CaerV... dimana C!=’r’
4	belajar	bel-ajar
5	beC1erC2...	be-C1erC2... dimana C1 !={‘r’|’l’}
6	terV...	ter-V... | te-rV...
7	terCerV...	ter-VerV... dimana C!=’r’ dan P!=’er’
8	terCP...	ter-CP... dimana C!=’r’ dan P!=’er’
9	teC1erC2...	te-C1erC2... dimana C1 !=’r’
10	me{l|r|w|y}V...	me-{l|r|w|y}V...
11	mem{b|f|v}...	mem-{b|f|v}...
12	Mempe{r|l}...	mem-pe...
13	Mem{rV|V}...	me-m{rV|V}... |me-p{rV|V}...
14	Men{c|d|j|z}...	men-{c|d|j|z}...
15	menV...	me-nV...| me-tV
16	Meng{g|h|q}...	meng-{g|h|q}...
17	mengV...	meng-V... |meng-kV...
18	menyV...	meny-sV...
19	mempV...	mem-pV... dimana V!=’e’
20	pe{w|y}V...	pe-{w|y}V...
21	perV...	per-V... |per-pV...
22	perCAP...	per-CAP... dimana C!=’r’ dan P!=’er’
23	perCAerV...	per-CaerV... dimana C!=’r’
24	pem{b|f|V}...	pem-{b|f|V}...
25	pem{rV|V}...	pe-m{rV|V}... |pe-p{rV|V}...
26	pen{c|d|j|za}...	pen-{c|d|j|z}...
27	penV...	pe-nV... | pe-tV...
28	peng{g|h|q}...	peng-{g|h|q}...
29	pengV...	peng-V... |peng-kV...
30	penyV...	peny-sV...
31	pelV...	pe-lV... kecuali “pelajar” yang menghasilkan “ajar”
32	peCerV...	per-erV... dimana C!={r|w|y|l|m|n}
33	peCP...	pe-CP... dimana C!={r|w|y|l|m|n} dan P!=’er’

c. Cari kata yang telah dihilangkan awalannya ini di dalam kamus. Apabila tidak ditemukan, maka langkah 4 diulangi kembali. Apabila ditemukan, maka keseluruhan proses dihentikan.

5. Apabila setelah langkah 4 kata dasar masih belum ditemukan, maka proses recoding dilakukan dengan mengacu pada aturan pada Tabel

2.2 Recoding dilakukan dengan menambahkan karakter recoding di awal kata yang dipenggal. Pada Tabel 2, karakter recoding adalah huruf kecil setelah tanda hubung (‘-’) dan terkadang berada sebelum tanda kurung. Sebagai contoh, kata “menangkap” (aturan 15), setelah dipenggal menjadi “nangkap”. Karena tidak valid, maka recoding dilakukan dan menghasilkan kata “tangkap”.

6. Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai root word. Proses selesai.

Tipe awalan ditentukan melalui langkah-langkah berikut:

1. Jika awalannya adalah: “di-”, “ke-”, atau “se-” maka tipe awalannya secara berturut-turut adalah “di-”, “ke-”, atau “se-”.
2. Jika awalannya adalah “te-”, “me-”, “be-”, atau “pe-” maka dibutuhkan sebuah proses tambahan untuk menentukan tipe awalannya.
3. Jika dua karakter pertama bukan “di-”, “ke-”, “se-”, “te-”, “be-”, “me-”, atau “pe-” maka berhenti.
4. Jika tipe awalan adalah “none” maka berhenti. Jika tipe awalan adalah bukan “none” maka awalan dapat dilihat pada Tabel 2.3 Hapus awalan jika ditemukan.

Tabel 2. 3 Cara Menentukan Tipe Awalan Untuk Awalan “te-“

Following Characters				**Tipe awalan**
Set 1	**Set 2**	**Set 3**	**Set 4**	
“-r-“	“-r-“	-	-	none
“-r-“	-	-	-	ter-luluh
“-r-“	not (vowel or “-r-”)	“-er-“	vowel	ter
“-r-“	not (vowel or “-r-”)	“-er-“	not vowel	ter-
“-r-“	not (vowel or “-r-”)	not “-er-“	-	ter
not (vowel or “-r-”)	“-er-“	vowel	-	none
not (vowel or “-r-”)	“-er-“	not vowel	-	none

Tabel 2. 4 Jenis Awalan Berdasarkan Tipe Awalannya

Tipe awalan	Awalan yang harus dihapus
di-	di-
ke-	ke-
se-	de-
te-	te-
ter-	ter-
ter-luluh	ter

Untuk mengatasi keterbatasan pada algoritma di atas, maka ditambahkan aturan-aturan dibawah ini :

1. Aturan untuk reduplikasi.
    a. Jika kedua kata yang dihubungkan oleh kata penghubung adalah kata yang sama maka root word adalah bentuk tunggalnya, contoh : “buku-buku” root word-nya adalah “buku”.
    b. Kata lain, misalnya “bolak-balik”, “berbalas-balasan, dan ”seolah-olah”. Untuk mendapatkan root word-nya, kedua kata diartikan secara terpisah. Jika keduanya memiliki root word yang sama maka diubah menjadi bentuk tunggal, contoh: kata “berbalas-balasan”, “berbalas” dan “balasan” memiliki root word yang sama yaitu “balas”, maka root word “berbalas-balasan” adalah “balas”. Sebaliknya, pada kata “bolak-balik”, “bolak” dan “balik” memiliki root word yang berbeda, maka root word-nya adalah “bolak-balik”.
2. Tambahan bentuk awalan dan akhiran serta aturannya.
    a. Untuk tipe awalan “mem-“, kata yang diawali dengan awalan “memp-” memiliki tipe awalan “mem-”.
    b. Tipe awalan “meng-“, kata yang diawali dengan awalan “mengk-” memiliki tipe awalan “meng-”.

Kelebihan dan Kelemahan Algoritma Nazief dan Adriani

- Kelebihan :
    1. Memperhatikan kemungkinan adanya partikel-partikel yang mungkin mengikuti suatu kata berimbuhan.

2. Proses stemming dokumen teks berBahasa Indonesia menggunakan Algoritma Nazief dan Adriani memiliki prosentase keakuratan (presisi) lebih besar dibandingkan dengan stemming menggunakan Algoritma Porter.

- Kelemahan :
    1. Penyamarataan makna variasi kata
    2. Jumlah database kata dan kata dasarnya harus besar. Kesalahan terjadi bila kata tidak ditemukan di database dan kemudian dianggap kata dasar, padahal bukan
    3. Lamanya waktu yang diperlukan dalam proses pencarian kata di dalam kamus. [7]

2.4 Term-Document Matrix

Term-Document matriks adalah sebuah string yang berisikan jumlah kemunculan term dalam suatu lokasi dokumen. Terms yang digunakan dalam hal ini tidak meliputi *terms* berupa frasa namun hanya dibatasi pada *terms* berupa kata tunggal, karena penyimpanan frasa akan menambah jumlah kata yang tidak sesuai karena frasa terdiri atas 1 atau lebih kata. Pada matriks ini, kolom mewakili dokumen, dan baris mewakili *terms.*

Term-document matrix dapat berukuran sangat besar, terdiri atas ribuan baris dan kolom, dan tahap selanjutnya perlu dilakukan dekomposisi terhadap matriks ini, maka untuk mempersingkat proses, perlu dilakukan penyederhanaan terhadap *term-document matrix.*

Term-document matrix hanya perlu menyimpan *term* yang dapat menjelaskan isi dari suatu dokumen atau disebut juga sebagai *content words.* Kata-kata yang merupakan *stopwords*, yakni kata-kata yang tidak memiliki makna jika berdiri sendiri dapat diabaikan.

Dengan *term-document matrix* yang ada, bobot setiap kata (*term weight*) hanya bergantung pada jumlah kata pada tiap dokumen. Bobot tersebut disebut sebagai *local weight* karena hanya berlaku pada tiap-tiap dokumen. Pada perhitunga *local weight, terms* yang muncul berulang kali pada suatu dokumen

akan memiliki bobot yang lebih besar. Bobot *terms* untuk keseluruhan koleksi dokumen disebut sebagai *global weight*, perhitungan *global weight* berkebalikan dengan *local weight*, di mana *terms* yang muncul pada dokumen tertentu saja dianggap lebih signifikan sehingga memiliki bobot yang lebih besar.

2.5 Latent Semantic Indexing (LSI)

Mesin pencari (*search engine*) memegang peranan penting dalam pencarian informasi. Kebanyakan mesin pencari hanya membandingkan isi dari dokumen terhadap *query* berdasarkan kata kunci yang tersedia berupa statistik kemunculan kata. Masalah akan muncul pada saat sebuah kata yang secara statistik berbeda tetapi memiliki makna yang sama (sinonim). Hal ini mengakibatkan recall yang rendah karena mesin pencari akan mengembalikan dokumen sedikit. Masalah lain ialah apabila ada satu kata yang secara statistik sama tetapi memiliki banyak makna (polisemi) yang akan menghasilkan nilai precision yang rendah karena mesin pencari akan mengembalikan dokumen yang tidak sesuai dengan keinginan pengguna. Hal ini dapat mengakibatkan kinerja mesin pencari menjadi kurang baik karena tidak memperhatikan kata secara semantik.

Pendekatan lainnya untuk mengatasi permasalahan ini ialah menggunakan *latent semantic indexing* (LSI) dengan pendekatan ruang vektor. LSI adalah suatu teknik yang memetakan *query* dan dokumen ke dalam suatu ruang vektor yang disebut *latent semantic space*. Dalam *latent semantic space*, suatu *query* dan dokumen tersebut tidak memiliki nilai kesamaan yang tinggi walaupun kueri dan dokumen tersebut tidak memiliki term yang sama. Selama term tersebut mirip secara semantic, nilai kesamaan yang dihasilkan akan tinggi. Pada dasarnya, LSI menggunakan teknik *Singular Value Decomposition* (SVD) untuk mendekomposisikan matriks *term*-dokumen. Dengan mengurangi ruang *term* dan dokumen menjadi dimensi yang lebih kecil, *SVD* menampakkan hubungan yang mendasari *term* dan dokumen dalam semua kombinasi yang memungkinkan dan membuang *noise* yang ada pada ruang vektor.

Proses pencarian dengan metode LSI pada sistem ini menerima masukan berupa sebuah *keyword* yang akan dicari pada dokumen. Pada proses pencarian dengan metode LSI kata-kata yang unik pada setiap dokumen akan direpresentasikan sebagai baris matriks dan dokumen-dokumen akan direpresentasikan sebagai kolom matriks. Nilai dari matriks tersebut adalah banyaknya kemunculan sebuah kata di setiap dokumen yang akan dibandingkan.[8]

2.5.1 Singular Value Decomposition (SVD)

Pembuatan matriks term-dokumen bertujuan untuk membuat indeks dari kumpulan dokumen yang disimpan. Pada metode ruang vektor, matriks term-dokumen yang dibuat berukuran m x n, dimana m adalah banyaknya term pada kumpulan dokumen. Terdapat dua masalah utama yang ditemui dalam metode ruang vektor yaitu: masalah akurasi dan masalah ruang penyimpanan yang dibutuhkan untuk menyimpan matriks term-dokumen.

Sebagian besar sistem temu kembali informasi mempresentasikan dokumen dan *query* menggunakan kata-kata yang terdapat di dalamnya (termasuk metode ruang vektor), dan menggunakan jumlah kata yang sama yang terdapt pada *query* dan dokumen sebagai dasar perbandingan. Semakin banyak kata dalam dokumen yang sama dengan kata-kata *query*, semakin tinggi peringkat kesesuaian dokumen tersebut.

Ada dua masalah yang terjadi jika menggunakan cara diatas dalam sistem temu kembali dokumen, yaitu:

a. Kata-kata bersifat ambigu, yang mengakibatkan temu kembali menghasilkan dokumen yang tidak relevan. Masalah ini terjadi karena adanya kata-kata yang sama penulisannya namun memiliki arti berbeda (polysemy).
b. Suatu dokumen bisa saja relevan terhadap suatu *query* meskipun tidak mengandung kata-kata yang terdapat dalam query tersebut. Hal ini karena suatu konsep bisa didefinisikan dengan banyak cara (synonym).

Latent Semantic Indexing merupakan pengembangan dari metode ruang vektor. Tujuannya adalah mengatasi kedua masalah diatas. Pada LSI tidak menggunakan term-dokumen yang utuh melainkan pendekatannya saja. Untuk membentuk pendekatan dari matriks term-dokumen digunakan Singular Value Decomposition (SVD). SVD adalah suatu teknik untuk mendekomposisi matriks. Misalkan A adalah matriks term-dokumen, maka SVD dari A adalah $A = USV^T$, dimana U dan V adalah matriks ortogonal dan S adalah matriks diagonal.

Karena hasil SVD sebagian cukup dekat dengan matriks semula, maka sistem temu kembali berbasis SVD sebagian bisa diharapkan mempunyai sebaik sistem berbasis matriks semula.

Kelemahan dari SVD adalah ruang yang dibutuhkan untuk penyimpanan cukup besar. Pada metode ruang vektor hanya A yang perlu disimpan. Karena matriks A adalah matriks jarang, hanya sebagian kecil elemennya saja yang perlu disimpan. Pada LSI berbasis SVD, harus disimpan matriks U, V, dan S.[6]

Tahapan perhitungan matriks menggunakan *Singular Value Decomposistion* dapat dilihat dalam tahapan berikut :

1. Membuat matrik A
2. Hitung A^TA (2.1)
3. Menghitung determinan dari $|A^TA - CI| = 0$ (2.2)
4. Menghitung nilai Singular Value dan S^{-1}
5. Menghitung nilai Eigenvector dan normalisasi dari $(A^TA - c_iI)X_1 = 0$. (2.3)
6. Membentuk matrik V dari hasil normalisasi eigenvektor.
7. Membentuk matrik U dengan $U = AVS^{-1}$ (2.4)
8. Menghitung vektor dari *queri* $q = q^TU_kS_k^{-1}$ (2.5)

Dalam perhitungan pembobotan antara hubungan *query* dengan dokumen digunakan rumus *Cosine Similarity* untuk mendapatkan dokumen yang relevan terhadap *query* masukan dengan rumus :

$$sim(q,d) = \frac{q.d}{|q||d|} \quad (2.6)$$

Rumus diatas digunakan untuk mendapatkan nilai cosinus minimal = 0 dan maksimalnya = 1. [9]

2.6 Android

Android merupakan subset perangkat lunak untuk perangkat meliputi sistem operasi, *middleware* dan aplikasi. Sedangkan Android SDK (dan API yang diperlukan untuk mengembangkan aplikasi pada platform Android dengan menggunakan bahasa pemrograman Java. Dikembangkan bersama antara Google, HTC, Intel, Motorola, Qualcomm, T-Mobile, NVIDIA yang tergabung dalam OHA (*Open Handset Alliance*) dengan tujuan membuat sebuah standar terbuka untuk perangkat bergerak (*mobile device*)..

Berikut adalah perkembangan versi android :

1. **Android versi 1.1**

   Pada 9 Maret 2009, Google merilis Android versi 1.1. Android versi ini dilengkapi dengan pembaruan estetis pada aplikasi, jam alarm, *voice search* (pencarian suara), pengiriman pesan dengan Gmail, dan pemberitahuan email.

2. **Android versi 1.5 (Cupcake)**

   Pada pertengahan Mei 2009, Google kembali merilis telepon seluler dengan menggunakan Android dan SDK (*Software Development* Kit) dengan versi 1.5 (Cupcake). Terdapat beberapa pembaruan termasuk juga penambahan beberapa fitur dalam seluler versi ini yakni kemampuan merekam dan menonton video dengan modus kamera, mengunggah video ke Youtube dan gambar ke Picasa langsung dari telepon, dukungan Bluetooth A2DP, kemampuan terhubung secara otomatis ke headset

Bluetooth, animasi layar, dan keyboard pada layar yang dapat disesuaikan dengan sistem.

3. **Android versi 1.6 (Donut)**

   Donut (versi 1.6) dirilis pada September dengan menampilkan proses pencarian yang lebih baik dibanding sebelumnya, penggunaan baterai indicator dan kontrol applet VPN. Fitur lainnya adalah galeri yang memungkinkan pengguna untuk memilih foto yang akan dihapus pada kamera, camcorder dan galeri yang dintegrasikan pada CDMA / EVDO, 802.1x, VPN, Gestures, dan Textto- speech engine. Kemampuan dial kontak teknologi text to change speech (tidak tersedia pada semua ponsel).

4. **Android versi 2.0/2.1 (Eclair)**

   Pada 3 Desember 2009 kembali diluncurkan ponsel Android dengan versi 2.0/2.1 (Eclair), perubahan yang dilakukan adalah pengoptimalan hardware, peningkatan Google Maps 3.1.2, perubahan UI dengan *browser* baru dan dukungan HTML5, daftar kontak yang baru, dukungan flash untuk kamera 3,2 MP, digital Zoom, dan Bluetooth 2.1. Untuk bergerak cepat dalam persaingan perangkat generasi berikut, Google melakukan investasi dengan mengadakan kompetisi aplikasi Mobile terbaik (killer apps - aplikasi unggulan). Kompetisi ini berhadiah $25,000 bagi setiap pengembang aplikasi terpilih. Kompetisi diadakan selama dua tahap yang tiap tahapnya dipilih 50 aplikasi terbaik. Dengan semakin berkembangnya dan semakin bertambahnya jumlah handset Android, semakin banyak pihak ketiga yang berminat untuk menyalurkan aplikasi mereka kepada sistem operasi Android. Aplikasi terkenal yang diubah ke dalam sistem operasi Android adalah Shazam, Backgrounds, dan WeatherBug. Sistem operasi Android dalam situs Internet juga dianggap penting untuk menciptakan aplikasi Android asli, contohnya oleh MySpace dan Facebook.

5. **Android versi 2.2 (Froyo: Frozen Yoghurt)**

   Pada 20 Mei 2010, Android versi 2.2 (Froyo) diluncurkan. Perubahanperubahan umumnya terhadap versi-versi sebelumnya antara

lain dukungan Adobe Flash 10.1, kecepatan kinerja dan aplikasi 2 sampai 5 kali lebih cepat, intergrasi V8 JavaScript engine yang dipakai Google Chrome yang mempercepat kemampuan rendering pada browser, pemasangan aplikasi dalam *SD Card*, kemampuan WiFi Hotspot portabel, dan kemampuan *auto update* dalam aplikasi Android Market.

6. **Android versi 2.3 (Gingerbread)**

   Pada 6 Desember 2010, Android versi 2.3 (Gingerbread) diluncurkan. Perubahan-perubahan umum yang didapat dari Android versi ini antara lain peningkatan kemampuan permainan (*gaming*), peningkatan fungsi *copy paste*, layar antar muka *(User Interface)* didesain ulang, dukungan format video VP8 dan WebM, efek audio baru *(reverb, equalization, headphone virtualization*, dan *bass boost)*, dukungan kemampuan *Near Field Communication (NFC)*, dan dukungan jumlah kamera yang lebih dari satu.

7. **Android versi 3.0 (Honeycomb)**

   Android Honeycomb dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. *User Interface* pada Honeycomb juga berbeda karena sudah didesain untuk tablet. Android Honeycomb dirancang khusus untuk tablet. Android versi ini mendukung ukuran layar yang lebih besar. User Interface pada Honeycomb juga berbeda karena sudah didesain untuk tablet. Honeycomb juga mendukung multi prosesor dan juga akselerasi perangkat keras (hardware) untuk grafis. Tablet pertama yang dibuat dengan menjalankan Honeycomb adalah Motorola Xoom. Perangkat tablet dengan platform Android 3.0 akan segera hadir di Indonesia. Perangkat tersebut bernama Eee Pad Transformer produksi dari Asus. Rencana masuk pasar Indonesia pada Mei 2011.

8. **Android versi 4.0 (ICS: Ice Cream Sandwich)**

   Diumumkan pada tanggal 19 Oktober 2011, membawa fitur Honeycomb untuk smartphone dan menambahkan fitur baru termasuk membuka kunci dengan pengenalan wajah, jaringan data pemantauan penggunaan dan kontrol, terpadu kontak jaringan sosial, perangkat tambahan fotografi, mencari email secara offline, dan berbagi informasi dengan menggunakan NFC. Ponsel pertama yang menggunakan sistem operasi ini adalah Samsung Galxy Nexus.

9. **Android versi 4.1 (Jelly Bean)**

   Android Jelly Bean yaang diluncurkan pada acara Google I/O lalu membawa sejumlah keunggulan dan fitur baru. Penambahan baru diantaranya meningkatkan input keyboard, desain baru fitur pencarian, UI yang baru dan pencarian melalui Voice Search yang lebih cepat. Tak ketinggalan Google Now juga menjadi bagian yang diperbarui. Google Now memberikan informasi yang tepat pada waktu yang tepat pula. Salah satu kemampuannya adalah dapat mengetahui informasi cuaca, lalu-lintas, ataupun hasil pertandingan olahraga. Sistem operasi Android Jelly Bean 4.1 muncul pertama kali dalam produk tablet Asus, yakni Google Nexus 7.

2.7 Unified Modeling Language (UML)

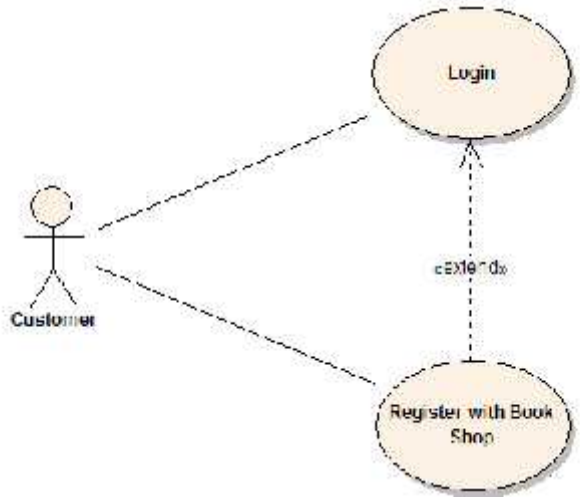
Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem.

Pada UML dikenal beberapa diagram, diantaranya sebagai berikut:

2.7.1 Use Case Diagram

Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem. Use case merupakan sebuah pekerjaan tertentu, misalnya login ke

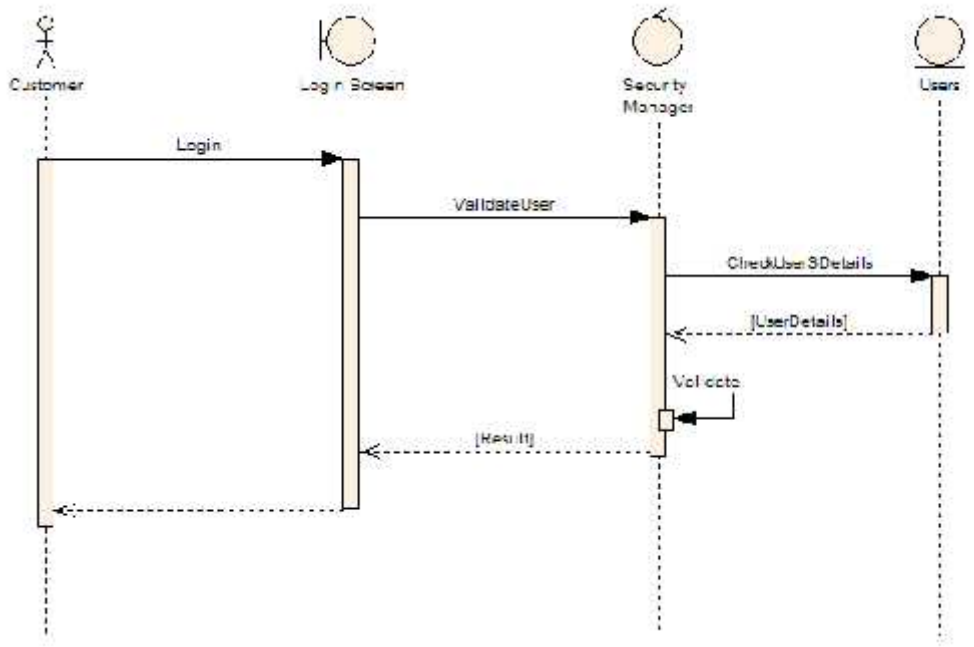
sistem, meng-create sebuah daftar belanja, dan sebagainya. Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.



Gambar 2. 2 Contoh *use case diagram*

2.7.2 Sequence Diagram

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

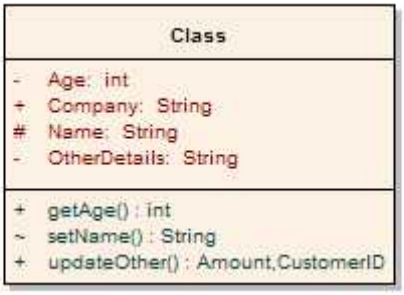


Gambar 2. 3 Contoh *sequence diagram*

2.7.3 Class Diagram

Class (Kelas) adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek. Kelas menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).

Diagram kelas menggambarkan struktur dan deskripsi kelas, paket dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.

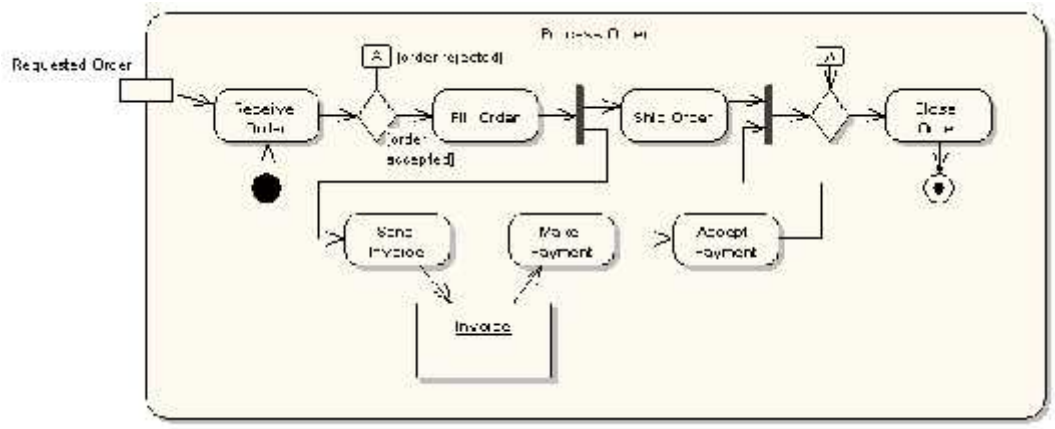


Gambar 2. 4 Contoh *class diagram*

2.7.4 Activity Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.

Activity diagram merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (internal processing). Oleh karena itu activity diagram tidak menggambarkan perlakuan internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses- proses dan jalur-jalur aktivitas dari level atas secara umum. [9]



Gambar 2. 5 Contoh *activity diagram*

2.8 Kuesioner

Kuesioner (atau sering disebut juga angket) adalah penyelidikan mengenai suatu masalah yang banyak menyangkut kepentingan umum (orang banyak), dengan cara mengedarkan fomulir daftar pertanyaan, diajukan secara tertulis kepada sejumlah objek, untuk mendapatkan jawaban (tanggapan, respon) tertulis seperlunya. Kuesioner adalah satu set pertanyaan yang berurusan dengan satu topik tunggal atau satu set topik yang saling berkaitan, yang harus dijawab oleh objek.

Menurut pertanyaan yang diajukan, kuesioner dapat dibedakan menjadi dua bagian besar yaitu:

1. Kuesioner bentuk isian

   Dalam kuesioner bentuk isian ini responden diperbolehkan memberikan jawaban secara singkat maupun dalam bentuk uraian. Jenis pertanyaan yang memberikan kebebasan bagi responden untuk menjawabnya disebut open question. Kuesioner yang berisi pertanyaan-pertanyaan dalam bentuk open question disebut kuesioner bentuk terbuka (open form questionaire). Kuesioner bentuk terbuka ini banyak memberikan kemungkinan bagi para peneliti untuk menggali unsur-unsur emosional dan afeksi, opini pribadi, dan interpretasi yang orisinil, selain itu juga bisa menggali motivasi-motivasi yang melatarbelakangi perbuatan/tingkah laku dan unsur-unsur psikis lainnya. Nilai terpenting dari kuesioner bentuk terbuka adalah sebagai alat eksploratif untuk memperkokoh satu opini, atau untuk lebih menjelaskan suatu masalah.

   Kuesioner jenis ini juga efektif sekali dalam menggali motivasi-motivasi pribadi dan informasi-informasi primer, langsung dari pribadi responden yang kualitatif sifatnya.

   Selain kuesioner bentuk terbuka ini, dalam kuesioner bentuk isian terdapat pula satu bentuk yang hanya menyediakan ruangan-ruangan khusus yang amat terbatas untuk memberikan informasinya. Pada kuesioner jenis ini responden diminta untuk mengisi pertanyaan-pertanyaan dengan jawaban yang singkat. Pertanyaan jenis ini disebut supply type question. Kuesioner jenis ini disebut *closed form questionnaire*.

2. Kuesioner bentuk pilihan

   Pada kuesioner bentuk pilihan responden diminta untuk memilih salah satu dari beberapa kemungkinan atau alternative jawaban yang disediakan. Pilihan itu ada yang berbentuk paksaan (*forced choice*), yaitu orang diminta untuk memilih salah satu dari dua alternatif, misalnya dengan jawaban „setuju‟ atau „tidak setuju‟, „ya‟ atau „tidak‟, „bisa‟ atau „tidak bisa‟, dan seterusnya. Kuesioner pilihan paksaan atau forced choice ini

disebut juga sebagai kuesioner dikotomis. Kuesioner jenis ini memang praktis, hemat, dan mudah dijawab, tetapi kurang dapat menyajikan informasi yang cukup mengenai masalah yang akan diteliti.

Jenis lain dari kuesioner bentuk pilihan adalah pilihan ganda (multiple choice), dimana responden diminta untuk memilih salah satu dari beberapa alternatif jawaban yang ada atau memilih lebih dari satu jawaban dari banyaknya kemungkinan jawaban yang disediakan.

Kuesioner jenis ini memang lebih susah untuk dibuat, karena pembuat kuesioner harus memikirkan alternatif jawaban yang mungkin dan menyusunnya dalam format yang terstruktur, tetapi waktu yang dibutuhkan menghitung dan menganalisa jawaban responden lebih singkat dibanding dengan kuesioner jenis isian. Selain itu variasi jawaban-jawaban yang diberikan responden juga lebih sedikit, karena responden hanya dapat memilih dari alternatif-alternatif yang disediakan saja.

2.8.1 Skala Likert

Skala Likert adalah suatu skala psikometrik yang umum digunakan dalam kuesioner, dan merupakan skala yang paling banyak digunakan dalam riset berupa survei. Nama skala ini diambil dari nama Rensis Likert, yang menerbitkan suatu laporan yang menjelaskan penggunaannya Sewaktu menanggapi pertanyaan dalam skala Likert, responden menentukan tingkat persetujuan mereka terhadap suatu pernyataan dengan memilih salah satu dari pilihan yang tersedia. Biasanya disediakan lima pilihan skala dengan format seperti:

1. Sangat tidak setuju
2. Tidak setuju
3. Netral
4. Setuju
5. Sangat setuju

Selain pilihan dengan lima skala seperti contoh di atas, kadang digunakan juga skala dengan tujuh atau sembilan tingkat. Suatu studi empiris menemukan

bahwa beberapa karakteristik statistik hasil kuesioner dengan berbagai jumlah pilihan tersebut ternyata sangat mirip.

Rumus yang digunakan dalam skala Likert adalah :

$$P = S/skorideal \times 100\% \quad [10]$$

Keterangan:

P = Nilai persentase yang dicari

S = jumlah frekuensi dikalikan dengan nilai yang ditetapkan jawaban

Skorideal = nilai tertinggi dikalikan dengan jumlah sampel

100% = bilangan tetap