BAB II

LANDASAN TEORI

2.1 Data Mining

Data mining adalah suatu konsep yang digunakan untuk menemukan pengetahuan yang tersembunyi di dalam database. *Data mining* merupakan proses semi otomatik yang menggunakan teknik statistik, matematika, kecerdasan buatan, dan *machine learning* untuk mengekstraksi dan mengidentifikasi informasi pengetahuan potensial dan berguna yang tersimpan di dalam database besar.

Menurut Gartner Group data mining adalah suatu proses menemukan hubungan yang berarti, pola, dan kecenderungan dengan memeriksa dalam sekumpulan besar data yang tersimpan dalam penyimpanan dengan menggunakan teknik pengenalan pola seperti teknik statistik dan matematika. Data mining didefinisikan sebagai proses menemukan pola-pola dalam data.

Karakteristik data mining sebagai berikut (Kusrini,2009) :

a. Data mining berhubungan dengan penemuan sesuatu yang tersembunyi dan pola data tertentu yang tidak diketahui sebelumnya.
b. Data mining biasa menggunakan data yang sangat besar. Biasanya data yang besar digunakan untuk membuat hasil lebih dipercaya.
c. Data mining berguna untuk membuat keputusan yang kritis, terutama dalam strategi.

2.1.1 Tahap-Tahap Data Mining

Sebagai suatu rangkaian proses, data mining dapat dibagi menjadi beberapa tahap. Tahap-tahap tersebut bersifat interaktif, pemakai terlibat langsung atau dengan perantaraan *knowledge base*. Tahap-tahap data mining ada 7 yaitu (Han, 2006) :

1. Pembersihan data *(data cleaning)*

   Pembersihan data merupakan proses menghilangkan noise dan data yang tidak konsisten atau data tidak relevan. Pada umumnya data yang diperoleh, baik dari database suatu perusahaan maupun hasil eksperimen, memiliki isian-isian yang tidak sempurna seperti data yang hilang, data yang tidak valid atau juga hanya sekedar salah ketik.Selain itu, ada juga atribut-atribut data yang tidak relevan dengan hipotesa data mining yang dimiliki. Data-data yang tidak relevan itu juga lebih baik dibuang. Pembersihan data juga akan mempengaruhi performasi dari teknik data mining karena data yang ditangani akan berkurang jumlah dan kompleksitasnya.

2. Integrasi data *(data integration)*

   Integrasi data merupakan penggabungan data dari berbagai database ke dalam satu database baru. Tidak jarang data yang diperlukan untuk data mining tidak hanya berasal dari satu database tetapi juga berasal dari beberapa database atau file teks. Integrasi data dilakukan pada atribut-aribut yang mengidentifikasikan entitas-entitas yang unik seperti atribut nama,jenis produk, nomor pelanggan dan lainnya.Integrasi data perlu dilakukan secara cermat karena kesalahan pada integrasi data bisa menghasilkan hasil yang menyimpang dan bahkan menyesatkan pengambilan aksi nantinya. Sebagai contoh bila integrasi data berdasarkan jenis

produk ternyata menggabungkan produk dari kategori yang berbeda maka akan didapatkan korelasi antar produk yang sebenarnya tidak ada.

3. Seleksi Data *(data selection)*

Data yang ada pada database sering kali tidak semuanya dipakai, oleh karena itu hanya data yang sesuai untuk dianalisis yang akan diambil dari database. Sebagai contoh, sebuah kasus yang meneliti faktor kecenderungan orang. membeli dalam kasus market basket analisis, tidak perlu mengambil nama pelanggan, cukup dengan id pelanggan saja.

4. Transformasi data *(data transformation)*

Data diubah atau digabung ke dalam format yang sesuai untuk diproses dalam data mining. Beberapa metode data mining membutuhkan format data yang khusus sebelum bisa diaplikasikan. Sebagai contoh beberapa metode standar seperti analisis asosiasi dan clustering hanya bisa menerima input data kategorikal. Karenanya data berupa angka numerik yang berlanjut perlu dibagi-bagi menjadi beberapa interval. Proses ini sering disebut transformasi data.

5. *Proses mining*

Merupakan suatu proses utama saat metode diterapkan untuk menemukan pengetahuan berharga dan tersembunyi dari data.

6. Evaluasi pola *(pattern evaluation)*

Untuk mengidentifikasi pola-pola menarik kedalam knowledge based yang ditemukan.Dalam tahap ini hasil dari teknik data mining berupa pola-pola yang khas maupun model prediksi dievaluasi untuk menilai apakah hipotesa yang ada memang tercapai. Bila ternyata hasil yang diperoleh tidak sesuai hipotesa ada beberapa alternatif yang dapat diambil seperti menjadikannya umpan balik untuk memperbaiki proses data mining,mencoba metode data mining lain yang lebih sesuai, atau menerima hasil ini sebagai suatuhasil yang di luar dugaan yang mungkin bermanfaat.

7. Presentasi pengetahuan *(knowledge presentation)*

Merupakan visualisasi dan penyajian pengetahuan mengenai metode yang digunakan untuk memperoleh pengetahuan yang diperoleh pengguna. Tahap terakhir dari proses data mining adalah bagaimana memformulasikan keputusan atau aksi dari hasil analisis yang didapat. Ada kalanya hal ini harus melibatkan orang-orang yang tidak memahami data mining. Karenanya presentasi hasil data mining dalam bentuk pengetahuan yang bisa dipahami semua orang adalah satu tahapan yang diperlukan dalam proses data mining. Dalam presentasi ini,visualisasi juga bisa membantu mengkomunikasikan hasil data mining.

2.2 Algoritma Apriori

Ide dasar dari algoritma ini adalah dengan mengembangkan frequent itemset. Dengan menggunakan satu item dan secara rekursif mengembangkan frequent itemset dengan dua item, tiga item dan seterusnya hingga frequent itemset dengan semua ukuran (Santoso, 2007).

Untuk mengembangkan *frequent set* dengan dua item, dapat menggunakan *frequent set item*. Alasannya adalah bila set satu item tidak melebihi *support minimum*, maka sembarang ukuran itemset yang lebih besar tidak akan melebihi *support minimum* tersebut. Secara umum, mengembangkan set dengan fc-item menggunakan *frequent set* dengan k – 1 item yang dikembangkan dalam langkah sebelumnya. Setiap langkah memerlukan sekali pemeriksaan ke seluruh isi database.

Dalam asosiasi terdapat istilah *antecedent* dan *consequent*, *antecedent* untuk mewakili bagian "jika" dan *consequent* untuk mewakili bagian "maka". Dalam analisis ini, *antecedent* dan *consequent* adalah sekelompok item yang tidak punya hubungan secara bersama.

Dari jumlah besar aturan yang mungkin dikembangkan, perlu memiliki aturan - aturan yang cukup kuat tingkat ketergantungan antar item dalam *antecedent* dan *consequent*. Untuk mengukur kekuatan aturan asosiasi ini, digunakan ukuran *support* dan *confidence*. *Support* adalah rasio antara jumlah transaksi yang memuat *antecedent* dan *consequent* dengan jumlah transaksi. *Confidence* adalah rasio antara jumlah transaksi yang meliputi semua item dalam *antecedent* dan *consequent* dengan jumlah transaksi yang meliputi semua item dalam *antecedent.*

Algoritma Apriori adalah salah satu algoritma yang melakukan pencarian *frequent itemset* dengan menggunakan teknik *association rule* (Erwin, 2009). Algoritma Apriori menggunakan pengetahuan frekuensi atribut yang telah diketahui sebelumnya untuk memproses informasi selanjutnya. Pada algoritma Apriori menentukan kandidat yang mungkin muncul dengan cara memperhatikan minimum *support* dan minimum *confidence*. *Support* adalah nilai pengunjung atau persentase kombinasi sebuah *item* dalam *database*. Rumus support adalah sebagai berikut :

$$Support\ (A,B) = P\ (A \cap B)$$

$$Support\ (A,B) = \frac{\sum \text{Transaksi mengandung A dan B}}{\sum \text{Total Transaksi}} X\ 100\%$$

Sedangkan *confidence* adalah nilai kepercayaan yaitu kuatnya hubungan antar item dalam sebuah *Apriori. Confidence* dapat dicari setelah pola frekuensi munculnya sebuah item ditemukan. Berikut rumus *confidence* :

$$Confidence = P(B \mid A) = \frac{\sum Transaksi\ mengandung\ A\ dan\ B}{Transaksi\ Mengadung\ A} \text{x } 100\%$$

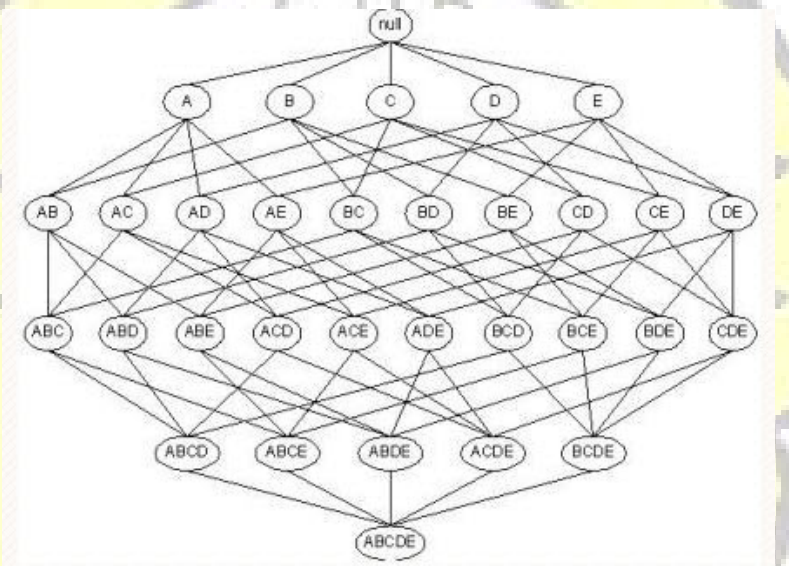
secara garis besar cara kerja algoritma apriori adalah:

1. Pembentukan kandidat itemset, Kandidat k-itemset dibentuk dari kombinasi (k-1)-itemset yang didapat dari iterasi sebelumnya. Satu ciri dari algoritma Apriori adalah adanya pemangkasan kandidat k-itemset yang subset-nya yang berisi k-1 item tidak termasuk dalam pola frekuensi tinggi dengan panjang k-1.
2. Penghitungan support dari tiap kandidat k-itemset. Support dari tiap kandidat k-itemset didapat dengan men-scan database untuk menghitung jumlah transaksi yang memuat semua item di dalam kandidat k-itemset tsb. Ini adalah juga ciri dari algoritme Apriori dimana diperlukan penghitungan dengan scan seluruh database sebanyak k-itemset terpanjang.
3. Tetapkan pola frekuensi tinggi. Pola frekuensi tinggi yang memuat k item atau k-itemset ditetapkan dari kandidat k-itemset yang support-nya lebih besar dari minimum support.
4. Bila tidak didapat pola frekuensi tinggi baru maka seluruh proses dihentikan. Bila tidak, maka k ditambah satu dan kembali ke bagian 1.

Kelebihan algoritma apriori :

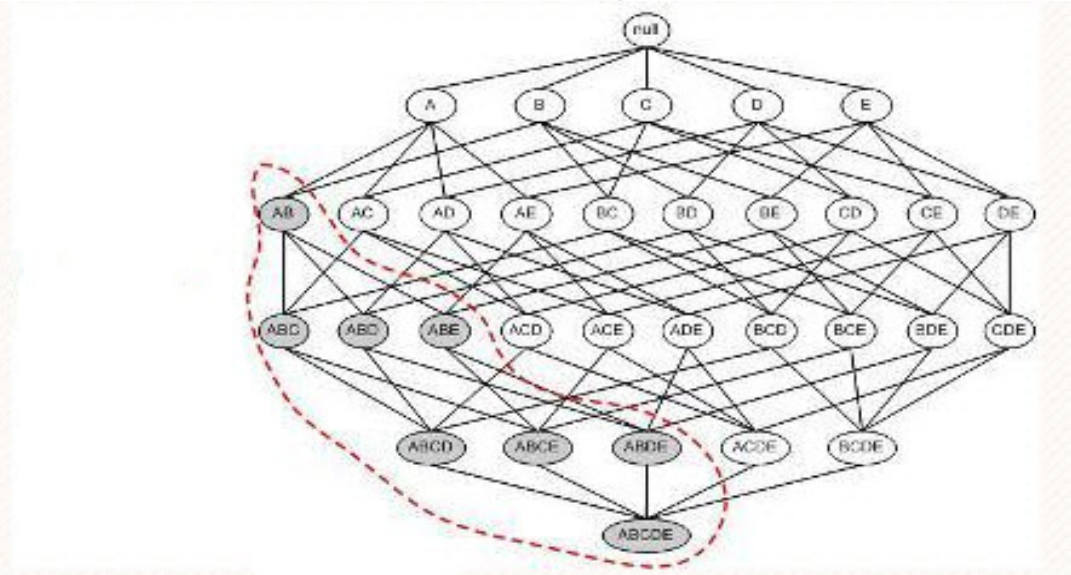
a. Kelebihan dari algoritma apriori ini adalah lebih sederhana dan dapat menangani data yang besar. Sedangkan algoritma lainnya memiliki kelemahan dalam penggunaan memori saat jumlah data besar, tentunya berpengaruh terhadap banyaknya item yang diproses serta mudah di pahami struktur kerja dan implementasinya.

Masalah utama pencarian Frequent Itemset adalah banyaknya jumlah kombinasi itemset yang harus diperiksa apakah memenuhi minimum support atau tidak. Salah satu cara untuk mengatasinya adalah dengan mengurangi jumlah kandidat itemset yang harus diperiksa. Apriori adalah salah satu pendekatan yang sering digunakan pada Frequent Itemset Mining. Prinsip Apriori adalah jika sebuah itemset infrequent, maka itemset yang infrequent tidak perlu lagi diexplore supersetnya sehingga jumlah kandidat yang harus diperiksa menjadi berkurang. Kira-kira ilustrasinya seperti ini :



Gambar 2.1 Contoh Hash Tree (Fajar Astuti Herawati, 2013)

Pada gambar di atas, pencarian Frequent Itemset dilakukan tanpa menggunakan prinsip Apriori. Dengan menggunakan prinsip Apriori, pencarian Frequent Itemset akan menjadi seperti di bawah ini:



Gambar 2.2 Contoh Hash Tree (Fajar Astuti Herawati, 2013)

Dapat dilihat bahwa dengan menggunakan Apriori, jumlah kandidat yang harus diperiksa cukup banyak berkurang. Apriori sendiri terus dikembangkan untuk meningkatkan efisiensi dan efektivitasnya. Salah satunya adalah dengan memanfaatkan Hash Tree untuk perhitungan support yang efisien (mengurangi Database scan yang berulang-ulang).

2.3 Algoritma Genetika

Untuk membandingkan antara algoritma apriori dengan algoritma lain, penulis menambahkan algoritma genetika sebagai bahan perbandingan. Untuk pembahasan lebih lanjut, berikut penjelasannya.

Algoritma ini ditemukan di Universitas Michigan, Amerika Serikat oleh John Holland (1975) melalui sebuah penelitian dan dipopulerkan oleh salah satu muridnya, David Goldberg (1989). Dimana mendefenisikan algoritma genetika ini sebagai metode algoritma pencarian berdasarkan pada mekanisme seleksi alam dan genetik alam.

Algoritma genetika adalah algoritma yang berusaha menerapkan pemahaman mengenai evolusi alamiah pada tugas-tugas pemecahan-masalah (*problem solving*). Pendekatan yang diambil oleh algoritma ini adalah dengan menggabungkan secara acak berbagai pilihan solusi terbaik di dalam suatu kumpulan untuk mendapatkan generasi solusi terbaik berikutnya yaitu pada suatu kondisi yang memaksimalkan kecocokannya atau lazim disebut *fitness*. Generasi ini akan merepresentasikan perbaikan-perbaikan pada populasi awalnya. Dengan melakukan proses ini secara berulang, algoritma ini diharapkan dapat mensimulasikan proses evolusioner.

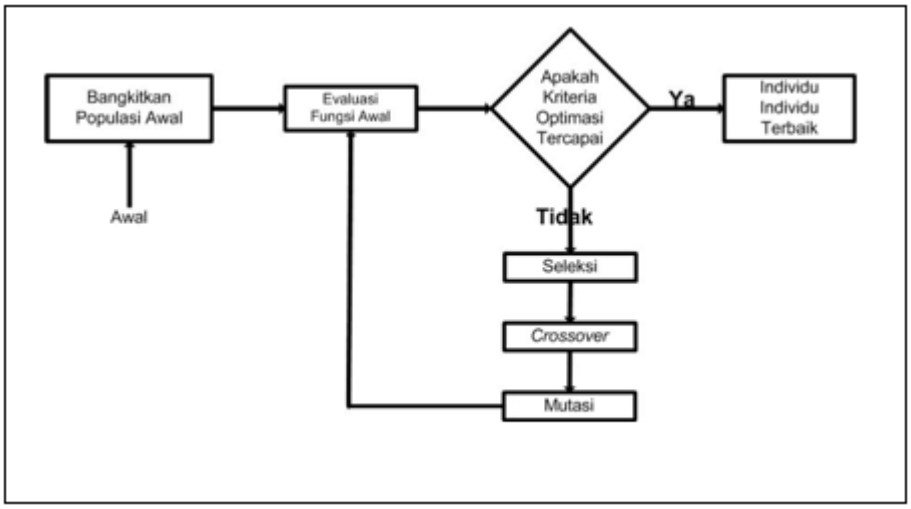
Pada akhirnya, akan didapatkan solusi-solusi yang paling tepat bagi permasalahan yang dihadapi. Untuk menggunakan algoritma genetik, solusi permasalahan direpresentasikan sebagai kromosom. Tiga aspek yang penting untuk enggunaan algoritma genetik:

1. Defenisi fungsi *fitness*
2. Defenisi dan implementasi representasi genetic
3. Defenisi dan implementasi operasi genetic

Jika ketiga aspek di atas telah didefinisikan, algoritma genetika akan bekerja dengan baik. Tentu saja, algoritma genetika bukanlah solusi terbaik untuk memecahkan segala masalah. Sebagai contoh, metode tradisional telah diatur untuk untuk mencari penyelesaian dari fungsi analitis *convex* yang "berperilaku baik" yang variabelnya sedikit. Pada kasus-kasus ini, metode berbasis kalkulus lebih unggul dari algoritma genetika karena metode ini dengan cepat menemukan solusi minimum ketika algoritma genetika masih menganalisa bobot dari populasi awal.

2.3.1 Struktur Umum Algoritma Genetika

Algoritma genetika memberikan suatu pilihan bagi penentuan nilai parameter dengan meniru cara reproduksi genetika, pembentukan kromosom baru serta seleksi alami seperti yang terjadi pada makhluk hidup. Algoritma Genetika secara umum dapat diilustrasikan dalam diagram alir berikut ini:



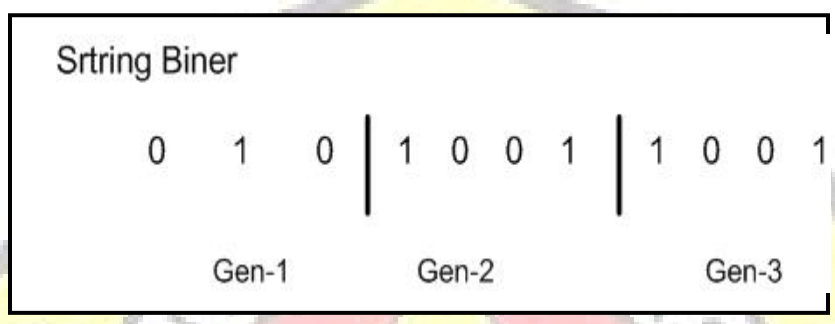
Gambar 2.3 Diagram Alir Algoritma Genetika (Kusumadewi, 2003)

(Kusumadewi, 2003) Pada algoritma ini, teknik pencarian dilakukan sekaligus atas sejumlah solusi yang mungkin dikenal dengan istilah populasi. Individu yang terdapat dalam satu populasi individu yang terdapat dalam satu populasi disebut dengan istilah **kromosom**, Charles L Karr (1999). Kromosom ini merupakan suatu solusi yang masih berbentuk simbol. Populasi awal dibangun secara acak, sedangkan populasinya merupakan hasil evolusi kromosom-kromosom melalui iterasi yang disebut dengan istilah **generasi**. Pada setiap generasi kromosom akan melalui proses evaluasi dengan menggunakan alat ukur yang disebut **fungsi fitness**. Nilai fitness dari suatu kromosom akan menunjukkan kualitas kromosom dalam populasi tersebut.

2.3.2 Penyandian

Teknik penyandian disini meliputi penyandian gen dari kromosom. Gen merupakan bagian dari kromosom. Satu gen biasanya akan mewakili satu variable.

Gen dapat direpresentasikan dalam bentuk : *string bit, pohon, array* bilangan real, daftar aturan, elemen permutasi, elemen program, atau representasi lainnya yang dapat diimplementasikan untuk operator genetika.



Gambar 2.4 Penyandian Biner pada Operator Genetika

(Kusumadewi, 2003)

Demikian juga, kromosom dapat direpresentasikan dengan menggunakan :

- String bit : 011, 01101, 11101, dst.
- Bilangan Real : 65.65, -67.98. 562.88, dst.
- Elemen Program : pemrograman genetika
- Struktur lainnya

2.3.3 Operator Genetika

Algoritma genetik merupakan proses pencarian yang heuristik dan acak sehingga penekanan pemilihan operator yang digunakan sangat menentukan keberhasilan algoritma genetik dalam menemukan solusi optimum suatu masalah yang diberikan. Hal yang harus diperhatikan adalah menghindari terjadinya konvergensi *premature*, yaitu mencapai solusi optimum yang belum waktunya, dalam arti bahwa solusi yang diperoleh adalah hasil optimum lokal.

Operator genetika yang digunakan setelah proses evaluasi tahap pertama membentuk populasi baru dari generasi sekarang. Operator-operator tersebut adalah operator seleksi, *crossover* dan mutasi. (Kusumadewi, 2003). Berikut ini akan di jelaskan masing-masing operator pada Genetika.

1. Seleksi

Seleksi bertujuan memberikan kesempatan reproduksi yang lebih besar bagi anggota populasi yang paling *fit*. Langkah pertama dalam seleksi ini adalah pencarian nilai *fitness*. Masing-masing individu dalam suatu wadah seleksi akan menerima probabilitas reproduksi yang tergantung pada nilai objektif dirinya sendiri terhadap nilai objektif dari semua individu dalam wadah seleksi tersebut. Nilai *fitness* inilah yang nantinya akan digunakan pada tahap seleksi berikutnya (Kusumadewi, 2003).

Kemampuan algoritma genetik untuk memproduksi kromosom yang lebih baik secara progresif tergantung pada penekanan selektif (*selective pressure*) yang diterapkan ke populasi. Penekanan selektif dapat diterapkan dalam dua

cara. Cara pertama adalah membuat lebih banyak kromosom anak yang dipelihara dalam populasi dan memilih hanya kromosom-kromosom terbaik bagi generasi berikut. Walaupun orang tua dipilih secara acak, metode ini akan terus menghasilkan kromosom yang lebih baik berhubungan dengan penekanan selektif yang diterapkan pada individu anak tersebut.

Cara lain menerapkan penekanan selektif adalah memilih orang tua yang lebih baik ketika membuat keturunan baru. Dengan metode ini, hanya kromosom sebanyak yang dipelihara dalam populasi yang perlu dibuat bagi generasi berikutnya. Walaupun penekanan selektif tidak diterapkan ke level keturunan, metode ini akan terus menghasilkan kromosom yang lebih baik, karena adanya penekanan selektif yang diterapkan ke orangtua.

Ada beberapa metode untuk memilih kromosom yang sering digunakan antara lain adalah seleksi roda rolet (*roulette wheel selection*), seleksi ranking (*rank selection*) dam seleksi turnamen (*tournament selection*).

2. Crossover

Crossover (perkawinan silang) bertujuan menambah keanekaragaman string dalam populasi dengan penyilangan antar-string yang diperoleh dari sebelumnya. Beberapa jenis *crossover* tersebut adalah:

1. *Crossover* 1-titik

   Pada *crossover* dilakukan dengan memisahkan suatu string menjadi dua bagian dan selanjutnya salah satu bagian dipertukarkan dengan salah satu bagian dari string yang lain yang telah dipisahkan dengan cara yang sama. Proses yang

demikian dinamakan operator *crossover* satu titik seperti diperlihatkan pada gambar berikut:

Tabel 2.1 Contoh Crossover 1 titik

Kromosom Orangtua 1	**11001**011
Kromosom Orangtua 2	11011**111**
Keturunan	**11001111**

2. Crossover 2 Titik

Proses *crossover* ini dilakukan dengan memilih dua titik *crossover*. Kromosom keturunan kemudian dibentuk dengan barisan bit dari awal kromosom sampai titik *crossover* pertama disalin dari orang tua pertama, bagian dari titik *crossover* pertama dan kedua disalin dari orang tua kedua, kemudian selebihnya disalin dari orang tua pertama lagi.

Tabel 2.2 Contoh Crossover 2 titik

Kromosom Orangtua 1	**11**0010**11**
Kromosom Orangtua 2	11**011**11
Keturunan	**11011111**

3. Crossover Seragam

Crossover seragam manghasilkan kromosom keturunan dengan menyalin bit-bit secara acak dari kedua orangtuanya.

Table 2.3 Contoh Crossover Seragam

Kromosom Orangtua 1	**110**010**11**
Kromosom Orangtua 2	110**111**11
Keturunan	**11011111**

2.3.4 Mutasi

Mutasi merupakan proses mengubah nilai dari satu atau beberapa gen dalam suatu kromosom. Operasi *crossover* yang dilakukan pada kromosom dengan tujuan untuk memperoleh kromosom-kromosom baru sebagai kandidat solusi pada generasi mendatang dengan *fitness* yang lebih baik, dan lama-kelamaan menuju solusi optimum yang diinginkan. Akan tetapi, untuk mencapai hal ini, penekanan selektif juga memegang peranan yang penting. Jika dalam proses pemilihan kromosom-kromosom cenderung pada kromosom yang memiliki *fitness* yang tinggi saja, konvergensi *premature*, yaitu mencapai solusi yang optimal lokal sangat mudah terjadi.

Mutasi ini berperan untuk menggantikan gen yang hilang dari populasi akibat proses seleksi yang memungkinkan munculnya kembali gen yang tidak muncul pada inisialisasi populasi.

- Mutasi bilangan real

  Pada mutasi bilangan real, ukuran langkah mutasi biasanya sangat sulit ditentukan. Ukuran yang kecil biasanya sering mengalami kesuksesan, namun adakalanya ukuran yang lebih besar akan berjalan lebih cepat.

- Mutasi bilangan real

  Cara sederhana untuk mendapatkan mutasi biner adalah dengan mengganti satu atau beberapa nilai gen dari kromosom.

2.3.5 Parameter Genetika

algoritma genetik dibutuhkan 4 parameter (Juniawati, 2003) yaitu :

1. Probabilitas Persilangan (*Crossover Probability*)

   Menunjukkan kemungkinan *crossover* terjadi antara 2 kromosom. Jika tidak terjadi *crossover* maka keturunannya akan sama persis dengan kromosom orangtua, tetapi tidak berarti generasi yang baru akan sama persis dengan generasi yang lama. Jika probabilitas *crossover* 100% maka semua keturunannya dihasilkan dari *crossover*. *Crossover* dilakukan dengan harapan bahwa kromosom yang baru akan lebih baik.

2. Probabilitas Mutasi (*Mutation Probability*)

   Menunjukkan kemungkinan mutasi terjadi pada gen-gen yang menyusun sebuah kromosom. Jika tidak terjadi mutasi maka keturunan yang dihasilkan setelah *crossover* tidak berubah. Jika terjadi mutasi bagian kromosom akan berubah. Jika

probabilitas 100%, semua kromosom dimutasi. Jika probabilitasnya 0%, tidak ada yang mengalami mutasi.

3. Jumlah Individu

   Menunjukkan jumlah kromosom yang terdapat dalam populasi (dalam satu generasi). Jika hanya sedikit kromosom dalam populasi maka algoritma genetik akan mempunyai sedikit variasi kemungkinan untuk melakukan *crossover* antara orangtua karena hanya sebagian kecil dari *search space* yang dipakai. Sebaliknya jika terlalu banyak maka algoritma genetik akan berjalan lambat.

4. Jumlah Populasi

   Menetukan jumlah populasi atau banyaknya generasi yang dihasilkan, digunakan sebagai batas akhir proses seleksi, persilangan dan mutasi.

Setelah didapatkan kesimpulan dari sedikit pembahasan mengenai algoritma genetika, penulis lebih memilih menggunakan algoritma apriori untuk di terapkan pada aplikasi, karena selain lebih mudah untuk dipahami, mudah juga untuk diimplementasikan ke dalam program.

2.4 Internet

Menurut (Iskandar, 2009) dalam buku Panduan Lengkap Internet, *Internet* atau *interconnected network* adalah sebuah sistem komunikasi *global* yang menghubungkan komputer-komputer dan jaringan-jaringan komputer di seluruh dunia. Setiap komputer dan jaringan terhubung secara langsung maupun tidak langsung ke beberapa jalur utama yang disebut *internet backbone.* Masing-masing

dibedakan antara satu dengan yang lainnya menggunakan *unique name* yang disebut alamat *IP* 32 bit.

1. Komputer dan jaringan dengan berbagai platform (*Unix*, *Linux*, *Windows*, *Mac* dan lain-lain) dapat bertukar informasi dengan adanya sebuah protokol standar yang dikenal dengan nama *TCP/IP* (*Transmission Control Protocol/Internet Protocol*). *TCP/IP* tersusun atas 4 layer, yaitu *network access*, *internet*, *host-to-host transport*, dan *application*.
2. *Internet* merupakan sekumpulan jaringan komputer yang menghubungkan situs akademik, pemerintahan, komersial, organisasi, maupun perorangan. *Internet* menyediakan akses untuk layanan telekomunikasi dan sumber daya informasi untuk jutaan pemakainya yang tersebar di seluruh dunia. Layanan *internet* meliputi komunikasi langsung (*email,chat*), diskusi (*Usenet news*, *email*, *milis*), sumber daya informasi yang terdistribusi (*World Wide web*, *Gopher*), *remote login*, lalu lintas *file* (*Telnet*,*FTP*) dan aneka layanan lainnya.

2.5 Definisi Website

Menurut (Wikipedia bahasa indonesia), *website* / situs *web* atau sering disingkat dengan istilah situs adalah sejumlah halaman *web* yang memiliki topik saling terkait, terkadang disertai pula dengan berkas-berkas gambar, video, atau jenis-jenis berkas lainnya. Sebuah situs *web* biasanya ditempatkan setidaknya pada sebuah *serverweb* yang dapat diakses melalui jaringan seperti *internet*, ataupun jaringan wilayah lokal (*LAN*) melalui alamat *internet* yang dikenali sebagai *URL*. Gabungan atas semua situs yang dapat diakses publik di *internet* disebut pula sebagai *World*

Wide Web atau lebih dikenal dengan singkatan *WWW*. Meskipun setidaknya halaman beranda situs *internet* umumnya dapat diakses publik secara bebas, pada prakteknya tidak semua situs memberikan kebebasan bagi publik untuk mengaksesnya, beberapa situs *web* mewajibkan pengunjung untuk melakukan pendaftaran sebagai anggota, atau bahkan meminta pembayaran untuk dapat menjadi aggota untuk dapat mengakses isi yang terdapat dalam situs *web* tersebut, misalnya situs-situs yang menampilkan pornografi, situs-situs berita, layanan surel (*e-mail*), dan lain-lain. Pembatasan-pembatasan ini umumnya dilakukan karena alasan keamanan, menghormati privasi, atau karena tujuan komersil tertentu.

Perlu, diketahui struktur desain website dibagi menjadi :

a. *Headerweb* adalah bagian judul dari *website*, biasanya berisi alamat *website* atau judul yang menerangkan secara jelas mengenai tema *website* yang dibangun.
b. *Navigasi* adalah bagian menu utama yang posisinya bisa *vertical* atau *horizontal.* Pada bagian ini, pemilik *web* juga dapat meletakan tools.
c. *Content* adalah bagian utama dari *website* yang digunakan untuk menampilkan semua informasi yang dihasilkan dari *link* menu dan tombol.
d. *Link* adalah bagian dari *website* yang berfungsi untuk berpindah halaman atau *website* yang telah ditentukan.

Footer web adalah bagian paling bawah *website*. Pada bagian ini, biasanya berisi informasi pemilik *web* atau hal penting yang terkait dengan pengembang *web*.

2.5.1 Script Pemrograman Web

2.5.1.1 HTML

HTML *(Hypertext Markup Laguange)* adalah bahasa dasar untuk *web scripting* bersifat *client side* yang memungkinkan untukmenampilkan informasi dalam bentuk teks, graafik, serta multimedia dan juga untuk menghubungkan antar tampilan *web page(hyperlink).(Bernard Renaidy Suteja,S.Kom, M.Kom : 2010)*

```
<HTML>
<HEAD>
<TITLE>Judul Web Page</TITLE>
<BODY>
        Isi Documen ada Disini
</BODY>
</HTML>
```

Gambar 2.5 Contoh Koding HTML (Sumber : Bernard Reinady Sutedja, 2010)

2.5.1.2 PHP

Bahasa pemrograman PHP merupakan bahasa pemrograman untuk membuat web yang bersifat server-side scripting. PHP memungkinkan kita untuk membuat halaman web yang bersifat dinamis. PHP dapat dijalankan pada berbagai macam Operating System (OS), misalnya Windows, Linux dan Mac OS. Selain Apache, PHP juga mendukung beberapa web server lain, misalnya Microsoft IIS, Caudium, PWS

dan lain-lain. Sistem manajemen database yang sering digunakan bersama PHP adalah MySQL. Namun PHP juga mendukung system manajemen Database Oracle, Microsoft Acces, Interbase, d-Base, PostgreSQL dan sebagainya

PHP mendukung penuh Object Oriented Programing (OOP), integrasi XML, mendukung semua ekstensi terbaru MySQL, pengembangan web services dengan SOAP dan REST, serta ratusan kemampuan. Sama dengan web server lainnya PHP juga bersifat open source sehingga setiap orang dapat menggunakannya dengan gratis..(Lukmanul Hakim, 2005)

```
<?php
echo "This is a HEADER, And          ";
echo "This is a HEADER, And          ";
echo "Today is ";
echo date("F d");
echo ", ";
echo date("Y");?>
```

Gambar 2.6 Contoh Koding PHP (Sumber : Lukmanul Hakim, 2005)

2.5.1.3 JAVASCRIPT

JavaScript adalah suatu bahasa script yang di-*interpreter* oleh browser *(client side).* Pada awalnya bahasa ini dinamakan *"LiveScript"* yang berfungsi sebagai bahasa sederhana untuk *browser Netcape Navigator2*. Sintak penulis Javascript memiliki kemiripan dengan bahasa pemrograman Java dan juga C sehingga banyak aturan dari bahasa Java atau C yang bisa diterapkan dalam JavaScript, tetapi perlu diingat JavaScript tidak sama dengan Java, karena JavaScript yang dikembangkan oleh Netscape, produknya untuk web disebut *web script.*

JavaScript bergantung pada *browser(Navigator)* yang memanggil halaman *web* yang berisi script dari JavaScript dan tentu saja tersisipkan didalam dokumen HTML. JavaSkript juga tidak memerlukan *compiler* atau penerjemah khusus untuk menjalankannya (karena pada kenyataannya compiler JavaScript sendiri sudah termasuk didalam *browser tersebut).* Penulisan kode JavaScript diletakkan diantara tag HTML. Dengan menambahkan JavaScript akan membuat halaman *web* menjadi lebih menarik dan interaktif. Menggunakan JavaScript, memungkinkan kustomisasi terhadap dokumen HTML pada saat diakses dengan menulis melalui penanganan *event* terhadap elemen-elemen tag HTML dalam halaman tersebut, memeriksa data *Jorm* pada sisi client dan melakukan perhitungan pada sisi *client* hingga membuat animasi kursor mouse dan sebagainya. (Komang Wiswakarma,2009)

```
<BODY>
<script language=”javascript”>
        //Baris kode Javaskrip diketikkan disini
      //Setiap perintah diakhiri tanda;
</script></BODY>
```

Gambar 2.7 Contoh koding body javascript dalam dokumen HTML. (Sumber : Komang Wiswakarma,2005)

2.5.1.4 CSS

CSS adalah singkatan dari Cascading Style Sheet, digunakan untuk mengatur style atau tampilan dari dokumen HTML. CSS adalah suatu bahasa stylesheet yang digunakan untuk mengatur tampilan suatu website, baik tata letak, jenis huruf, warna, dan semua yang berhubungan dengan tampilan atau gaya suatu web (Menurut Wikipedia 2009). Disamping itu, desain web yang dibuat dengan CSS lebih cepat loadingnya dibandingkan dengan desain menggunakan tabel dari HTML, bukankah selain konten, kecepatan akses merupakan faktor penting dalam dunia maya. Dengan berbagai keunggulan tersebut, CSS menjadi salah satu bahasa pemrograman yang paling disarankan dalam pembuatan website.(Komang Wiswakarma, 2009)

```
<style type="text/css" media="screen">
```

Gambar 2.8 Contoh koding CSS (Sumber : Komang Wiswakarma, 2009)

2.6 Database

Menurut Bambang Hariyanto (2008), data adalah rekaman mengenai fenomena/fakta yang ada atau yang terjadi. Data pada pokonya adalah refleksi fakta yang ada. Data mengenai fakta-fakta penting organisasi harus dikelola secara baik sehingga dapat dipakai/diakses secara efisien sehingga efektif mendukung operasi dan pengendalian organisasi. Data merupakan sumber daya penting pada manajemen modern. Untuk itu, organisasi perlu melakukan penataan dan manajemen data yang baik agar data yang dimiliki organisasi dapat berdaya guna secara maksimal.

Database adalah kumpulan data yang saling berhubungan yang merefleksikan fakta-fakta yang terdapat di organisasi. Saat satu kejadian muncul si dunia nyata mengubah *state* organisasi/perusahaan/sistem maka satu perubahan pun harus dilakukan terhadap data yang disimpan di *database*. *Database* merupakan komponen utama sistem informasi karena semua informasi untuk pengambilan keputusan berasal dari data di *database*. Pengelolaaan *database* yang buruk dapat mengakibatkan ketidaktersediaan data penting yang digunakan untuk menghasilkan informasi yang diperlukan dalam pengambilan keputusan.

Sistem manajemen basisdata atau DBMS (*Database Management System*) adalah perangkat lunak untuk mendefinisikan, menciptakan, mengelola, dan mengendalikan pengaksesan basis data. Fungsi sistem manajemen basis data saat ini yang penting adalah menyediakan basis untuk sistem informasi manjemen.

2.6.1 Fungsi Database

Fungsi database umumnya memang banyak diterapkan dalam dunia industri, hampir seluruh industri di belahan dunia memanfaatkan teknologi database untuk menunjang sistem dan aplikasinya. Berikut adalah beberapa fungsi yang melekat pada sebuah database (Betha Sidik,2005):

a. Mengelompokkan data, database bertujuan untuk mengelompokkan data agar mudah dipahami. Contoh dalam sebuah system perpustakaan, ada kelompok data buku, penerbit, transaksi peminjaman, dan mahasiswa.
b. Menghindari terjadinya duplikasi atau inkonsistensi data.
c. Memudahkan dalam menyimpan, mengakses, dan memperbarui, serta menghapus data.
d. Menjamin kualitas data dan informasi yang diakses sesuai dengan yang dimasukkan (integritas data).
e. Menjadi solusi dalam proses penyimpanan sebuah data, terutama data yang besar.
f. Menunjang kinerja aplikasi yang membutuhkan sebuah penyimpanan data.

2.6.2 MySQL

Menurut (Betha Sidik, 2005) dalam buku MySQL, *MySQL* merupakan *software sistem manajemen database* (*Database Management System - DBMS*) yang sangat populer di kalangan pemrograman *web*, terutama di lingkungan *Linux* dengan menggunakan script *PHP* dan *Perl*. *Softwaredatabase* ini kini telah tersedia juga pada *platform* sistem operasi *Windows* (*98/ME* atau pun *NT/2000/XP*).

MySQL merupakan *database* yang paling populer digunakan untuk membangun aplikasi *web* yang menggunakan *database* sebagai sumber dan pengelola datanya.

Kepopuleran *MySQL* dimungkinkan karena kemudahannya untuk digunakan, cepat secara kerja *query*, dan mencukupi untuk kebutuhan *database* perusahaan-perusahaan skala menengah kecil. *MySQL* merupakan *database* yang digunakan oleh situs-situs terkemuka di *internet* untuk menyimpan datanya.

Software database MySQL kini dilepas sebagai *software* manajemen *database* yang *open source*, sebelumnya merupakan *softwaredatabase* yang *shareware*. *Shareware* adalah suatu *software* yang dapat didistribusikan secara bebas untuk keperluan penggunaan secara pribadi, tetapi jika digunakan secara komersial maka pemakai harus mempunyai lisensi dari pembuatnya. *Software open source* menjadikan *software* dapat didistribusikan secara bebas dan dapat dipergunakan untuk keperluan pribadi atau pun komersial, termasuk di dalamnya *source code* dari *software* tersebut.

Database MySQL tersedia secara bebas cuma-cuma dan boleh digunakan oleh setiap orang, dengan lisensi *open source GNUGeneral Public License* (*GPL*) atau pun lisensi komersial *non GPL*. Saat ini diperkirakan lebih dari 3 juta pemakai di seluruh dunia, dengan lebih dari setengah juta *server* yang memasangnya, termasuk di dalamnya *Yahoo!, MP3.com, Motorola, NASA, Silicon Graphics, HP, Xerox, Cisco*, dan *Texas Instruments*.

Database MySQL, merupakan *database* yang menjanjikan sebagai alternatif pilihan *database* yang dapat digunakan untuk sistem *database* personal atau organisasi kita. *Oracle* sebagai *database* besar telah membuat *kit* (modul) untuk

memudahkan proses migrasi dari *MySQL* ke dalam *Oracle*, hal ini dapat menunjukkan bahwa *Oracle* telah memperhitungkan *database MySQL* sebagai *database* alternatif masa depan. Demikian juga dengan pengguna dari *database MySQL*, menunjukkan makin banyaknya perusahaan besar menggunakannya.(Betha Sidik, 2005)

2.7 UML

Menurut(Munawar, 2005) dalam buku Pemodelan Visual Dengan UML, *UML* (*Unified Modeling Language)* adalah salah satu alat bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek. Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembang sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (sharing) dan mengkomunikasikan rancangan mereka dengan yang lain.

UML merupakan kesatuan dari bahasa pemodelan yang dikembangkan oleh Booch, *Object Modeling Technique (OMT)* dan Object Oriented Software Engineering (OOSE). Metode Booch dari Grandy Booch sangat terkenal dengan nama metode Design Object Oriented. Metode ini menjadikan proses analisis dan design kedalam empat tahapan iteratif, yaitu : identifikasi kelas-kelas dan obyek-obyek, identifikasi semantik dari hubungan obyek dan kelas tersebut, perincian interface dan implementasi. Keunggulan metode Booch adalah pada detail dan karyanya dengan notasi dan elemen. Pemodelan OMT yang dikembangkan oleh Rumbaugh didasarkan pada analisis terstruktur dan pemodelan entity-relationship.

Tahapan utama dalam metodologi ini adalah analisis, design sistem, design obyek dan implementasi. Keunggulan metode ini adalah dalam penotasian yang mendukung semua konsep OO. Metode OOSE dari Jacobson lebih memberi penekanan pada use case. OOSE memiliki tiga tahapan yaitu membuat model requirement dan analisis, design dan implementasi, dan model pengujian (test model). Keunggulan metode ini adalah mudah dipelajari karena memiliki notasi yang sederhana namun mencakup seluruh tahapan dalam rekayasa perangkat lunak. Dengan UML akan bisa menceritakan apa yang seharusnya dilakukan oleh sebuah sistem bukan bagaimana yang seharusnya dilakukan oleh sebuah sistem.(Munawar, 2005)

Karena tergolong bahasa visual, UML lebih mengedepankan penggunaan diagram untuk menggambarkan aspek dari sistem yang sedang dimodelkan. Memahami UML itu sebagai bahasa visual itu penting, karena penekanan tersebut membedakannya dengan bahasa pemrograman yang lebih dekat ke mesin. Bahasa visual lebih dekat ke mental model pikiran kita, sehingga pemodelan menggunakan bahasa visual bisa lebih mudah dan lebih cepat dipahami dibandingkan apabila dituliskan dalam sebuah bahasa pemrograman.

UML adalah salah satu bentuk notasi atau bahasa yang sama yang digunakan oleh professional dibidang software untuk menggambarkan atau memodelkan sebuah system software. Sebelumnya ada banyak notasi atau bahasa lain untuk mencapai keperluan yang sama misalnya DFD (Data Flow Diagram). Tetapi sejak matang dan populernya teknologi pemrograman, perancangan, dan analisis berorientasi objek, UML telah menjadi de facto standard language.

Ada tiga cara dalam memakai UML dalam melakukan pemodelan system:

1. UML sebagai sketsa

   UML digambarkan dalam sketsa coretan-coretan dalam kertas atau white board secara tidak formal. Biasanya digunakan dalam sesi diskusi tim untuk membahas aspek tertentu dalam tahap analisis dan perancangan.

2. UML sebagai blueprint system

   Seperti diagram kelistrikan adalah blueprint dari komponen atau produk yang akan dihasilkan, UML juga bisa menggambarkan blueprint yang identik untuk sebuah system software.

3. UML sebagai bahasa pemrograman

   UML berfungsi sebagai bahasa pemrograman mencoba melakukan semuanya dengan UML sampai kepada produk jadinya. Analisis dan perancangan dilakukan dengan diagram-diagram yang ada dalam UML, sementara sebuah tool atau generator bisa menghasilkan produk akhir dari diagram-diagram ini. Diagram-diagram yang terdapat dalam UML antara lain:

   a. *use case diagram*
   b. *class diagram*
   c. *statechart diagram*
   d. *activity diagram*
   e. *sequence diagram*
   f. *collaboration diagram*
   g. *component diagram*
   h. *deployment diagram* (Munawar, 2005).

2.7.1 Use Case

Menurut(Prabowo Pudjo Widodo, 2011) dalam buku Menggunakan UML, diagram *use case* bersama dengan narasi *use case* dan skenario mendefinisikan tujuan suatu sistem atau pengklasifikasi lain seperti *enterprise*, sub sistem atau komponen. Konsep ini diperkenalkan oleh Ivar Jacobson bersama organisasinya dalam bentuk metodologi yang mereka namakan *Object-Oriented Software Engineering* (*OOSE*). Tujuan dibentuknya metode ini adalah agar dihasilkan fokus yang baik pada pengembangannya dan tujuan utama tanpa terpengaruh oleh implementasi praktis.

Dalam pembicaraan tentang *use case*, pengguna biasanya disebut dengan actor. Actor adalah sebuah peran yang bisa dimainkan oleh pengguna dalam interaksinya dengan system.

Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.

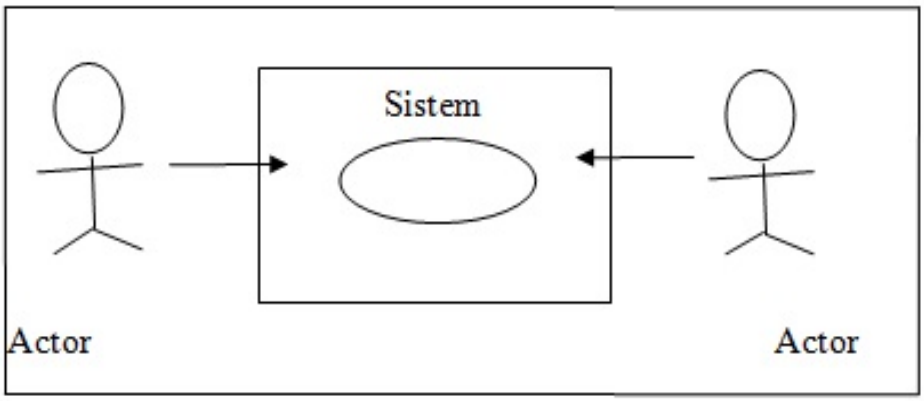
Use case diagram dapat sangat membantu bila kita sedang menyusun *requirement* sebuah sistem, mengkomunikasikan rancangan dengan klien, dan merancang *test case* untuk semua *feature* yang ada pada sistem.

Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.

Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.

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

Diagram use case menunjukan 3 aspek dari system yaitu Actor, use case dan system/sub system boundary. Actor mewakili peran orang, system yang lain atau alat ketika berkomunikasi dengan use case.

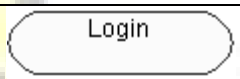


Gambar 2.9 *Use Case Model* (Munawar, 2005)

2.7.2 *Activity* Diagram

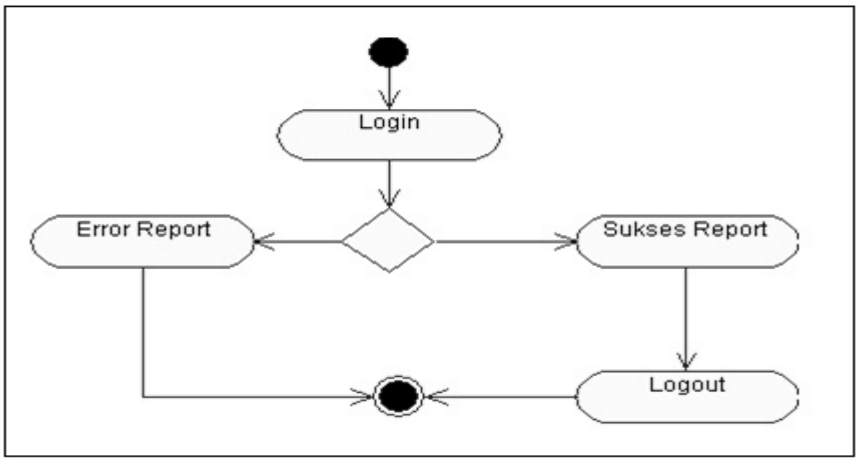
Menurut (Romi Satria Wahono, 2003)Pada dasarnya Diagram aktivitas adalah Diagram *flowchart* yang diperluas yang menunjukkan aliran kendali satu aktivitas ke aktivitas lain. Kegunaan diagram ini adalah untuk memodelkan workflow atau jalur kerja, memodelkan operasi, bagaimana objek-objek bekerja, aksi-aksi dan pengaruh terhadap objek. Simbol-simbol yang terdapat dalam *Activity* Diagram, diantaranya sebagai berikut :

Tabel 2.4 Simbol Activity Diagram (Romi Satrio Wahono, 2003)

Keterangan	**Simbol**
Titik Awal atau permulaan.	
Titik Akhir atau akhir dari aktivitas.	
Aktiviti, atau aktivitas yang dilakukan oleh aktor.	
Decision, atau pilihan untuk mengambil keputusan.	
Arah tanda panah alur proses.	

Activity diagram menunjukkan apa yang terjadi, tetapi tidak menunjukkan siapa yang melakukan apa. Dalam pemprograman hal tersebut tidak menunjukkan *class* mana yang bertanggungjawab atas setiap *action*. Pada pemodelan bisnis, hal tersebut tidak bisa menunjukkan organisasi mana yang menjalankan sebuah *action*. *Swimlane* adalahsebuah cara untuk mengelompokan *activity* berdasarkan *actor* (mengelompokkan *activity* dari sebuah urutan yang sama). *Actor* bisa ditulis nama *actor* ataupun sekaligus dengan lambang *actor* (*stick figure*) pada *usecase diagram. Swimlane* digambarkan secara vertikal, walaupun terkadang digambarkan secara horizontal.

Activity diagram merupakan salah satu diagram yang umum digunakan dalam UML untuk menjabarkan proses atau aktivitas dari aktor. Sebagai contoh, pelanggan melakukan *login* (masuk) pada halaman *website* untuk bergabung, jika pelanggan belum terdaftar, maka akan ditolak oleh sistem dan dikembalikan. Proses penjabarannya adalah sebagai berikut :

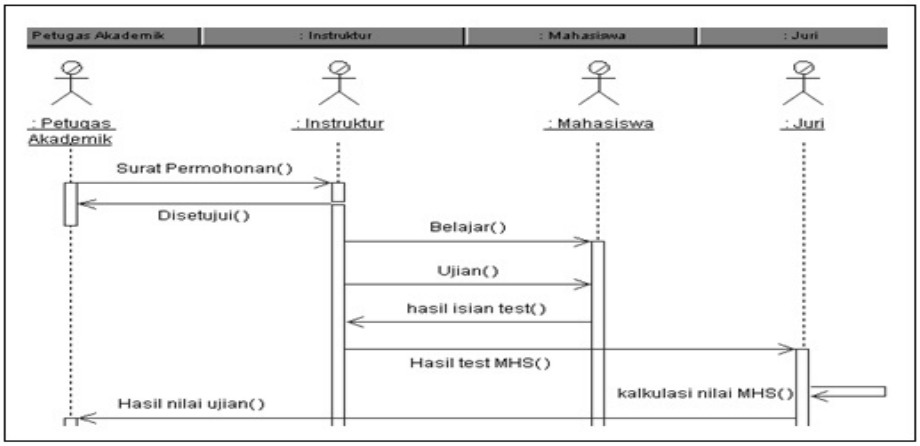


Gambar 2.10 *Activity* Diagram (Romi Satrio Wahono, 2003)

Di dalam *Activity* diagram tersebut dijelaskan bahwa *user* melakukan proses *login* untuk dapat memasuki area sistem, jika proses *login* dan/atau *user* belum teregistrasi, maka *user* akan ditolak oleh sistem tersebut dan diberi pesan *error*. Selain itu, bila *user* telah teregistrasi dan memasukkan kode *login* dengan benar maka akan diberi akses untuk masuk ke sistem, dan diberikan pesan sukses. *User* dapat *logout* (keluar) untuk mengakhiri sesi.

2.7.3 *Sequence* Diagram

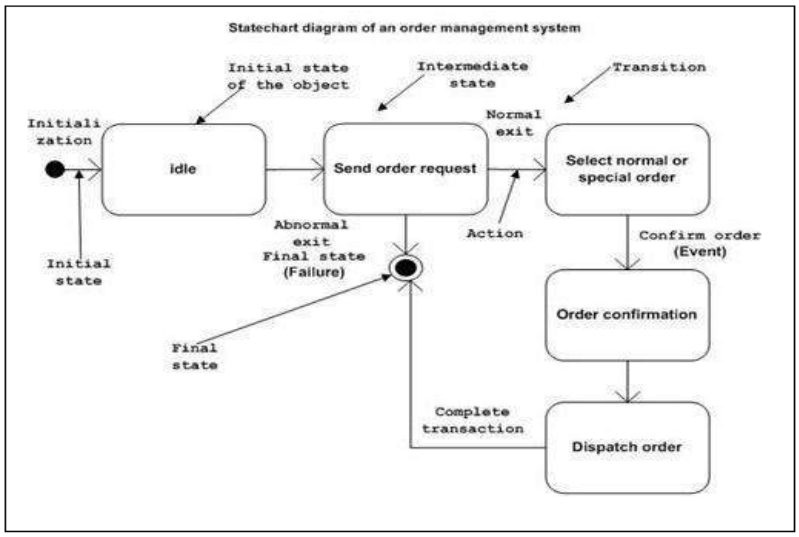
Sequence diagram menjelaskan secara detail urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case* : interaksi yang terjadi antar class, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi Berikut contoh sederhana *Sequence* diagram pada gambar 2.11.



Gambar 2.11 Contoh *Sequence* Diagram(Sri Dharwiyanti, 2003)

2.7.4 *Statechart Diagram*

Yaitu suatu diagram yang menggambarkan daur hidup pada suatu sistem dari awal objek itu terjadi hingga objek tersebut dieksekusi sampai proses destroy, dan menggambarkan perubahan keadaan atau transisi sistem pada suatu objek sebagai akibat dari stimulans yang diterima.



Gambar 2.12 Contoh *Statechart* Diagram(Sri Dharwiyanti, 2003)