BAB II

LANDASAN TEORI

II.1. Data Mining

”Data Mining adalah proses yang mempekerjakan satu atau lebih teknik pembelajaran komputer (*machine learning*) untuk menganalisis dan mengekstraksi pengetahuan (*knowledge*) secara otomatis” (Hermawati, 2009:3).

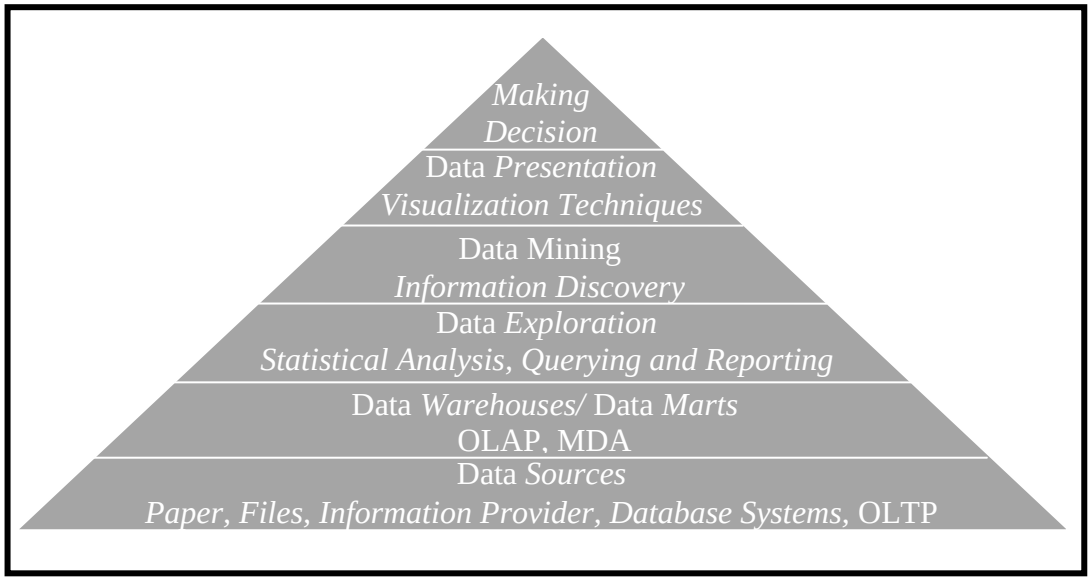
Definisi lain diantaranya adalah pembelajaran berbasis induksi (*induction-based learning*) adalah proses pembentukan definisi - definisi konsep umum yang dilakukan dengan cara mengobservasi contoh-contoh spesifik dari konsep-konsep yang akan dipelajari. *Knowledge Discovery in Databases* (KDD) adalah penerapan metode saintifik pada data mining. Dalam konteks ini, data mining merupakan satu langkah dari proses KDD.

Data mining merupakan proses iteratif dan interaktif untuk menemukan pola atau model baru yang sahih (sempurna), bermanfaat dan dapat dimengerti dalam suatu *database* yang sangat besar (*massive databases*).

1. Sahih, yaitu dapat digeneralisasi untuk masa yang akan datang.
2. Baru, yaitu apa yang sedang tidak diketahui.
3. Bermanfaat, yaitu dapat digunakan untuk melakukan suatu tindakan.
4. Iteratif, memerlukan sejumlah proses yang diulang.
5. Interaktif, memerlukan interaksi manusia dalam prosesnya.

Data mining berisi pencarian *trend* atau pola yang diinginkan dalam *database* besar untuk membantu pengambilan keputusan di waktu yang akan datang. Pola-pola ini dikenali oleh perangkat tertentu yang dapat memberikan

suatu analisa data yang berguna dan berwawasan yang kemudian dapat dipelajari dengan lebih teliti, yang mungkin saja menggunakan perangkat pendukung keputusan yang lainnya.



Gambar II.1 Data Mining dan Teknologi *Database* Lainnya
(Sumber : Hermawati, 2009:4)

Dari gambar di atas terlihat bahwa teknologi data *warehouse* digunakan untuk melakukan OLAP, sedangkan data mining digunakan untuk melakukan *information discovery* yang informasinya lebih ditujukan untuk seorang *Data Analyst* dan *Business Analyst* (dengan ditambah visualisasi tentunya). Dalam prakteknya, data mining juga mengambil data dari data *warehouse*. Hanya saja aplikasi dari data mining lebih khusus dan lebih spesifik dibandingkan OLAP mengingat *database* bukan satu-satunya bidang ilmu yang mempengaruhi data mining, banyak lagi bidang ilmu yang turut memperkaya data mining seperti : *information science* (ilmu informasi), *high performance computing,* visualisasi, *machine learning,* statistik, *neural network (*jaringan syaraf tiruan), pemodelan matematika, *information retrieval* dan *information extraction* serta pengenalan

pola. Bahkan pengolahan citra (*image processing*) juga digunakan dalam rangka melakukan data mining terhadap data *image/ spatial*.

Kemajuan yang luar biasa yang terus berlanjut dalam bidang data mining didorong oleh beberapa factor, antara lain :

1. Pertumbuhan yang cepat dalam kumpulan data.
2. Penyimpanan data dalam data *werehouse*, sehingga seluruh perusahaan memiliki akses ke dalam database yang baik.
3. Adanya peningkatan akses data melalui navigasi web dan internet.
4. Tekanan kompetisi bisnis untuk meningkatkan penguasaan pasar dalam globalisasi ekonomi.
5. Perkembangan teknologi perangkat lunak untuk data mining (ketersediaan teknologi).
6. Perkembangan yang hebat dalam kemampuan komputasi dan pengembangan kapasitas media penyimpanan.

Hubungan yang dicari dalam data mining dapat berupa hubungan antara dua atau lebih dalam satu dimensi. Misalnya dalam dimensi produk, dapat di lihat keterkaitan pembelian suatu produk dengan produk yang lain. Selain itu, hubungan juga dapat dilihat antara dua atau lebih atribut dan dua atau lebih objek.

Alasan mengapa melakukan data mining dari sudut pandang komersial karena :

1. Meledaknya volume data yang dihimpun dan disimpan dalam data *warehouse* seperti data *web*, *e-commerce*, penjualan di *departement store*, transaksi bank/ *credit card*.

2. Proses komputasi yang dapat diupayakan.
3. Kuatnya tekanan kompetitif untuk dapat menyediakan yang lebih baik, layanan-layanan *custom*-isasi dan informasi sedang menjadi produk yang berarti.

II.1.1. Operasi Data Mining

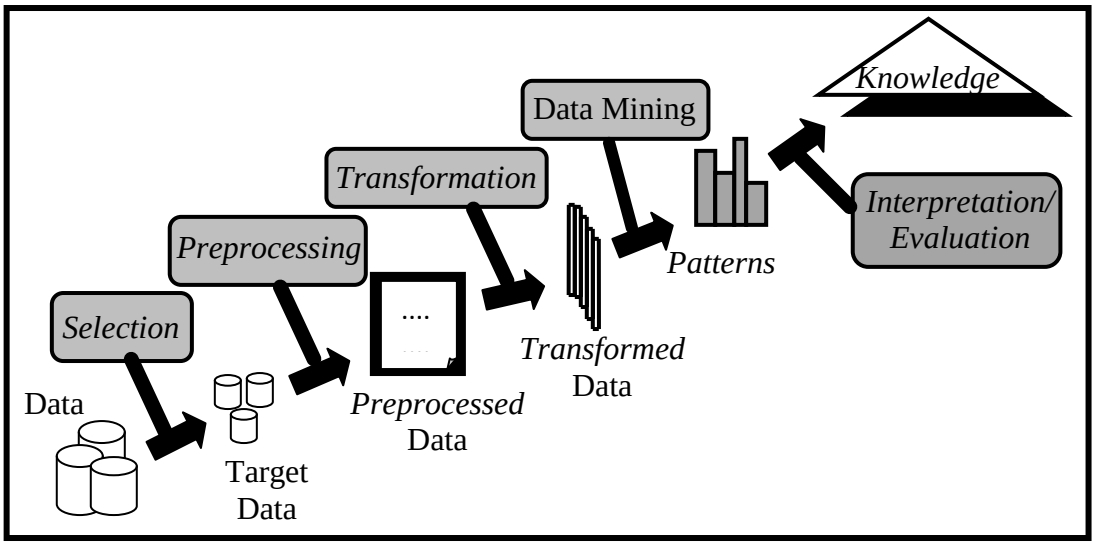
Operasi data mining menurut sifatnya dibedakan menjadi dua yaitu :

1. Prediksi (*prediction driven*), untuk menjawab pertanyaan apa dan sesuatu yang bersifat transparan. Operasi prediksi digunakan untuk validasi hipotesis, *querying* dan pelaporan, analisis multidimensi, OLAP (*Online Analytic Processing*) serta analisis statistik.
2. Penemuan (*discovery driven*) bersifat transparan dan untuk menjawab pertanyaan "mengapa?". Operasi penemuan digunakan untuk analisis data eksplorasi, pemodelan prediktif, segmentasi *database*, analisis keterkaitan (*link analysis*) dan deteksi deviasi.

Tahapan proses dalam penggunaan data mining yang merupakan proses *Knowledge Discovery in Databases* (KDD) seperti yang terlihat pada gambar 2.2 dapat diuraikan sebagai berikut :

1. Memahami *domain* aplikasi untuk mengetahui dan menggali pengetahuan awal serta apa sasaran pengguna.
2. Membuat target data-set yang meliputi pemilihan data dan fokus pada sub-set data.
3. Pembersihan dan transformasi data meliputi eliminasi derau, *outliers, missing value* serta pemilihan fitur dan reduksi dimensi.

4. Penggunaan algoritma data mining yang terdiri dari asosiasi, sekuensial, klasifikasi, klasterisasi, dll.
5. Interpretasi, evaluasi dan visualisasi pola untuk melihat apakah ada sesuatu yang baru dan menarik dan dilakukan iterasi jika diperlukan.



Gambar II.2 Proses KDD
(Sumber : Hermawati, 2009:6)

II.1.2. Tantangan Dalam Data Mining

Tantangan dalam data mining meliputi :

1. *Scalability*, yaitu besarnya ukuran basis data yang digunakan.
2. *Dimensionality*, yaitu banyaknya jumlah atribut dalam data yang akan diproses.
3. *Complex and Heterogeneous Data*, yaitu data yang kompleks dan mempunyai variasi yang beragam.
4. Data *Quality*, kualitas data yang akan diproses seperti data yang bersih dari *noise*, *missing value*, dsb.

5. *Data Ownership and Distribution*, yaitu siapa yang memiliki data dan bagaimana distribusinya.
6. *Privacy Preservation*, yaitu menjaga kerahasiaan data yang banyak diterapkan pada data nasabah perbankan.
7. *Streaming Data*, yaitu aliran data itu sendiri.

(Hermawati, 2009:19)

II.1.3. Teknik Data Mining

Beberapa teknik dan sifat data mining adalah sebagai berikut :

1. *Classification* (Klasifikasi)

Klasifikasi adalah menentukan sebuah *record* data baru ke salah satu dari beberapa kategori (*Class*) yang telah di defenisikan sebelumnya. Disebut juga dengan '*Supervised Learning*'. Berikut beberapa aplikasi dari klasifikasi :

a) Penjualan Langsung (*direct marketing*)

Tujuan : mengurangi *cost* surat menyurat dengan menentukan (*targeting*) satu set konsumen yang mempunyai kesamaan dalam membeli produk telepon seluler baru.

Pendekatan :

a. Gunakan data penjualan untuk suatu produk telepon selular.
b. Kita mengetahui pelanggan yang memutuskan untuk membeli dan yang memutuskan untuk tidak membeli. Keputusan {*buy, don't buy*} ini membentuk *class attribute*.

c. Himpunan bermacam demografi, gaya hidup dan *company-interaction* sehubungan dengan informasi mengenai pelanggan tertentu. Misalkan : Tipe bisnis, dimana mereka tinggal, berapa banyak mereka membayar, dll.
d. Gunakan informasi tersebut sebagai atribut *input* untuk mempelajari suatu model klasifikasi.

b) *Fraund Detection*

Tujuan : memprediksi kasus – kasus transaksi curang dengan menggunakan kartu kredit.

Pendekatan :

a. Gunakan transaksi kartu kredit dan informasi pemegang kartu kredit sebagai atirbutnya. Misalkan : kapan seorang pelanggan membeli, apa yang dibeli, apa selalu membayar tepat waktu, dsb.
b. Beri label transaksi – transaksi sebelumnya sebagai transaksi '*fraund* atau '*fair*' dan bentuk ini menjadi *class attribute*.
c. Pelajari satu model untuk *class* transaksi tersebut.
d. Gunakan model ini untuk mendeteksi kecurangan dengan mengobservasi transaksi kartu kredit tiap *account*.

c) *Customer Attrition / Churn*

Tujuan : untuk memprediksi pelanggan mana yang akan berpindah ke competitor kita.

Pendekatan :

a. Gunakan *record* transaksi dengan pelanggan yang lalu maupun dengan yang sekarang untuk mendapatkan atribut, seperti : seberapa sering pelanggan menghubungi, dimana dia menghubungi, pada hari apa dia paling sering menghubungi, status keuangannya, status perkawinannya, dsb.
b. Beri label pelanggan sebagai ‘setia’ atau ‘tidak setia’.
c. Temukan suatu model untuk ‘*loyalty*’.

2. Regrasi

Memprediksi nilai dari suatu *variable continue* yang diberikan berdasarkan nilai dari variable yang lain, dengan mengamsusikan sebuah model ketergantungan linier atau nonlinier. Teknik ini banyak dipelajari dalam statitiska, bidang jaringan saraf tiruan (*neural network*). Contoh aplikasinya:

a. Memprediksi jumlah penjualan produk baru berdasarkan pada belanja promosi / iklan.
b. Memprediksi kecepatan angina sebagai suatu fungsi suhu, kelembapan, tekanan udara, dsb.
c. *Time Series Prediction* dari indeks *stock market*.

3. *Clustering* (Klasterisasi)

Mempartisi data-set menjadi beberapa sub-set atau kelompok sedemikian rupa sehingga elemen – elemen dari suatu kelompok tertentu memiliki set property yang di *share* bersama, dengan tingkat simillitas yang tinggi dalam suatu kelompok dan tingkat similiritas antar kelompok yang rendah. Disebut juga dengan ‘*unsupersived learning*’. Jika diberikan sejumlah titik

data yang masing – masing mempunyai sejumlah atribut dan dengan menggunakan satu ukuran similaritas, dapat ditemukan klaster – klaster sedemikian sehingga :

a. Titik – titik data dalam satu klaster mempunyai similaritas yang lebih besar.
b. Titik – titik data dalam klaster yang berbeda mempunyai similaritas yang kecil.

Ukuran similaritas yang digunakan (a) *Euciledean Distance* jika atributnya kontinyu (b) Permasalahan lain – ukuran tertentu. Aplikasi klasterisasi diantaranya adallah :

1) *Market Segmentation*

   Tujuan : membagi pasar kedalam sub-set pelanggan yang berbeda, dimana suatu sub-set mungkin dapat dipillih sebagai target pasar yang dicapai dengan satu kombinasi pemasaran yang berbeda.

   Pendekatan :

   a. Kumpulkan atribut dari pelanggan yang berbeda berdasarkan pada informasi tempat tinggal dan gaya hidup.
   b. Tentukan klaster dari pelanggan – pelanggan yang sama.
   c. Hitung kualitas klaster dengan mengobservasi pola daya beli pelanggan pada klaster yang samaversus dari kalster yang berbeda.

2) *Document Clustering*

Tujuan : untuk mendapatkan kelompok dokumen yang mempunyai kesamaan berdasarkan pernyataan atau kata – kata penting yang muncul dalam dokumen tersebut.

Pendekatan : untuk mengenali kata – kata yang sering muncul dalam tiap dokumen. Dari suatu pengukuran similaritas yang didasarkan pada frekuensi *term* yang berbeda. Gunakan pengukuran ini untuk membentuk klaster – klaster.

Pencapaian : *Information Retrieval* dapat dimanfaatkan untuk menghubungkan suatu dokumen baru atau mencari *term* ke dokumen – dokumen yang di klaster.

4. *Association Rule Discovery*

Mendeteksi kumpulan atribut – atribut yang muncul bersamaan dalam frekuensi yang sering, dan membentuk sejumlah kaidah dari kumpulan – kumpulan tersebut. Contoh : 90% orang yang berbelanja di suatu supermarket yang membeli roti juga membeli selai, dan 60% dari semua orang yang berbelanja membeli keduanya.

Jika diberikan sekumpulan *record* yang masing – masing terdiri dari sejumlah item dari kumpulan yang diberikan akan menghasilkan aturan ketergantungan (*depency rules*) yang akan memprediksi kejadian dari satu item berdasarkan kejadian item lainnya.

Contoh aplikasi kaidah asosiasi adalah sebagai berikut :

a. *Marketing and Sales Promotion*

Misalkan diketahui aturan ketergantungan dimana {Bagels,…} ➔ {Potato Chips}

Potato Chips sebagai Consequent => dapat digunakan untuk menentukan apa yang dapat dilakukan untuk meningkatkan penjualan.

Bagels in the antecedent => dapat digunakan untuk melihat produk mana yang akan terkena dampak jika took tersebut tidak lagi menjual *bagels*.

Bagels in antecedent and Potato chips in consequent => dapat digunakan untuk melihat produk apa yang harus dijual dengan *bagels* untuk mempromosikan penjualan *Potato Chips*.

b. *Supermarket Shelf Management*

Tujuan : untuk mengenali item – item yang dibeli bersama – sama oleh cukup banyak pelanggan.

Pendekatan : memproses data *point-of-sale* yang dikumpulkan dengan pemindai *barcode* untuk menemukan ketergantungan antar item.

Aturan klasik : jika seorang pelanggan membeli diaper dan susu maka dia juga akan membeli beer. Sehingga jangan kaget jika anda akan menemukan enam pak beer yang ditumpuk dekat diaper.

c. *Inventory Management*

Tujuan : seorang pelanggan perusahaan-perbaikan-peralatan mengharapkan keaslian dari perbaikan produk konsumen dan menjaga pelayanan dengan menggunakan suku cadang yang baik untuk mengurangi jumlah kunjungan ke rumah pelanggan.

Pendekatan : memproses data peralatan dan suku cadang yang dibutuhkan pada perbaikan sebelumnya ditempat pelanggan yang berbeda dan menemukan pola – pola kejadian yang berulang.

Association rules merupakan salah satu metode yang bertujuan mencari pola yang sering muncul diantara banyak transaksi, dimana setiap transaksi terdiri dari beberapa *item* sehingga metode ini akan mendukung sistem rekomendasi melalui penemuan pola antar *item* dalam transaksi-transaksi yang terjadi. Metodologi dasar analisis asosiasi terbagi menjadi tiga tahap, yaitu :

1. Analisa Pola Frekuensi Tinggi

   Tahap ini mencari kombinasi *item* yang memenuhi syarat minimum dari nilai *support* dalam *database*. Nilai *support* sebuah *item* diperoleh dengan rumus berikut :

$$\textbf{\textit{Support}} = \frac{\text{Jumlah Transaksi mengandung A}}{\text{Total Transaksi}} \times 100\%$$

   Sedangkan nilai *support* dari dua *item* diperoleh dari dua rumus berikut :

$$\textbf{\textit{Support}}\ A \cap B = \frac{\text{Jumlah Transaksi mengandung A dan B}}{\text{Total Transaksi}} \times 100\%$$

2. Pembentukan Aturan Asosiatif

   Setelah semua pola frekuensi tinggi ditemukan, barulah dicari aturan assosiatif yang memenuhi syarat *minimum* untuk *confidence* dengan menghitung *confidence* aturan assosiatif A_B. Nilai *confidence* dari aturan A_B diperoleh dari rumus berikut :

$$\textbf{\textit{Confidence}} = P(A \mid B) \quad \frac{\text{Jumlah Transaksi Mengandung A dan B}}{\text{Jumlah Transaksi Mengandung A}} \times 100\%$$

3. *Frequent Item set*

   Langkah pertama pada *association rule* adalah menghasilkan semua *Item set* yang memungkinkan dengan kemungkinan item *set* yang muncul dengan *m-item* adalah 2*m*. Karena besarnya komputasi untuk menghitung *frequent item set*, yang membandingkan setiap kandidat item *set* dengan setiap transaksi, maka ada beberapa pendekatan untuk mengurangi komputasi tersebut, salah satunya dengan algoritma *apriori*.

5. *Sequence Mining* (Pencarian Pola Sekuensial)

Mencari sejumlah *event* yang secara umum terjadi bersama – sama. Contoh : dalam satu set urutan DNA, ACGTC diikuti oleh GTCA setelah suatu celah selebar 9 dengan probabilitas sebesar 30%.

Jika diberikan sekumpulan objek, dengan masing – masing objek dihubungkan dengan waktu kejadiannya maka dapatkan pola yang memprediksi ketergantungan sekuensial (*sequential dependencies*) yang kuat diantara kejadian – kejadian yang berbeda.

Pola – pola sekuensial pertama, pada dasarnya dibentuk dengan cara mencari semua kemungkinan pola yang ada. Nilai – nilai kejadian dalam pola diatur berdasarkan urutan waktu kejadian.

II.2. Algoritma *Apriori*

Algoritma *apriori* termasuk jenis aturan asosiasi pada data mining. Selain apriori, yang termasuk pada golongan ini adalah metode *Generalized Rule*

Induction dan algoritma *Hash Based.* Aturan yang menyatakan asosiasi antara beberapa atribut sering disebut *affinity analisys* atau *market basket analysis.*

Analisis asosiasi atau *association rule mining* adalah teknik data mining untuk menemukan aturan asosiatif antara suatu kombinasi item. Algoritma *apriori* yang bertujuan untuk menemukan *frequent item set* yang dijalankan pada sekumpulan data. Analsis apriori didefenisikan sebagai suatu proses untuk menemukan semua aturan apriori yang memenuhi syarat *minimum support* kemudian mendapatkan *rule* yang memenuhi *minimum confidence* dari *frequent item set*. Algoritma ini mengontrol berkembangnya kandidat item *set* dari hasil *frequent item set* dengan *support-based pruning* untuk menghilangkan item *set* yang tidak menarik dengan menetapkan *minimum support*. Prinsip dari apriori ini adalah bila item *set* digolongkan sebagai *frequent item set*, yang memiliki *support* lebih dari yang ditetapkan sebelumnya, maka semua subsetnya juga termasuk golongan *frequent item set*, dan sebaliknya. Contoh aturan asosiatif dari analisis pembelian disuatu pasar swalayan adalah dapat diketahuinya berapa besar kemungkinan seorang pelanggan membeli roti bersamaan dengan susu. Dengan pengetahuan tersebut, pemilik pasar swalayan dapat mengatur penempatan barangnya atau memasang kampanye pemasaran dengan memakai kupon diskon untuk kombinasi barang tertentu. (Kusrini, 2009:149)

Algoritma apriori adalah suatu algoritma dasar yang diusulkan oleh Agrawal & Srikant pada tahun 1994 untuk menentukan *frequent* itemset untuk aturan asosiasi Boolean.

Algoritma apriori dibagi menjadi beberapa tahap yang disebut narasi atau pass :

1. Pembentukan kandidat itemset.

   Kandidat k-itemset dibentuk dari kombinasi (k-1) itemset yang didapat dari iterasi sebelumnya. Satu cara dari algoritma apriori adalah pemangkasan kandidat k-itemset yang subsetnya 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 menscan *database* untuk menghitung jumlah transaksi yang memuat semua item didalam kandidat k-itemset tersebut. Ini juga adalah ciri dari algoritma apriori dimana diperlukan perhitungan dengan cara 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 supportnya lebih besar dari *minimum support*.

4. Bila tidak didapati pola frekuensi tinggi baru maka seluruh proses dihentikan.

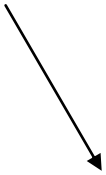
Contoh dari penerapan algoritma *apriori* adalah diilustrasikan digambar berikut :

Item	**Count**
Bread	4
Coke	2
Milk	4
Beer	3
Diaper	4
Eggs	1

Item (1-*Item Set*)

Item set	Count
Bread, Milk	3
Bread, Beer	2
Bread, Diaper	3
Milk, Beer	2
Milk, Diaper	3
Beer, Diaper	3

Pairs (2-*Item Set)*



Triplets (3-*Item Set*)

Item set	Count
Bread, Milk, Diaper	3
Milk, Diaper, Beer	2

***Minimum Support* = 3**

Gambar II.3 Ilustrasi Algoritma *Apriori*

II.2.1. *Rule Generation*

Setelah mendapatkan *frequent item set* menggunakan algoritma apriori, langkah selanjutnya adalah mendapatkan *rule* yang memenuhi *confidence*. Karena *rule* yang dihasilkan berasal dari *frequent item set*, dengan kata lain, dalam menghitung *rule* menggunakan *confidence*, tidak perlu lagi menghitung *support*-nya karena semua calon *rules* yang dihasilkan telah memenuhi *minimum support* sesuai yang ditentukan. Penghitungan ini juga tidak perlu melakukan perulangan *scanning* pada *database* untuk menghitung *confidence*, cukup dengan mengambil *Item set* dari hasil *support.*

II.2.2. Implementasi Apriori

Misalkan :

ID	Itemset
1	A.Susu, C.Teh, D.Kopi
2	B.Tepung, C.Teh, E.Mentega
3	A.Susu, B.Tepung, C.Teh, E.Mentega
4	B.Tepung, E.Mentega

Misalkan diinginkan *minimum support* : 50% (2 dari 4 transaksi)

Langkah 1:

L1={large 1-itemset}

Itemset	**Support**
A	50%
B	75%
C	75%
D	25%
E	75%

Langkah 2: Mencari kandidat itemset untuk L2:

2.1 : Gabungkan itemset pada L1 (algoritma apriori-gen)

{ A B, A C, A D, A E, B C, B D, B E, C D, C E, D E}

2.2 : Hapus yang tidak ada dalam itemset

Itemset { B D, DE} dihapus karena tidak ada dalam itemset

Langkah 3 :

Hitung support dari setiap kandidat itemset

Itemset	**Support**
A B	25 %
A C	50 %
A D	25 %
A E	25%
B C	50%
B E	75%
C D	25%
C E	50%

Langkah 4 :

L2 { large 2-itemset}

Itemset	**Support**
A C	50 %
B C	50%

B E	75%
C E	50%

Langkah 5 : Ulangi langkah 2-4

5.1 : Gabungkan itemset pada L2 & L2:

Itemset	**Hasil Gabungan 3 itemset**
A C + B C	A C B
A C + B E	A C B, A C E, A B E
A C + C E	A C E
B C + B E	B C E
B C + C E	B C E
B E + C E	B C E

5. 2 : Hapus yang tidak ada dalam itemset : { A C E }

Langkah 6 : Hitung support dari setiap kandidat itemset L3

Itemset	**Support**
A B C	25 %
A B E	25 %
B C E	50 %

Langkah 7 : L3 { large 3-itemset } { B C E}

Langkah 8 : Hentikan penelusuran karena sudah tidak ada lagi kandidat untuk 4-itemset.

Dari hasil – hasil diatas hasil akhir sebagai berikut:

L1

A	50%
B	75%
C	75%

L2

A C	50%
B C	50%
B E	75%

L3

B C E	50%

D	25%
E	75%

C E	50%

Untuk mencari aturan asosiasi diperlukan juga *minimum confidence*

Misal minconf : 75 %, aturan asosiasi yang mungkin terbentuk:

Aturan (X ➔ Y)	Sup(X ∪ Y)	Sup(X)	Confidence
B C → E	**50%**	**50%**	**100%**
B E → C	50%	75%	66.67%
C E → B	**50%**	**50%**	**100%**
A → C	**50%**	**50%**	**100 %**
C → A	50%	75%	66.67%
B → C	50%	75%	66.67%
C → B	50%	75%	66.67%
B → E	**75%**	**75%**	**100%**
E → B	**75%**	**75%**	**100%**
C → E	50%	75%	66.67%
E → C	50%	75%	66.67%

II.3. Pemodelan Sistem

II.3.1. *Unified Modelling Language (UML)*

Pada perkembangan teknologi perangkat lunak, diperlukan adanya bahasa yang digunakan untuk memodelkan perangkat lunak yang akan dibuat dan perlu adanya standarisasi agar orang diberbagai negara dapat mengerti pemodelan perangkat lunak. Seperti yang kita ketahui bahwa menyatukan banyak kepala untuk menceritakan sebuah ide dengan tujuan untuk memahami hal yang sama

tidaklah mudah, oleh karena itu diperlukan sebuah bahasa pemodelan perangkat lunak yang dapat dimengerti oleh banyak orang.

Banyak orang yang telah membuat bahasa pemodelan pembangunan perangkat lunak sesuai dengan teknologi pemrograman yang berkembang pada saat itu, misalnya yang sempat berkembang dan digunakan oleh banyak pihak adalah *Data Flow Diagram* (DFD) untuk memodelkan perangkat lunak yang menggunakan pemrograman prosedural atau struktural, kemudian juga ada *State Transition Diagram* (STD) yang digunakan untuk memodelkan sistem *real time* (waktu nyata).

Pada perkembangan teknik pemrograman berorientasi objek, munculah sebuah standarisasi bahasa pemodelan untuk pembangunan perangkat lunak yang dibangun dengan menggunakan teknik pemrograman berorientasi objek, yaitu *Unified Modelling Language* (UML). UML muncul karena adanya kebutuhan pemodelan visual untuk menspesifikasi, menggambarkan, membangun dan dokumentasi dari sistem perangkat lunak.

"UML merupakan bahasa visual untuk pemodelan dan komunikasi mengenai sebuah sistem dengan menggunakan diagram dan teks - teks pendukung" (Shalahuddin, M. dan Rosa A.S, 2014:137)

UML hanya berfungsi untuk melakukan pemodelan. Jadi penggunaan UML tidak terbatas pada metodologi tertentu, meskipun pada kenyataannya UML paling banyak digunakan pada metodologi berorientasi objek. Seperti yang kita ketahui bahwa banyak hal di dunia sistem informasi yang tidak dapat dibakukan, semua tergantung kebutuhan, lingkungan dan konteksnya. Begitu juga dengan

perkembangan penggunaan UML bergantung pada *level* abstraksi penggunaannya. Jadi belum tentu pandangan yang berbeda dalam penggunaan UML adalah suatu yang salah, tapi perlu ditelaah dimanakah UML digunakan dan hal apa yang ingin divisualkan.

II.3.2. *Use Case Diagram*

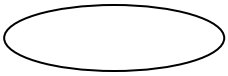
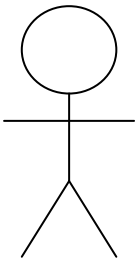
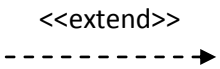
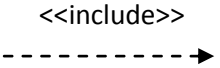
Use case atau diagram use case merupakan pemodelan untuk melakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi - fungsi itu.

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 aktor dan *use case.*

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

   Berikut adalah simbol-simbol yang ada pada diagram *use case* :

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

Simbol	Nama	Keterangan
	*Use Case*	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*.
	*Actor*	Orang, proses, atau sistem lain yang berorientasi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang; biasanya dinyatakan menggunakan kata benda di awal frase nama aktor.
	Association	Komunikasi antar aktor dan *use case* yang berpartisipasi pada *use case* atau *use case* memiliki interaksi dengan aktor.
	*Extend*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu; mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek; biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan.
	Generalization	Hubungan generalisasi dan spesialisasi (umum-khusus) antara dua buah *use case* dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya.
	*Include*	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini.

(Sumber : Shalahuddin, M. dan Rosa A.S, 2014:156-158, Rekayasa Perangkat Lunak)

II.3.3. *Class Diagram*

Diagram kelas atau *class diagram* menggambarkan struktur system dari segi pendefenisian kelas – kelas yang akan dibuat untuk membangun system. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variable – variable yang dimiliki oleh suatu kelas. Sedangkan metode atau operasi adalah fungsi – fungsi yang dimiliki oleh suatu kelas.

Diagram kelas dibuat agar pembuatan program atau *programmer* membuat kelas – kelas sesuai rancangan didalam diagram kelas agar antara dokumentasi perancangan dan perangkat lunak sinkron.

Kelas – kelas yang ada pada struktur system harus dapat melakukan fungsi – fungsi sesuai dengan kebutuhan system sehingga pembuatan perangkat lunak atau *programmer* dapat membuat kelas – kelas didalam program perangkat lunak sesuai dengan perancangan diagram kelas. Susunan struktur kelas yang baik pada diagram kelas sebaiknya memiliki jenis – jenis kelas berikut :

1. Kelas Main

   Kelas yang memiliki fungsi awal dieksekusi ketika system dijalankan.

2. Kelas yang menangani tampilan system (*view*)

   Kelas yang mendefenisikan dan mengatur tampilan ke pemakai.

3. Kelas yang diambil dari pendefenisian *use case* (*controller*)

   Kelas yang menangani fungsi – fungsi yang harus ada diambil dari pendefenisian *use case,* kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada perangkat lunak.

4. Kelas yang diambil dari pendefenisian data (model)

Kelas yang digunakan untuk memgang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data. Semua table yang dibuat di basis data dapat dijadikan kelas, namun untuk table dari hasil relasi atau atribut *multivalue* pada ERD dapat dijadikan kelas tersendiri dapat juga tidak asalkan pengaksesannya dapat dipertanggung jawabkan atau tetap ada didalam perancangan kelas. (Rosa A.S, 2013:141-143)

II.3.4. *Activity Diagram*

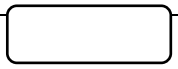
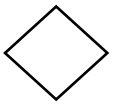
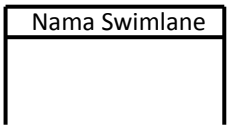
Diagram aktivitas atau *activity diagram* menggambarkan *work flow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis atau menu yang ada pada perangkat lunak. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan *actor*, jadi aktivitas yang dapat dilakukan oleh sistem.

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 system / *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.
4. Rancangan menu yang ditampilkan pada perangkat lunak.

Berikut adalah symbol - simbol yang ada pada diagram aktivitas :

Tabel II.2 Simbol *Activity Diagram*

Simbol	Nama	Keterangan
	Status Awal	Status awal aktivitas sistem, sebuah diagram aktivitas memiliki sebuah status awal.
	Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
	Percabangan/ *Decision*	Asosiasi percabangan dimana jika ada pilihan aktivitas lebih dari satu.
	Penggabungan/ *Join*	Asosiasi penggabungan dimana lebih dari satu aktivitas digabungkan menjadi satu.
	Status Akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
	Swimlane	Memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

(Sumber : Shalahuddin, M. dan Rosa A.S, 2014:161-163, Rekayasa Perangkat Lunak)

II.3.5. *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 dikethaui objek – objek yang terlibat dalam sebuah *use case* beserta metode – metode yang dimiliki kelas yang diinstasiasi menjadi objek itu. Membuat diagram sekuen juga dibutuhkan untuk melihat *scenario* yang ada pada *use case*.

Banyaknya diagram sekuen yang harus digambar adalah minimum sebanyak pendefenisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefenisikan interaksi jalannya pesan sudah dicakup pada diagram sekuen sehingga semakin banyak *use case* yang didefenisikan maka diagram sekuan yang harus dibuat juga semakin banyak. (Shalahuddin, M. dan Rosa A.S, 2014:165).

II.4. Microsoft SQL Server 2008

SQL server 2008 adalah sebuah terobosan baru *Microsoft* dalam bidang *database*. SQL Server adalah DBMS (*Database Management System*) yang dibuat oleh *Microsoft* untuk ikut berkecimpung dalam persaingan dunia pengoalahan data menyusul pendahulunya seperti IMB dan *Oracle*. SQL Server 2008 dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa SQL Server 2008 membawa beberapa terobosan dalam bidang pengolahan dan penyimpanan data.

Microsoft merilis SQL Server 2008 dalam beberapa versi yang disesuaikan dengan segmen – segmen pasar yang dituju. Menurut cara pemrosesan data pada prosesor maka *Microsoft* mengelompokkan produk ini berdasarkan dua jenis yaitu:

a. Versi 32-bit(x86), yang biasanya digunakan untuk computer dengan *single processor* (Pentium 4) atau lebih tepatnya prosesor 32 bit dan system operasi Windows XP.

b. Versi 64-bit(x64), yang biasanya digunakan untuk computer dengan lebih dari satu prosesor (misalnya *Core* 2 *Duo*) dan system operasi 64-bit seperti Windows XP 64, Vista, dan Windows 7.

Sedangkan secara keseluruhan terdapat versi – versi seperti berikut ini :

a. Versi *Compact*, ini adalah versi "tipis" dari semua versi yang ada. Versi ini seperti versi *desktop* pada SQL Server 2008. Versi ini juga digunakan pada *handled drice* seperti pocket PC, PDA, *Smartphone*, Tablet PC.

b. Versi *Express*, ini adalah versi "ringan" dari semua versi yang ada (tetapi versi ini berbeda dengan versi *compact*) dan paling cocok untuk latihan para pengembangan aplikasi. Versi ini memuat *Express Manager Standar*, integrasi dengan CLR dan XML. (Wahana Komputer, 2011:2)

II.5. Microsoft Visual Studio 2010

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap (*suite*) yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasinya, dalam bentuk aplikasi console, aplikasi Windows, ataupun aplisai Web. Visual Studio mencakup *compiler*, SDK, *Integrated Development Environment* (IDE), dan dokumentasi (umumnya berupa MSDN *Library*). *Compiler* yang dimasukkan ke dalam paket Visual Studio antara lain Visual C++, Visual C#, Visual Basic, Visual Basic.Net, Visual InterDev, Visual J++, Visual J#, Visual FoxPro, dan Visual SourceSafe.