

## **BAB 2**

### **LANDASAN TEORI**

#### **2.1 *Data Warehouse***

##### **2.1.1 *Pengertian Data Warehouse***

*Data Warehouse* merupakan sebuah gudang data yang berisi data dalam jumlah besar dan digunakan untuk proses analisa dan pembuatan laporan yang dibutuhkan perusahaan.

Hal ini diperkuat oleh teori dari Han Jiawei (2011:p143) yang mengatakan bahwa *data warehouse* adalah tempat penyimpanan informasi yang dikumpulkan dari berbagai sumber, disimpan dalam skema yang terpadu dan biasanya berada dalam suatu lokasi / situs.

Menurut W.H. Inmon dan Richard D.H., *data warehouse* adalah koleksi data yang mempunyai karakteristik berorientasi subjek, terintegrasi, *time-variant*, dan bersifat tetap dari koleksi data dalam mendukung proses pengambilan keputusan *management*.

##### **2.1.2 *Karakteristik Data Warehouse***

Berdasarkan definisi yang dikemukakan Han Jiawei (2011:p144) tentang *data warehouse*, maka *data warehouse* mempunyai empat buah karakteristik yaitu :

###### **1. *Subject Oriented***

*Data warehouse* diorganisasikan ke dalam banyak *subject* yang utama seperti *customer*, *product*, dan *sales*. *Data warehouse* tidak berkonsentrasi pada kegiatan operasional sehari-hari dan

proses transaksi pada suatu organisasi, *data warehouse* fokus terhadap *modeling* dan analisis data untuk pengambilan keputusan, oleh karena itu, *data warehouse* menghasilkan pandangan sederhana dan ringkasan terhadap *subject* tertentu dan mengabaikan data yang tidak *relevant* terhadap proses pendukung keputusan.

## **2. *Integrated***

*Data Warehouse* biasanya dibangun dengan mengintegrasikan sumber data yang berbeda-beda, seperti *relational database*, *flat files*, dan *online transactional*. Teknik *data cleaning* dan *data integration* digunakan untuk memastikan konsistensi dalam konversi penamaan, struktur pengkodean, ukuran atribut, dan sebagainya.

## **3. *Non-volatile***

*Data warehouse* secara fisik memisahkan pengumpulan data dari aplikasi data yang ditemukan dalam *operational environment*. Di dalam pemisahan *data warehouse* tidak memerlukan proses transaksi, *recovery*, dan *concurency control mechanism*. Biasanya hanya membutuhkan dua operasi dalam mengakses data, *initial loading data* dan *access of data*.

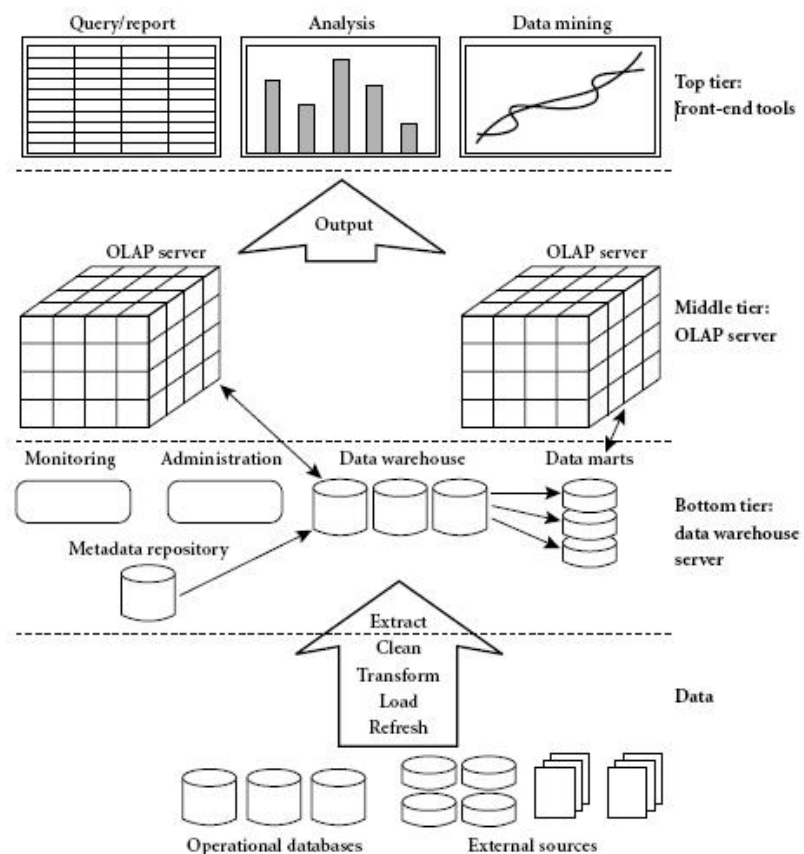
## **4. *Time variant***

Data disimpan untuk memberikan informasi dari perspektif *history* (misalnya, 5-10 tahun terakhir). Setiap struktur

kunci dalam *data warehouse* berisi baik secara implisit maupun eksplisit elemen waktu.

### 2.1.3 Arsitektur Data Warehouse

*Data warehouse* mengandung tiga tingkatan dalam arsitektural, seperti digambarkan pada gambar 2.1.3 :



12 A three-tier data warehousing architecture.

Gambar 2.1.3 Struktur *Data Warehouse*

### 1. *Data warehouse server*

Merupakan tingkatan paling bawah pada arsitektural *data warehouse*. *Data warehouse server* merupakan *relational database system*. *Back-end tools* dan *utilities* digunakan untuk menghasilkan data ke tingkat bawah dari *operasional database* atau sumber eksternal lainnya (misalnya, pelanggan informasi profil yang disediakan oleh konsultan eksternal). *Tools* dan *utilities* menghasilkan *data extraction*, *cleaning*, dan *transformation* (misalnya, untuk menggabungkan data yang sama dari sumber yang berbeda ke dalam format yang terpadu), seperti fungsi *load* dan *refresh* untuk *update* ke dalam *data warehouse*. Data yang diambil menggunakan antar-muka program aplikasi yang dikenal sebagai *gateway*. Sebuah *gateway* didukung oleh DBMS yang mendasari dan memungkinkan program klien untuk menghasilkan kode *SQL* yang akan dieksekusi di *server*. Contoh *gateway* meliputi *ODBC* (*Open Database Connection*) dan *OLEDB* (*Object Linking and Embedding Database*) dari *Microsoft* dan *JDBC* (*Java Database Connection*). Tingkatan ini berisi repositori *metadata*, yang menyimpan informasi tentang *data warehouse* dan isinya.

## 2. *OLAP Server*

Merupakan tingkatan menengah dalam arsitektural *data warehouse*. Biasanya diimplementasikan baik menggunakan model *relational OLAP (ROLAP)* (yaitu, perpanjangan dari *relational DBMS* yang memetakan operasi pada data multidimensi pada operasi relasional standar), atau model *multidimensional OLAP (MOLAP)* (yaitu, *server* yang mempunyai tugas yang khusus untuk mengarahkan implementasi multidimensi data dan operasi).

## 3. *Front End Client Layer*

Merupakan tingkatan atas pada arsitektural *data warehouse*. Berisikan *tool query*, alat analisis, dan *tools data mining* (Contoh, *trend analysis*, *prediction* dan sebagainya).

## 2.2 *Data Mining*

### 2.2.1 *Pengertian Data Mining*

*Data Mining* adalah proses menganalisa data yang berjumlah besar untuk menemukan suatu pola yang akan digunakan untuk pengambilan keputusan.

Hal ini diperkuat oleh teori dari Han Jiawei (2011:p36) yang mengatakan *data mining* adalah proses menemukan pola yang menarik dan pengetahuan dari data yang berjumlah besar.

Karakteristik *Data Mining* menurut Turban (2007:p230):

- a. Data seringnya terpendam dalam dalam *database* yang sangat besar yang kadang-kadang datanya sudah bertahun-tahun.
- b. Lingkungan *data mining* biasanya berupa arsitektur *client-server* atau arsitektur *system* informasi berbasis *web*.
- c. *Tool* baru yang canggih, termasuk *tool* visualisasi tambahan, membantu mennghilangkan lapisan informasi yang terpendam dalam *file-file* yang berhubungan atau *record-record* arsip *public*.
- d. Pemilik biasanya seorang *end user*, didukung dengan data *drill* dan *tool* penguasaan *query* yang lain untuk menanyakan pertanyaan dan mendapatkan jawaban secepatnya, dengan sedikit atau tidak ada kemampuan pemrograman.
- e. *Tool data mining* dengan kesediaannya dikombinasikan dengan *spreadsheet* dan *tool software* pengembangan yang lainnya.
- f. Karena besarnya jumlah data dan usaha pencarian yang besar-besaran, kadang-kadang diperlukan penggunaan proses paralel untuk *data mining*.

Kelebihan *Data Mining* sebagai alat analisis :

- a. *Data mining* mampu menangani *data* dalam jumlah besar dan kompleks
- b. *Data mining* dapat menangani *data* dengan berbagai macam tipe atribut.

- c. *Data mining* mampu mencari dan mengolah data secara semi-otomatis. Disebut semi-otomatis karena dalam beberapa teknik data mining, diperlukan parameter yang harus di-*input* oleh user secara manual
- d. *Data mining* dapat menggunakan pengalaman ataupun kesalahan terdahulu untuk meningkatkan kualitas dan hasil analisa sehingga mendapat hasil yang terbaik

Kekurangan *data mining* sebagai alat analisis :

- a. Terkadang ada *data* yang lebih mudah diselesaikan dengan *statistic manual* dibandingkan dengan menggunakan teknik *data mining*.
- b. *Data mining* tidak dapat menemukan *knowledge* secara *instant*.
- c. Algoritma *data mining* cukup kompleks
- d. Hasil dari *data mining* tidak dapat langsung digunakan, harus dianalisa dan diinterpretasi terlebih dahulu.

### 2.2.2 Tahap penemuan *Knowledge* pada *Data Mining*

Langkah menemukan pengetahuan pada *data mining* antara lain :

#### 1. *Data Cleaning*

Memperbaiki data yang salah, menghapus data yang rusak dan tidak konsisten

#### 2. *Data Integration*

Mengintegrasikan data dari berbagai macam sumber dan menyatukan agar mudah dipilih dan diproses nantinya

### 3. *Data Selection*

Memilih data yang dibutuhkan pada *database* dan digunakan untuk proses analisis

### 4. *Data Transformation*

Mengubah dan menggabungkan data dari berbagai macam bentuk menjadi satu bentuk yang sama agar mudah diproses.

### 5. *Data Mining*

Tahap untuk menerapkan metode dalam proses *modeling* data yang akan digunakan pada proses *Data mining*.

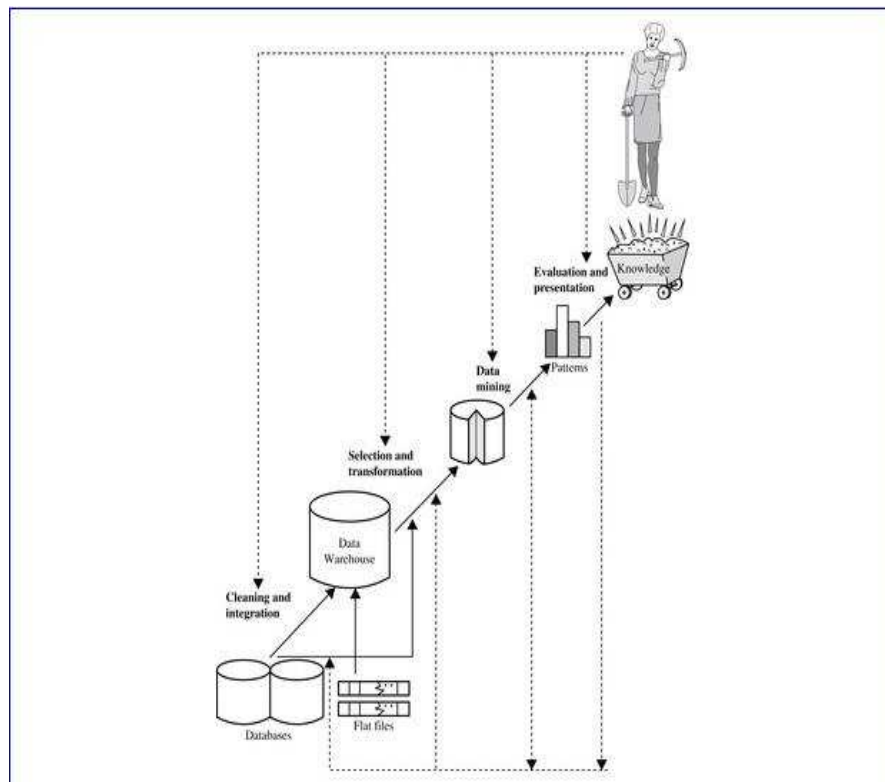
### 6. *Pattern Evaluation*

Melakukan evaluasi akan *pattern* yang telah diproses, aspek-aspek yang dievaluasi adalah hasil *output* yang didapat setelah proses *data mining* dilakukan

### 7. *Knowledge Presentation*

Melakukan penyajian hasil dari proses *data mining* yang sudah diproses.





Gambar 2.1 Proses *Knowledge Discovery*

Sumber : Jiawei, H. (2011:p35). *Data Mining : Concepts and Techniques*.

### 2.2.3 Teknik – Teknik *Data Mining*

Menurut Han Jiawei (2011) ada beberapa *teknik data mining* yang lazim digunakan, diantaranya :

#### 2.2.3.1. *Association Rule Mining / Frequent Pattern / Market Basket Analysis*

*Frequent patterns* adalah pola yang sering muncul dalam kumpulan data. Misalnya, satu *set item* seperti susu

dan roti yang sering muncul bersama-sama dalam satu *set* data transaksi adalah *frequent item set*. Sebuah *subsequence*, seperti membeli pertama kali *PC*, lalu kamera digital dan kemudian *memory card*, jika *sequence* tersebut sering terjadi dalam *history* pada *database* belanja, maka pola tersebut adalah *frequent pattern*. Menemukan *frequent pattern* adalah peranan penting dalam *mining associations*, *correlations*, dan hubungan menarik lainnya antara data. Selain itu, membantu dalam klasifikasi data, *clustering*, dan lainnya. *Frequent itemset mining* memungkinkan untuk menemukan asosiasi dan korelasi dari banyak *item* dari banyak data transaksi. Dengan banyaknya data yang terus terkumpul, banyak industri yang mulai tertarik pada pola *mining* tersebut dari *database* mereka. Penemuan hubungan korelasi yang menarik antara jumlah besar catatan transaksi bisnis dapat membantu dalam bisnis seperti dalam proses pengambilan keputusan untuk desain katalog, lintas pemasaran, dan analisis tingkah laku pelanggan. *Association rule mining* yang biasanya disebut juga *market basket analysis* adalah teknik *mining* untuk menemukan aturan asosiatif antara suatu kombinasi *item*. Contoh dari aturan asosiatif dari analisa pembelian di suatu pasar swalayan adalah bisa

diketahui berapa besar kemungkinan seorang pelanggan membeli roti bersamaan dengan susu. Dengan pengetahuan tersebut, pemilik pasar swalayan dapat mengatur penempatan barangnya atau merancang kampanye pemasaran dengan memakai kupon diskon untuk kombinasi barang tertentu. Penting tidaknya suatu aturan asosiatif dapat diketahui dengan dua *parameter*, *support* yaitu persentase kombinasi item tersebut dalam database dan *confidence* yaitu kuatnya hubungan antar *item* dalam aturan asosiatif.

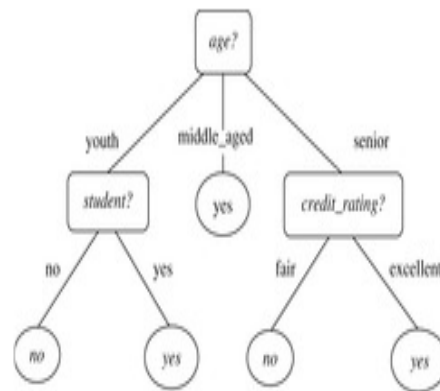
#### **2.2.3.2 Classification**

*Classification* adalah satu bentuk analisis data yang menghasilkan model untuk mendeskripsikan kelas data yang penting. Klasifikasi memprediksi kategori (*diskrit*, *unorderd*) ke dalam label kelas. Klasifikasi merupakan proses untuk menemukan model atau fungsi yang menjelaskan atau membedakan konsep atau kelas data, dengan tujuan untuk dapat memperkirakan kelas dari suatu objek yang labelnya tidak diketahui. Model itu sendiri bisa berupa aturan “jika-maka”, berupa *decision tree*, formula matematis atau *neural network*. Sebagai contoh, kita dapat membangun model klasifikasi untuk mengkategorikan

aplikasi pinjaman bank, aman atau beresiko. Analisa tersebut dapat membantu memberikan pemahaman yang lebih baik dari data pada umumnya. Klasifikasi memiliki berbagai aplikasi yaitu deteksi penipuan, pemasaran target, prediksi kinerja, manufaktur, dan diagnosa medis.

#### 2.2.3.2.1 *Decision tree*

*Decision Tree* adalah salah satu metode *classification* yang paling populer karena mudah untuk diinterpretasi oleh manusia. *Decision tree* menggunakan model seperti struktur pohon.



Gambar 2.2 *Decision tree*

Sumber : Jiawei, H. (2011:p334). *Data Mining : Concepts and Techniques*.

Pembangunan *decision tree* tidak memerlukan pengaturan *domain knowledge* atau parameter, karena itu cocok untuk

eksplorasi penemuan pengetahuan. Pohon keputusan dapat menangani data multidimensi. Perwakilan dari pengetahuan yang diperoleh dalam bentuk pohon memudahkan untuk dipelajari dan dipahami. *Decision tree* memiliki akurasi yang baik. Namun, keberhasilan penggunaannya tergantung pada data yang ada. Aplikasi klasifikasi *decision tree* telah digunakan dalam banyak area seperti kedokteran, manufaktur dan produksi, analisis keuangan, astronomi, dan biologi molekuler. Untuk menentukan proses pembangunan *decision tree* diperlukan adanya *attribute selection measure*, yaitu suatu metode untuk memilih kriteria pemisahan yang terbaik yang memisahkan partisi data yang diberikan, kelas-label ke dalam kelas individu. *Attribute selection measure* memberikan peringkat untuk setiap atribut. Jika atribut yang terpisah adalah *continuous-valued* atau jika kita dibatasi kedalam *binary trees*, maka *subset*

yang membelah juga harus ditentukan sebagai bagian dari kriteria pemisahan. *Node* pohon diciptakan untuk partisi yang dilabeli dengan kriteria pembagian, cabang yang tumbuh untuk setiap hasil dari kinerja. Tiga *selection measures attribute* yang populer adalah *information gain*, *gain ratio*, dan *gini index*.

#### 2.2.3.3. *Clustering*

*Clustering* adalah proses pengelompokan kumpulan data menjadi beberapa kelompok sehingga objek di dalam satu kelompok memiliki banyak kesamaan dan memiliki banyak perbedaan dengan objek di kelompok lain. Perbedaan dan persamaannya biasanya berdasarkan nilai atribut dari objek tersebut dan dapat juga berupa perhitungan jarak. *Clustering* sendiri juga disebut *Unsupervised classification*, karena *clustering* lebih bersifat untuk dipelajari dengan diperhatikan. *Cluster analysis* merupakan proses partisi satu *set* objek data ke dalam himpunan bagian. Setiap himpunan bagian adalah *cluster*, sehingga objek yang di dalam *cluster* mirip satu sama dengan lainnya, dan mempunyai perbedaan dengan objek dari *cluster* yang lain. Partisi tidak dilakukan dengan

manual tetapi dengan algoritma *clustering*. Oleh karena itu, *clustering* sangat berguna dan bisa menemukan *group* yang tidak dikenal dalam data. *Cluster analysis* banyak digunakan dalam berbagai aplikasi seperti *business intelligence*, *image pattern recognition*, *web search*, *biology*, dan *security*. Di dalam *business intelligence*, *clustering* bisa mengatur banyak *customer* ke dalam banyak *group*, contohnya mengelompokkan *customer* ke dalam beberapa *cluster* dengan kesamaan karakteristik yang kuat. *Clustering* juga dikenal sebagai *data segmentation* karena *clustering* mempartisi banyak data *set* ke dalam banyak *group* berdasarkan kesamaannya. *Clustering* juga bisa sebagai *outlier detection*, di mana *outliers* bisa menjadi menarik daripada kasus yang biasa. Aplikasinya adalah *outlier detection* untuk mendeteksi *card fraud* dan memonitori aktifitas kriminal dalam *e-commerce*. Contohnya adalah pengecualian dalam transaksi kartu kredit.

#### **2.2.3.3.1 Konsep dasar *Clustering***

Proses *clustering* akan menghasilkan *cluster* yang baik apabila:

1. Tingkat kesamaan yang tinggi dalam satu class

## 2. Tingkat kesamaan yang rendah antar class

Kesamaan yang dimaksud merupakan pengukuran secara *numeric* terhadap dua buah objek. Nilai kesamaan ini akan semakin tinggi bila dua objek yang dibandingkan memiliki kemiripan yang tinggi. Perbedaan kualitas hasil *clustering* bergantung pada metode yang dipakai.

Tipe *data* pada *Clustering* :

- a. Variabel berskala interval
- b. Variabel biner
- c. Variabel nominal, ordinal, dan rasio
- d. Variabel dengan tipe lainnya.

Metode *Clustering* juga harus dapat mengukur kemampuannya dalam usaha untuk menemukan suatu pola tersembunyi pada data yang tersedia. Dalam mengukur nilai kesamaan ini ada beberapa metode yang dapat dipakai. Salah satu metodenya adalah *weighted Euclidean Distance*. Dalam metode ini dua buah point dapat dihitung jaraknya bila diketahui nilai dari masing-



masing atribut pada kedua point tersebut, berikut rumusnya :

$$Distance(p, q) = (\sum_k^n \mu_k |P_k - q_k|^r)^{1/r}$$

Keterangan:

N = Jumlah *record* data

K= Urutan *field* data

r= 2

$\mu_k$ = Bobot *field* yang diberikan *user*

#### 2.2.3.3.2 *Requirements for Clustering*

Syarat untuk melakukan analisa clustering :

##### 1. *Scalability*

Mampu menangani data dalam jumlah yang besar. Karena *database* yang besar berisi lebih dari jutaan objek bukan hanya ratusan objek, maka dari itu diperlukan algoritma dengan *clustering* yang *scalable*.

##### 2. *Ability to deal with different types of attributes*

Banyak algoritma *clustering* yang hanya dibuat untuk menganalisa *data* bersifat *numeric*. Namun sekarang ini, aplikasi *data mining* harus dapat menangani berbagai macam

bentuk data seperti biner, *data nominal*, *data ordinal*, ataupun campuran.

### 3. *Discovery of clusters with arbitrary shape*

Banyak algoritma *clustering* yang menggunakan metode *Euclidean* atau *Manhattan*. Namun hasil dari metode tersebut bukan hanya berbentuk bulat seperti pada contoh. Hasil dapat berbentuk aneh dan tidak sama antara satu dengan yang lain. Maka dari itu diperlukan kemampuan untuk menganalisa *cluster* dengan bentuk apapun.

### 4. *Requirements for domain knowledge to determine input parameters*

Banyak algoritma *clustering* yang mengharuskan pengguna untuk memasukkan parameter tertentu, seperti jumlah *cluster*. Hasil *clustering* bergantung pada parameter yang ditentukan. Terkadang parameter sulit untuk menentukan, terutama pada data yang memiliki dimensi tinggi. Hal ini menyulitkan pengguna serta kualitas *clustering* yang dicapai pun tidak terkontrol.

5. *Ability to deal with noisy data*

Pada kenyataannya, data pasti ada yang rusak, *error*, tidak dimengerti, ataupun menghilang. Beberapa algoritma *clustering* sangat sensitif terhadap *data* yang rusak, sehingga menyebabkan *cluster* dengan kualitas rendah. Maka dari itu diperlukan *clustering* yang mampu menangani *data* yang rusak.

6. *Incremental clustering and insensitivity to input order*

*Data* yang dimasukkan dapat menyebabkan *cluster* menjadi berubah *total*. Hal ini dapat terjadi karena tidak sensitifnya algoritma *clustering* yang dipakai. Maka dari itu diperlukan algoritma yang tidak sensitif terhadap urutan *input data*.

7. *Capability of clustering high-dimensionality data*

Sebuah kelompok *data* dapat berisi banyak dimensi ataupun atribut. Kebanyakan algoritma

*clustering* hanya mampu menangani kelompok *data* dengan dimensi sedikit. Maka dari itu diperlukan algoritma *clustering* yang mampu menangani *data* dengan dimensi berjumlah banyak.

#### 8. *Constraint based clustering*

Pada kenyataannya, membuat *clustering* tentu saja memiliki beberapa pembatas ataupun syarat tertentu. Hal ini menjadi tugas yang menantang karena diperlukan kemampuan yang tinggi untuk mengelompokkan *data* dengan kendala dan perilaku tertentu.

#### 9. *Interpretability and usability*

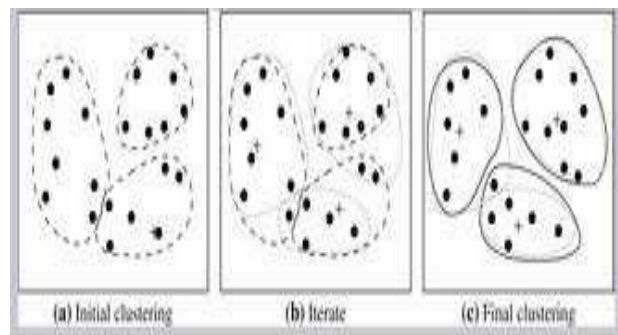
Pengguna tentu saja menginginkan hasil *clustering* mudah ditafsirkan, dimengerti, dan bermanfaat. Hal ini berarti *clustering* perlu ditandai dengan beberapa syarat sesuai kemauan *user* dan tentu saja hal itu mempengaruhi pemilihan metode *clustering* yang akan digunakan.

### 2.2.2.3.3 **Tipe *Clustering***

Berikut ini merupakan tipe *clustering* yang umum digunakan, antara lain :

### 1. *Partitional Clustering*

Metode yang paling sederhana dan paling mendasar dari analisis partisi *cluster*, yang mengatur objek dari suatu himpunan ke dalam beberapa kelompok eksklusif atau *cluster*. Intinya adalah memisahkan data per kelompok dengan kelompok lain nya.



Gambar 2.3 *Partitional Clustering*

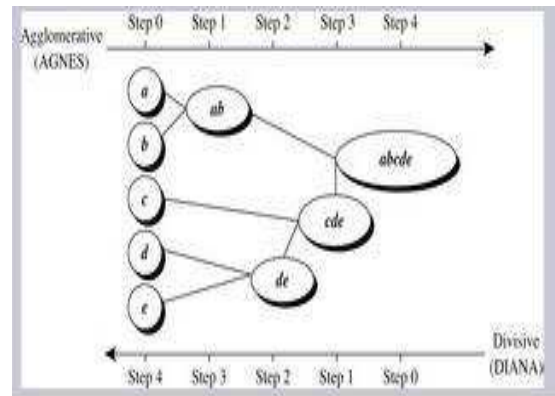
Sumber : Jiawei, H. (2011:p449). *Data Mining : Concepts and Techniques*.

Metode yang paling sering digunakan dalam *partitional clustering* adalah metode *K-Mean*. Algoritma *K-mean* mendefinisikan *centroid* dari *cluster* menjadi rata-rata point dari *cluster* tersebut. Ini hasil dari langkah-langkah dalam melakukan metode *K-mean*. Langkah melakukan metode *k-means* :

- a. Tentukan jumlah *cluster* yang akan dibuat
- b. Masukkan elemen yang akan di cluster secara acak ke masing masing *cluster*
- c. Hitung *centroid* (titik tengah) pada setiap *cluster*
- d. Ukur jarak antara satu titik ke titik tengah pada masing-masing *cluster*
- e. Masukkan titik ke *centroid* terdekat
- f. Ulangi sampai *cluster* benar benar tersusun dengan baik

## 2. Hierarchical Clustering

Pengelompokan *data* berdasar hierarki nya.



Gambar 2.4 Hierarchical Clustering

Sumber : Jiawei, H. (2011:p456). *Data Mining : Concepts and Techniques*

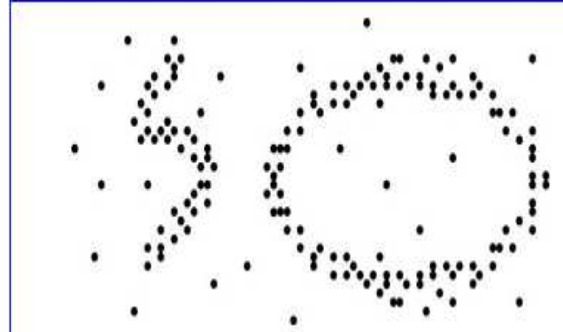
Langkah melakukan *Hierarchical clustering* :

1. Identifikasi *item* dengan jarak terdekat
2. Gabungkan *item* itu kedalam satu cluster
3. Hitung jarak antar *cluster*
4. Ulangi dari awal sampai semua terhubung

### 3. *Density-Based*

Metode partitioning dan hierarchial adalah dirancang untuk menemukan *spherical-shaped cluster*. Metode tersebut memiliki kesulitan untuk menemukan cluster berbentuk sembarang seperti bentuk “S” dan *cluster oval* seperti terlihat pada gambar 2.5. Untuk hal tersebut dengan menggunakan metode diatas kemungkinan besar tidak akurat di mana kebisingan atau *outlier* termasuk dalam *cluster*. Untuk menemukan *cluster* berbentuk sembarang, sebagai alternatif, kita dapat memodelkan *cluster* kedalam beberapa bagian dalam data *space*, yang dipisahkan dari bagian yang jarang. Ini adalah strategi utama di balik kepadatan metode berbasis *clustering*, yang

dapat menemukan *cluster* berbentuk *nonsphercial*.



Gambar 2.5 *Arbitrary Shape*

#### 4. *Grid-Based*

Metode *clustering* yang dibahas sejauh ini adalah metode yang mempartisi set dari *objek* dengan distribusi objek di *embedding space*. Pendekatan *clustering grid-based* menggunakan *grid multiresolusi* struktur data. Ini membagi objek *space* ke dalam jumlah yang terbatas dari struktur *grid* di mana operasi untuk *clustering* dilakukan. Keuntungan dari pendekatan ini adalah waktu proses yang cepat, yang biasanya tergantung dari jumlah objek data, namun tergantung pada jumlah sel dalam setiap dimensi dalam *quantized space*.



#### 2.2.3.3.4 Penggunaan Teknik *Clustering*

*Clustering* banyak digunakan pada berbagai bidang aplikasi seperti :

1. *Business intelligence*
2. *Image pattern recognition*
3. *Web search*
4. *Biology*
5. *Security*
6. *Economy*

Contoh aplikasi *data mining* yang menggunakan teknik *clustering* :

a. *Business intelligence*

*Clustering* dapat digunakan untuk mengorganisir pelanggan dalam jumlah besar ke dalam kelompok yang memiliki banyak persamaan. Hal ini membantu dalam proses CRM.

b. *Web Search*

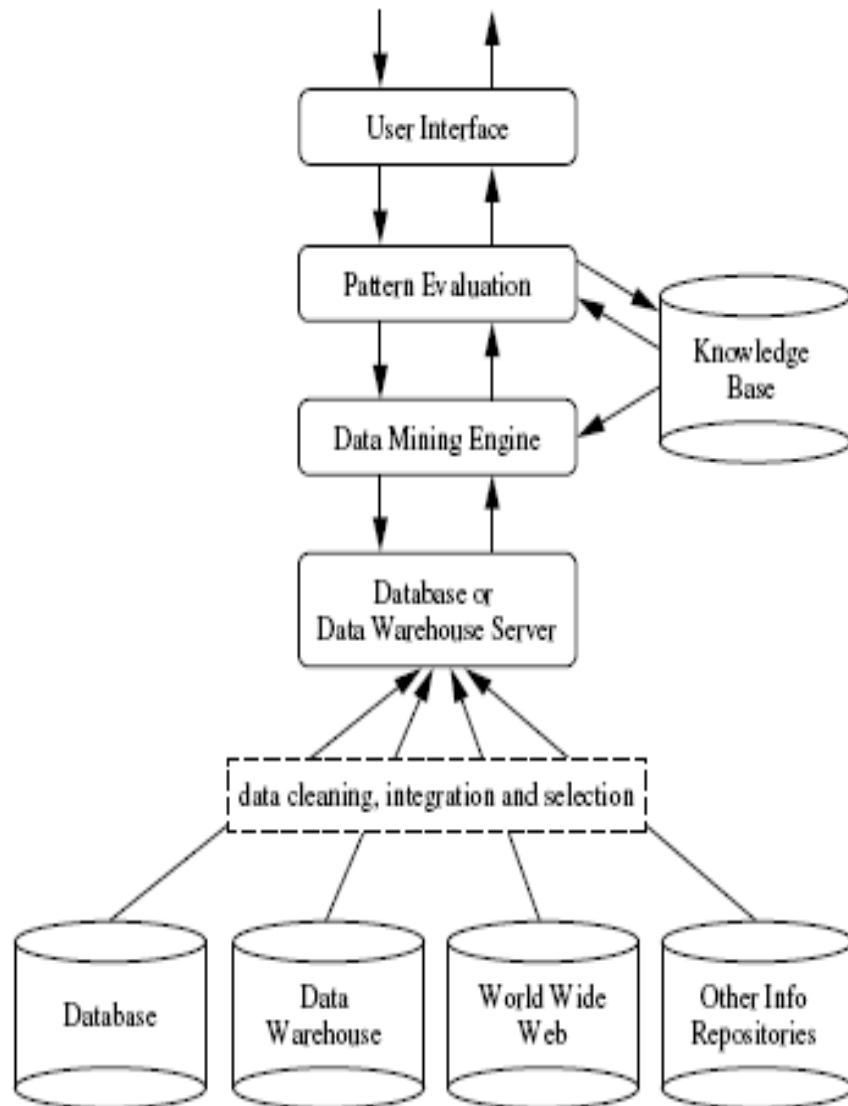
*Clustering* digunakan pada saat pencarian menggunakan *keyword*. Karena sangat banyaknya jumlah *website* yang ada, *clustering*

dapat digunakan untuk mengorganisir hasil pencarian ke dalam beberapa kelompok dan menyajikan hasil yang lebih mudah ditelusuri.

c. *Marketing*

Untuk mengelompokkan *customer* yang memiliki ke unikan dan mengembangkan program target *marketing* terhadap beberapa *customer* tersebut.

#### 2.2.4 Arsitektural Data Mining



Gambar 2.2.3 Arsitektural *Data Mining*

Arsitektur *Data Mining* memiliki beberapa komponen, sebagai berikut:

1. *Knowledge base*

Komponen pengetahuan yang digunakan untuk memandu pencarian atau mengevaluasi pola-pola yang menarik.

2. *Data Mining Engine*

Merupakan hal yang paling penting di dalam *data mining system*. Merupakan suatu *set* modul fungsi untuk tugas-tugas seperti *karakterisasi*, *association*, dan *correlation analysis*.

3. *Pattern evaluation module*

Komponen yang mempunyai tugas untuk mengevaluasi dan mengukur ketertarikan dan mengfokuskan pada interaksi modul *data mining* akan pola-pola yang akan di analisa.

4. *User Interface*

Modul yang mengkomunikasikan *user* dan *data mining system* yang memungkinkan *user* berinteraksi dengan sistem dengan cara menspesifikasi tugas *data mining*.

## 2.3 Saham

Saham adalah bentuk simbol kepemilikan modal seseorang dalam sebuah perusahaan. Hal ini didukung oleh teori dari Joko Salim (2010:p5) yang menyatakan bahwa saham adalah bentuk penyertaan modal dalam sebuah perusahaan

Berikut ini merupakan beberapa jenis saham :

1. Saham biasa (*common stocks*)

Memiliki saham biasa berarti kita memiliki bagian kepemilikan pada perusahaan tersebut.

Saham biasa sendiri dibagi menjadi beberapa jenis, diantaranya :

*a. Blue chips stocks*

Saham yang diterbitkan oleh perusahaan besar dan terkenal sehingga memperlihatkan kemampuan untuk memperoleh keuntungan

*b. Growth stocks*

Saham yang diterbitkan oleh perusahaan yang pertumbuhan penjualan dan pendapatan laba mengalami pertumbuhan yang lebih cepat dibandingkan industry lainnya.

*c. Emerging growth stocks*

Saham yang perusahaan yang relatif kecil namun memiliki daya tahan yang kuat meskipun di saat kondisi ekonomi kurang baik atau kurang mendukung.

*d. Income stocks*

Saham yang membayar deviden lebih dari total rata-rata pendapatan.

*e. Cyclical stocks*

Saham yang tidak terpengaruh oleh kondisi ekonomi.

## 2. Saham preferen (*preferred stock*)

Saham preferen merupakan gabungan dari obligasi dan saham biasa, di mana pemilik saham preferen bisa mendapat penghasilan tetap (seperti pada bunga obligasi), tetapi bisa juga tidak mendapat hasil.

Saham preferen sendiri dibagi menjadi beberapa jenis, diantaranya:

### *a. Convertible preferred stocks*

Jenis saham preferen yang dapat ditukar menjadi saham biasa dengan perbandingan yang sudah ditentukan

### *b. Callable preferred stocks*

Saham preferen yang memberikan hak kepada perusahaan yang menerbitkan saham ini untuk membeli kembali saham ini suatu saat dengan nilai tertentu

### *c. Floating rate preferred stocks*

Saham preferen yang memiliki dividen terikat pada tingkat bunga sekuritas pemerintah.

## 2.4 Kliring

Berdasarkan website [www.KPEI.co.id](http://www.KPEI.co.id) , pengertian kliring adalah proses penentuan hak dan kewajiban yang timbul dari transaksi bursa.

Kliring ditentukan dengan 2 cara, yaitu

### 1. *Netting*

Kliring dilakukan setelah user selesai transaksi dalam satu periode, misalnya setiap beberapa jam atau setiap beberapa transaksi.

### 2. Per transaksi

Kliring diadakan setiap Anggota Kliring selesai melakukan transaksi pembelian atau penjualan.

Kliring sendiri dilakukan atas transaksi bursa yang berbentuk Equity, Efek bersifat utang, dan Derivatif. Anggota kliring adalah anggota bursa efek yang memenuhi ketentuan dan persyaratan KPEI untuk mendapatkan layanan jasa Kliring dan Penjaminan Penyelesaian Transaksi Bursa

## 2.5 **Gagal Serah Dana**

Berdasarkan website [www.KPEI.co.id](http://www.KPEI.co.id), , pengertian Gagal Serah Dana adalah Tidak dipenuhinya kewajiban serah dana termasuk kewajiban serah efek yang diganti dengan serah uang.

Gagal Serah Dana dapat terjadi karena:

1. Pihak penjual tidak memiliki saham sejumlah yang dijual
2. Pihak pembeli tidak memiliki uang sejumlah nilai pembelian yang akan dilakukan

Jika terjadi gagal bayar, KPEI menjalankan mekanisme penjaminan sebagai berikut :

1. Suspensi perdagangan Anggota Kliring

2. Pencairan fasilitas kredit dari bank pembayaran untuk meng-*cover* kegagalan
3. Eksekusi *collateral* Anggota Kliring gagal seperti :
  - a. *Minimum cash collateral*
  - b. Setara *cash* (Time deposit, BG (Bilyet Giro), SBI (Sertifikat Bank Indonesia))
  - c. *Non Cash collateral* (Saham, obligasi)
4. Eksekusi saham bursa
5. Penggunaan Dana Jaminan Transaksi Bursa
6. Permohonan pengajuan kepailitan anggota kliring yang bersangkutan

## 2.6 OOAD (Object Oriented Analysis Design)

### 2.6.1 Definisi OOAD :

Menurut (Fatta, 2007) *Object Oriented Analysis Design* merupakan teknik yang mengintegrasikan data dan proses yang disebut objek

*Object-Oriented Analysis* (OOA) adalah semua jenis objek yang melakukan pekerjaan dalam sistem dan menunjukkan interaksi pengguna apa yang dibutuhkan untuk menyelesaikan tugas tersebut. *Object* diartikan suatu hal dalam sistem computer yang dapat merespon pesan (Satzinger, 2009, p60).

*Object-Oriented Design* (OOD) adalah semua jenis objek yang diperlukan untuk berkomunikasi dengan orang dan perangkat dalam



sistem, menunjukkan bagaimana objek berinteraksi untuk menyelesaikan tugas, dan menyempurnakan definisi dari masing-masing jenis objek sehingga dapat diimplementasikan dengan bahasa tertentu atau lingkungan (Satzinger, 2009, p60).

*Object-Oriented Programming* (OOP) menuliskan laporan dalam bahasa pemrograman untuk mendefinisikan apa yang setiap jenis objek ini termasuk pesan bahwa pengirim satu sama lain (Satzinger, 2009, p60).

### **2.6.2 Unified Modeling Language (UML)**

Menurut Satzinger (2009, p48), *Unified Modeling Language*(UML) merupakan suatu *set* standar konstruksi model dan notasi dikembangkan secara khusus untuk pengembangan berorientasi objek

*Unified Modelling Language* (UML) merupakan suatu bahasa pemodelan untuk membuat, mendokumentasikan, menggambarkan sistem informasi (Rama & Frederick, 2008)

### **2.6.3 The Systems Development Life Cycle (SDLC)**

Menurut Satzinger (2009, p39), *Systems Development Life Cycle* (SDLC) adalah seluruh proses yang membangun, menyebarkan, menggunakan, dan memperbarui sistem informasi.

*Sytems Development Life Cycle* (SDLC) merupakan pengembangan *system* dari pertama kali dipelajari dan digunakan hingga sistem tersebut mengalami pembaharuan, peningkatan atau tergantikan dengan sistem yang lebih baik (Morley & Parker, 2010)

#### **2.6.3.1 Tahapan dalam *Systems Development Life Cycle* (SDLC)**

Dalam *Systems Development Life Cycle* (SDLC) terdapat enam tahapan yaitu :

##### *Tahap 1 : Project Planning Phase*

Tahapan ini untuk mengidentifikasi cakupan dari sistem baru, dan memastikan bahwa *project* tersebut layak menggunakan sistem baru, sumber-sumber data untuk *project* tersebut dan batasan budget nya.

##### *Tahap 2: Analysis Phase*

Di Tahapan ini akan dilakukan pemahaman terutama terhadap dokumen kebutuhan bisnis secara rinci dan juga persyaratan pengelolaan sistem yang baru.

##### *Tahap 3 : Design Phase*

Untuk merancang sistem solusi berdasarkan persyaratan yang ditetapkan dan keputusan yang dibuat selama analisis.

##### *Tahap 4: Implementation Phase*

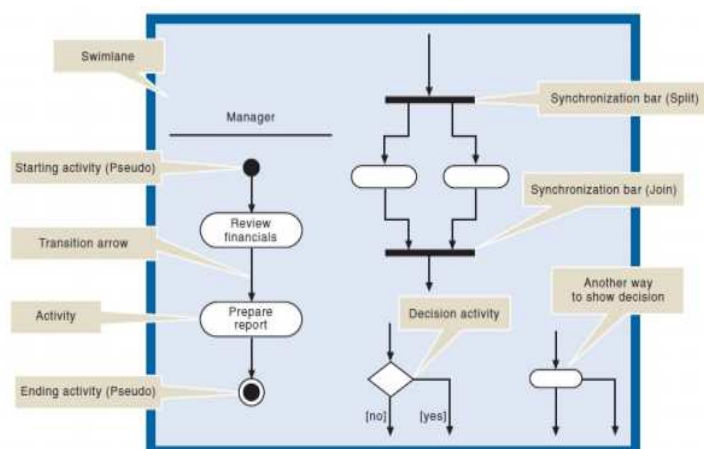
Untuk membangun tes dan memasang sebuah sistem yang dapat dipercaya dilengkapi dengan *user* yang terlatih dan siap untuk mendapatkan keuntungan seperti yang diharapkan sebelumnya dari penggunaan sistem tersebut.

#### Tahap 5 : *Support Phase*

Untuk menjaga sistem agar berjalan secara produktif dari awal dibangun sistem tersebut hingga bertahun-tahun sampai di mana masa hidup sistem tersebut berakhir.

### 2.6.4 Activity Diagram

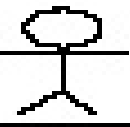

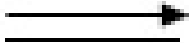
Menurut Satzinger (2009, p144), sebuah *activity diagram* hanyalah sebuah diagram alur kerja yang menggambarkan berbagai pengguna kegiatan, orang yang melakukan aktivitas masing-masing, dan aliran sekuensial kegiatan ini atau dapat dikatakan diagram yang menggambarkan alur proses bisnis.

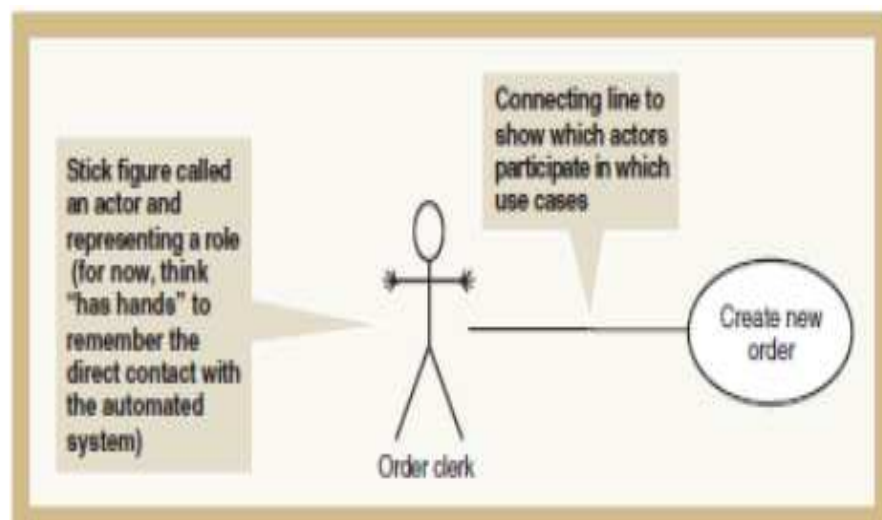


Gambar 2.6.4 Simbol *Activity Diagram*

### 2.6.5 Use Case

*Use Case* merupakan model fungsional yang di dalamnya terdapat actor dan use case itu sendiri, actor disini merupakan orang-orang yang menjalankan aktifitasnya dan berhubungan langsung dengan sistem, sedangkan *use case* itu sendiri berisi pekerjaan yang dilakukan *actor* yang menggunakan sistem. Berikut merupakan simbol yang digunakan dalam *Use Case Diagram* :

Simbol	Keterangan
	Actor
	Proses
	(hubungan)



Gambar 2.6.5 *Use Case*

### 2.6.6 Event Table

*Event table* adalah sebuah tabel yang meliputi baris dan kolom, yang berisi beberapa komponen yang mewakili peristiwa dan rincian dari peristiwa mereka, masing-masing.

Berikut penjelasan komponen dari *event table* antara lain :

a. *Event*

Merupakan katalog *use case* daftar peristiwa dalam baris dan potongan kunci informasi tentang setiap peristiwa dalam kolom.

b. *Trigger*

Merupakan sinyal yang memberitahukan sistem bahwa suatu peristiwa telah terjadi, baik kedatangan membutuhkan pengolahan data atau titik waktu.

c. *Source*

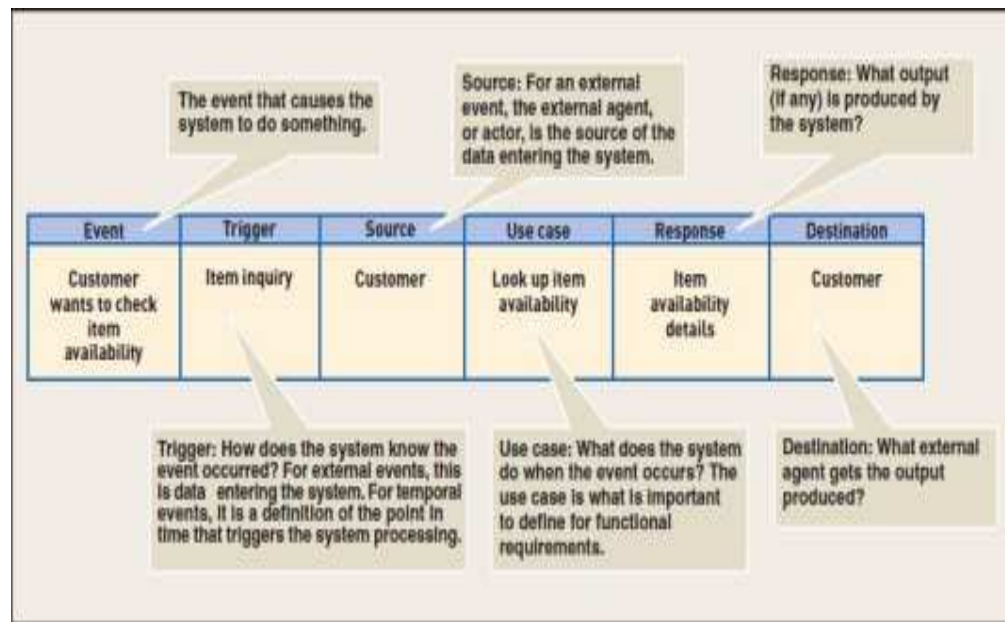
Merupakan agen eksternal atau *actor* yang memasok data ke sistem.

d. *Response*

Merupakan *output* yang dihasilkan oleh sistem, yang menuju ke tujuan.

e. *Destination*

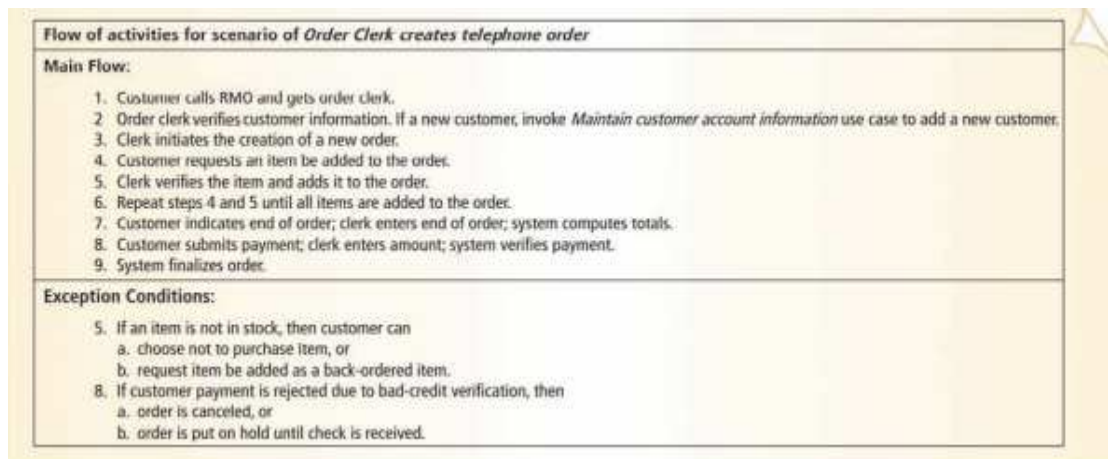
Merupakan agen eksternal atau *actor* yang menerima data dari sistem.



Gambar 2.6.6 Event Table

## 2.6.7 Use Case Description

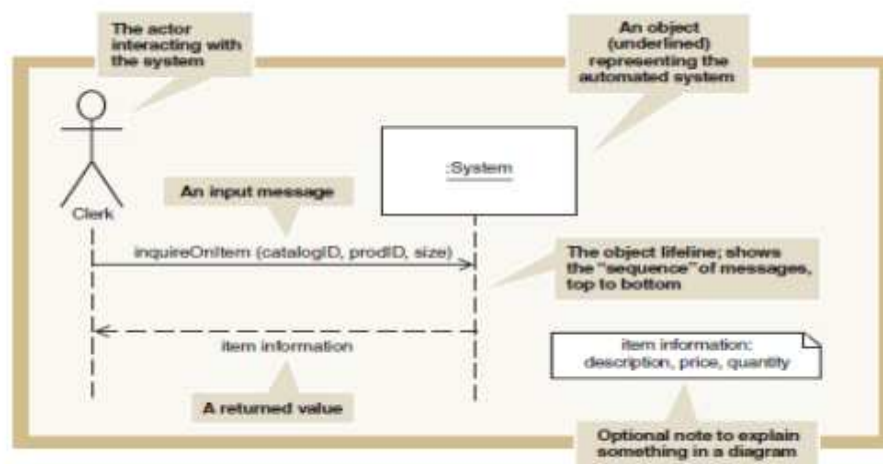
Deskripsi atau penjelasan mengenai *Use Case* yang berisi nama *use case* nya, *main flow*. Di sini dijelaskan tentang bagaimana *actor* yang menggunakan *system* yang telah digambarkan di *Use Case*.



Gambar 2.6.7 Use Case Description

### 2.6.8 System Sequence Diagram (SSD)

*System Sequence Diagram* (SSD) biasanya digunakan dalam hubungannya dengan deskripsi menggunakan kasus untuk membantu dokumen rincian kasus penggunaan tunggal atau skenario dalam kasus penggunaan atau menangkap interaksi antara sistem dan eksternal *entity* yang direpresentasikan oleh *actor*.



Gambar 2.6.4 *System Sequence Diagram*