BAB II

TINJAUAN PUSTAKA

II.1 Tinjauan Tempat Penelitian

Pada sub bab ini menjelaskan bahwa tempat penelitian yang telah dilakukan peninjauan adalah Pusat Penelitian Informatika – LIPI.

II.1.1 Sejarah Perusahaan

Pusat Penelitian Informatika didirikan dengan surat Keputusan Kepala LIPI No 1151/M/2001 tanggal 5 Juni 2001. Semula bernama Pusat Penelitian dan Pengembangan Informatika dan Ilmu Pengetahuan Komputer didirikan pada tanggal 13 Januari 1986.

Berdirinya Pusat Penelitian Informatika tidak dapat dipisahkan dengan sejarah terbentuknya Lembaga Elektroteknika Nasional (LEN). Lembaga Elektroteknika Nasional sebagai lembaga penelitian dalam bidang elektronika diresmikan pada tanggal 10 Juni 1965. Bermula di kampus ITB dengan ruang seluas 30 meter persegi yang merupakan pinjaman dari ITB. Pada tahun 1967, berdasarkan Keputusan Presiden nomor 128 tahun 1967, Lembaga Elektroteknika Nasional (LEN) bernaung dibawah koordinasi Lembaga Ilmu Pengetahuan Indonesia. Pada tahun 1969, Prusahaan Negara Pertambangan Minyak dan Gas Bumi Nasional (Pertamina), memberikan pinjaman gedung seluas 1000 meter persegi di Jl. Sawunggaling No. 14 Bandung, sehingga ruangan dikomplek ITB ditinggalkan. Pada awal tahun 1978, komplek gedung perkantoran dan laboratorium LEN-LIPI sudah tersebar di 9 lokasi di kota Bandung, dengan luas lantai sekitar 2500 meter persegi.

Tahun 1983 LEN-LIPI pindah ke Jl. Soekarno-Hatta No.407 250A Bandung, serta gedung tersebut diresmikan pemakaiannya oleh Presiden Republik Indonesia, Bapak Soeharto. pada tanggal 31 Mei 1983. Dengan dilakukannya reorganisasi di

Lembaga Ilmu Pengetahuan Indonesia (LIPI) berdasarkan Keppres No. l tahun 1986 tanggal 13 Januari 1986, Lembaga Elektroteknika Nasional berkembang menjadi tiga Puslitbang dan satu Unit Pelaksana Teknis, yaitu:

a. Pusat Penelitian dan Pengembangan Telekomunikasi, Elektronika Strategis, Komponen dan Material (Puslitbang Telkoma-LIPI).
b. Pusat Penelitian dan Pengembangan Informatika dan Ilmu Pengetahuan Komputer (Puslitbang Inkom LIPI).
c. Pusat Penelitian dan Pengembangan Tenaga Listrik dan Mekatronika (Puslitbang Telimek-LIPI)
d. UPT Pusat Laboratorium Enjiniring Nasional (UPT Pusat LEN LIPI).

Dalam struktur organisasi LIPI, keempat Puslibang tersebut diatas berada dibawah koordinasi Deputi Bidang Ilmu Pengetahuan Teknik -LIPI.

Pada tanggal 8 Maret 1990 seluruh fasilitas UPT LEN-LIPI diserahkan ke BPIS. Kantor Puslitbang Inkom LIPI pindah ke Jl. Ranggamalela No.11 Bandung, sedangkan laboratoriumnya menempati salah satu bagian gedung di kompleks LIPI Jl. Cisitu 21/154D Bandung. Pada tahun 2001 dalam Reorganisasi baru Lembaga Ilmu Pengetahuan Indonesia, Pusat Penelitian dan Pengembangan Informatika dan Ilmu Pengetahuan Komputer berubah nama dan struktur organisasinya menjadi Pusat Penelitian Informatika LIPI.

II.1.2 Logo Perusahaan

Makna Logo:

1. Bentuk bulatan merepresentasikan keselarasan dan melambangkan keutuhan.
2. Bentuk menyerupai huruf Q, direpresentasikan dengan lingkaran bergaris-garis dan ada tonjolan di sebelah kanan bawah, melambangkan "Quality" (Mutu), sebagai salah satu dari bagian dari AMTeQ.
3. Bentuk menyerupai huruf T, direpresentasikan dengan satu batang berwarna putih mendatar dan batang putih miring atas ke bawah, melambangkan "Testing" (Pengujian), sebagai salah satu dari bagian dari AMTeQ.

4. Garis-garis mendatar berselang-seling warna biru dan putih menunjukkan bahwa walaupun AMTeQ adalah bulat selaras dan utuh, tetapi terdapat spesialisasi/segmentasi dalam bidang mutu dan pengujian; garis-gari mendatar berselang-seling dari bawah ke atas juga melambangkan penjenjangan bahwa proses perbaikan berkelanjutan harus terus meningkat ke arah yang lebih baik



Gambar.II.1 Logo Perusahaan

II.1.3 Visi dan Misi Perusahaan

Adapun visi dan misi dari Pusat Penelitian(PUSLIT) Informatika-LIPI adalah:

Visi :

Pusat Penelitian Informatika mengabdi untuk kepentingan masyarakat dan menyelenggarakan kegiatan di bidang Informatika dan Ilmu Pengetahuan Komputer untuk pembangunan berkelanjutan.Maknanya bahwa berbagai kegiatan penelitian yang dilakukan di bidang Informatika adalah untuk kepentingan masayarakat, antara lain untuk mencerdaskan kehidupan bangsa dengan menggunakan kemampuan teknologi informasi yang baru. Teknologi ini digunakan untuk pengumpulan,

pengolahan, penyimpanan dan penyebarluasan data atau informasi kepada masyarakat dengan cepat, tepat waktu dan akurat. Sehingga akan terjadi percepatan peningkatan kemampuan masyarakat dalam pembangunan yang berkelanjutan.

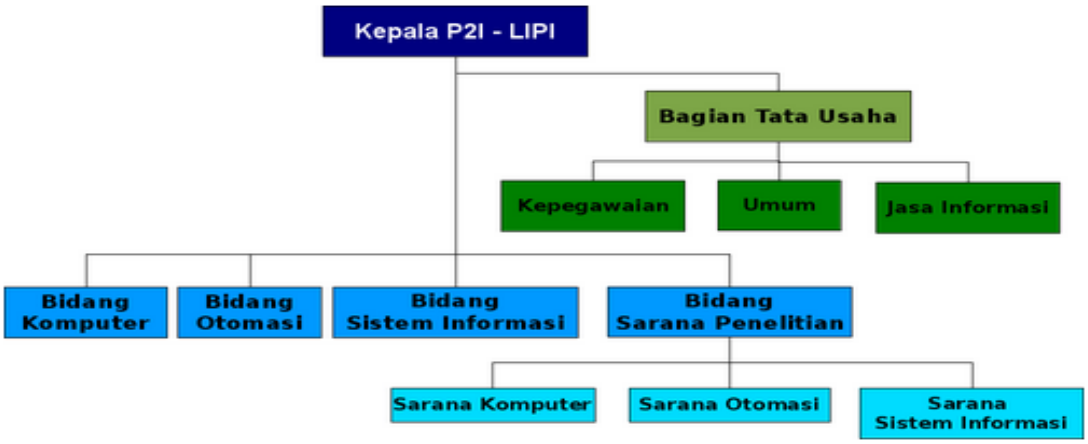
Misi :

Pusat Penelitian Informatika melaksanakan kegiatan penelitian, pengembangan, pelayanan jasa dan saran kebijaksanaan di bidang Informatika dan Ilmu Pengetahuan Komputer. Informatika adalah suatu ilmu yang sifatnya problem solving, artinya, digunakan untuk menyelesaikan suatu masalah tertentu yang berhubungan dengan data baik data tekstual, image, suara atau data spatial. Kemudian data ini diproses sesuai kebutuhan dengan menggunakan teknologi informasi yang baru, antara lain teknologi komputer, telekomunikasi dan jaringan, untuk menghasilkan informasi yang berkualitas yaitu informasi yang tepat waktu, relevan dan akurat.

II.1.4 Struktur Organisasi Perusahaan

Struktur organisasi dalam suatu institusi merupakan hal yang sangat penting, dengan adanya struktur organisasi ini memberikan pembagian tugas sesuai dengan bidangnya masing-masing.

Adapun struktur organisasi di Pusat Penelitian Informatika – LIPI adalah sebagai berikut :



Gambar.II.2 Struktur Organisasi Pusat Penelitian Informatika – LIPI

II.2 Landasan Teori

Landasan teori ini akan menjelaskan teori – teori yang digunakan dalam pembangunan sistem ini.

II.2.1. Data Mining

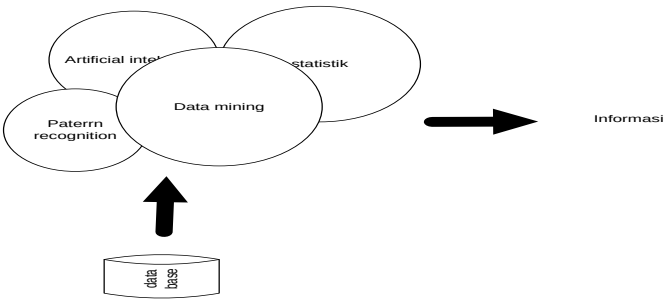
Data mining, sering juga disebut *knowledge discovery in database* (KDD), adalah kegiatan yang meliputi pengumpulan, pemakaian data historis untuk menemukan keteraturan, pola atau hubungan dalam set data berukuran besar. Keluaran dari data mining ini bisa dipakai untuk memperbaiki pengambilan keputusan di masa depan. Sehingga istilah *pattern recognition* sekarang jarang digunakan karena ia termasuk bagian dari data mining **[2]**.

Data mining diartikan sebagai suatu proses ekstraksi informasi berguna dan potensial dari sekumpulan data yang terdapat secara implisit dalam suatu basis data. Banyak istilah lain dari data mining yang dikenal luas seperti knowledge mining from databases, knowledge extraction, data archeology, data dredging, data analysis dan lain sebagainya **[2]**.

Data mining adalah proses yang menggunakan teknik statistic, perhitungan, kecerdasan buatan dan *machine learning* untuk mengekstraksi dan mengidentifikasi informasi yang bermanfaat dan pengetahuan terkait dari berbagai basis data besar **[1]**.

Dari pengertian mengenai *data mining* di atas maka dapat disimpulkan bahwa data mining adalah salah satu ilmu yang digunakan untuk ekstraksi datasuatu pengetahuan dari tumupkan data tersebut dengan menggunakan sebuah metode yang berlaku pada *data mining*. Dengan konsep *data mining* dapat membantu dalam mengambil sebuah keputusan dan kebijakan suatu instansi.

Data mining adalah satu ilmu di bidang komputer yang cukup banyak penerapanya, ditunjang kekayaan dan keanekaragaman berbagai bidang ilmu lainya seperti *artificial intelligence*, *database*, statistik, pemodelan matematika pengolahan citra, dan sebagainya, sehingga penerapan *data mining* menjadi kasus semakin luas.



Gambar.II.3 Akar ilmu data mining

Pada umumnya sistem *data mining* terdiri dari komponen – komponen berikut **[4]**:

a. *Database, data warehouse*

Media dalam ha ini bisa jadi berupa *database, data warehouse, spreadsheet,* atau jenis – jenis penampungan informasi lainnya. *Data cleaning* dan *data integration* dapat dilakukan pada data tersebut.

b. *Database,* atau *data warehouse server*

Database atau *data warehouse server* bertanggungjawab untuk menyediakan data yang relevan berdasarkan permintaan dari *user* pengguna *data mining*.

c. Basis pengalaman (*knowladge base*)

Merupakan basis pengetahuan yang digunakan sebagai panduan dalam pencarian pola.

d. *Data mining engine*

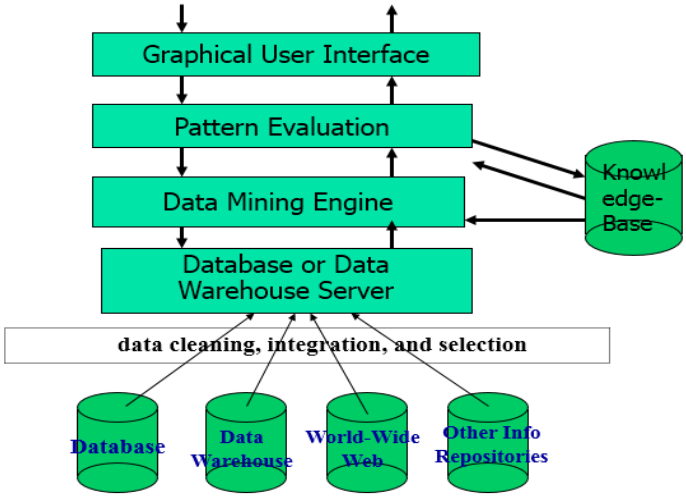
Bagian dari *software* yang menjalankan program berdasarkan algoritma yang ada.

e. Pattern evaluation module

Bagian dari *software* yang berfungsi untuk menemukan *paterrn* atau pola –pola yang yang terdapat di dalam *database* yang diolah sehingga nantinya proses *data mining* dapat menentukan *konwladge* yang sesuai.

f. Graphical user interface

Bagian ini merupakan sarana antara *user* dan sistem *data mining* untuk berkomunikasi, dimana *user* dapat berinteraksi dengan sistem melalui *data mining query,* untuk menyediakan informasi yang dapat membantu *user* untuk melakukan *browsing* pada *databse* dan *data warehouse,* mengevaluasi *pattern* yang telah dihasilkan, dan menampilkan *pattern* tersebut dengan tampilan yang berbeda – beda.



Gambar.II.4 Arsitektur data mining

Pada dasarnya data mining berhubungan dengan analisa data dan penggunaan teknik-teknik perangkat lunak untuk mencari pola dan keteraturan dalam himpunan data yang sifatnya tersembunyi. Data mining diartikan sebagai suatu proses ekstraksi informasi berguna dan potensial dari sekumpulan data yang terdapat secara implisit dalam suatu basis data. Tantangan-tantangan dalam Data Mining meliputi : penanganan berbagai tipe data, efisiensi dari algoritma data mining, kegunaan, kepastian dan keakuratan hasil, ekspresi terhadap berbagai jenis hasil dan data yang diambil dari berbagai sumber yang berbeda. Tahapan dalam Data Mining meliputi : proses seleksi, pembersihan data, tranformasi, implementasi teknik data mining dan interprestasi hasil. Fungsionalitas dalam Data Mining meliputi mining association rule, karakterisasi data multilevel, klasifikasi data, analisa cluster, dan pencarian pola sekuensial. Teknik-teknik dalam Data Mining yang bisa diterapkan antara lain : *market basket,* analysis dan Algoritma Apriori. **[2]**

II.2.2. Clustering

Clustering termasuk ke dalam *descriptive methods,* dan juga termasuk *unsupervised learning* dimana tidak ada pendefinisian kelas objek sebelumnya. Sehingga *clustering* dapat digunakan untuk menentukan label kelas bagi data – data yang belum diketahui kelasnya. Konsep dasar dari *clustering* adalah mengelompokkan sejumlah objek ke dalam *cluster* dimana *cluster* yang baik adalah *cluster* yang memiliki tingkat kesamaan atau *similarity* yang tinggi antar objek di dalam satu *cluster* dan tingkat ketidaksamaan atau *dissimilarity* yang tinggi dengan objek *cluster* yang lainnya. Aplikasi *clustering* bisa dibangun sebagai *stand-alone tool* atau sebagai *preprocessing* algoritma lainnya, misalnya untuk algoritma pendeteksian *outlier*. Aplikasi *clustering* bisa dibangun sebagai*stand-alone tool* atau sebagai *preprocessing* algoritma lainnya, misalnya untuk algoritma pendeteksian *outlier* dalam pembahasan tugas akhir ini. Aplikasi *clustering* juga dapat di implementasikan untuk *pattern recognition, spatial data analysis, image processing,* klasifikasi dokumen dan *market research.* Terdapat banyak algoritma *clustering* yang dalam penggunaannya tergantung pada tipe data yang akan dikelompokkan dan apa tujuan dari pembuatan aplikasinya. Dalam pembahasan di sini, algoritma *clustering* digunakan untuk mengelompokkan objek ke dalam cluster – cluster, kemudian dari hasil clustering akan dideteksi keberadaan *outlier* dalam data tersebut.

Algoritma clustering diklasifikasikan ke dalam 5 kategori, yaitu :

1. *Partitioning methods*

Pengelompokkan objek dimana tiap objek terpartisi dengan tepat, yang berarti 1 objek dimiliki oleh 1 klaster. Yang termasuk ke dalam metode ini adalah algoritma *k-means, k-medoid* atau *PAM, CLARA,* dan C*LARANS.*

2. *Hierarchical methods*

Pengelompokan objek dapat dilakukan dengan 2 cara, *agglomerative* yang dimulai dengan menggabungkan beberapa cluster hingga menjadi satu, atau *divisive* yang dimulai dengan *cluster* yang sama kemudian dipecah menjadi beberapa *cluster*

yang lebih kecil. Yang termasuk ke dalam metode ini adalah algoritma *CURE*, *BIRCH,* dan *Chameleon.*

3. *Density-based methods*

Pengelompokkan objek berdasarkan tingkat kerapatan objek atau densitas.Yang termasuk ke dalam metode ini adalah algoritma *DBSCAN, DENCLUE,* dan *OPTICS.*

4. *Grid-based methods*

Pengelompokan objek dengan menggunakan struktur data *grid* multi resolusi. Mampu untuk menangani data berdimensi tinggi. Yang termasuk ke dalam metode ini adalah algoritma *CLIQUE, Wave Cluster,* dan *STING.*

5. *Model-based methods.*

Pengelompokan objek dengan memodelkan tiap *cluster*, dan mencoba mengoptimasikan kesesuaian data dengan model matematika. Yang termasuk ke dalam metode ini adalah algoritma *COBWEB.*

II.2.3. Text Mining

Text Clustering adalah proses unsupervised learning (proses pembelajaran sendiri) yang pengelompokkan kumpulan dokumen berdasarkan hubungan kemiripannya dan memisahkannya ke dalam beberapa kelompok. **[5]**

II.2.6.1. Preprocessing

Preprocessing merupakan pemrosesan awal dokumen agar diperoleh suatu nilai yang dapat dipelajari oleh sistem clustering.

II.2.6.2. Case Folding

Case folding merupakan suatu tahap yang mengubah huruf besar menjadi huruf kecil.

II.2.6.3. Tokenization

Tokenization adalah proses pemotongan seluruh urutan karakter menjadi satu potongan kata**[5].**

II.2.6.4. Stopword Removal

Stopword removal merupakan proses penghapusan semua kata yang tidak memiliki makna**[5].**

II.2.6.5. Stemming

Stemming adalah proses membentuk suatu kata menjadi kata dasarnya. Algoritma stemming yang digunakan dalam sistem pengelompokkan ini adalah algoritma Nazief – Adriani**[4].**

II.2.6.6. Term Weighting

Term weighting merupakan proses pemberian bobot suatu token dalam suatu term.

II.2.6.7. Term Frequency

Term Frequency (TF) adalah pembobotan yang menghitung frekuensi kemunculan sebuah token pada suatu dokumen

$TF(t_k, d_j) = f(t_k, d_j)$ [1]……(5)

II.2.6.8. Document Frequency

Document Frequency (DF) adalah pembobotan yang menghitung frekuensi kemunculan sebuah token pada kumpulan dokumen

II.2.6.9. Pembobotan TF.IDF

Pembobotan TF • IDF adalah suatu pengukuran statistik untuk mengukur seberapa penting sebuah token dalam kumpulan dokumen

TF • IDF (t k , d j) = TF (t k , d j) • IDF (t k) [1]……………………………..(5)

II.2.4. Hierarchical *clustering*

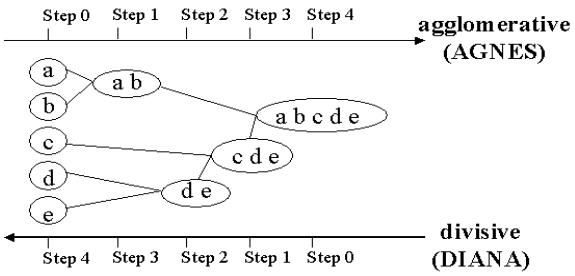
Pada algoritma *clustering*, *data* akan dikelompokkan menjadi *cluster-cluster* berdasarkan kemiripan satu *data* dengan yang lain. Prinsip dari *clustering* adalah memaksimalkan kesamaan antar anggota satu *cluster* dan meminimumkan kesamaan antar anggota *cluster* yang berbeda [4].

Kategori algoritma *clustering* yang banyak dikenal adalah *Hierarchical Clustering*. *Hierarchical Clustering* adalah salah satu algoritma *clustering* yang dapat digunakan untuk meng-*cluster* dokumen *(document clustering*). Dari teknik *hierarchical clustering*, dapat dihasilkan suatu kumpulan partisi yang berurutan, dimana dalam kumpulan tersebut terdapat:

a. *Cluster – cluster* yang mempunyai poin – poin individu. *Cluster – cluster* ini berada di level yang paling bawah.

b. Sebuah *cluster* yang didalamnya terdapat poin – poin yang dipunyai semua *cluster* didalamnya. *Single cluster* ini berada di *level* yang paling atas.

Hasil keseluruhan dari algoritma *hierarchical clustering* secara grafik dapat digambarkan sebagai *tree*, yang disebut dengan *dendogram*. *Tree* ini secara grafik menggambarkan proses penggabungan dari *cluster – cluster* yang ada, sehingga menghasilkan *cluster* dengan level yang lebih tinggi [9]. Gambar 2.6 adalah contoh *dendogram*.



Gambar.II.5 Dendogram

Metode ini menggunakan strategi disain *Bottom-Up* yang dimulai dengan meletakkan setiap obyek sebagai sebuah *cluster* tersendiri (*atomic cluster*) dan selanjutnya menggabungkan *atomic cluster – atomic cluster* tersebut menjadi *cluster* yang lebih besar dan lebih besar lagi sampai akhirnya semua obyek menyatu dalam sebuah *cluster* atau proses dapat pula berhenti jika telah mencapai batasan kondisi tertentu [6]. Metode *Agglomerative Hierarchical Clustering* yang digunakan pada penelitian ini adalah metode *AGglomerative NESting* (AGNES). Cara kerja AGNES dapat dilihat pada gambar 1.

Adapun ukuran jarak yang digunakan untuk menggabungkan dua buah obyek cluster adalah *Minimum Distance* [6], yang dapat dilihat pada persamaan 2.6.

$$d_{\min}(C_i, C_j) = \min_{p \in C_i, p' \in C_j} |p - p'| \quad ...(1)$$

Dimana |p – p’| jarak dua buah obyek p dan p’.

II.2.5. K-means Clustering

K-Means Clustering merupakan metode yang termasuk ke dalam golongan algoritma Partitioning Clustering.

Langkah-langkah dari metode K-Means adalah sebagi berikut : [4]

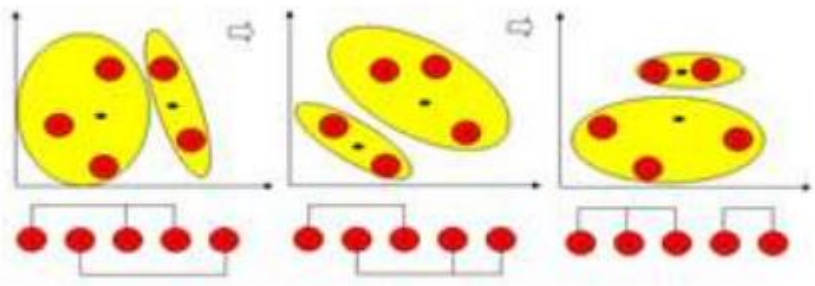
1. Tentukan nilai k sebagai jumlah cluster yang ingin dibentuk.
2. Bangkitkan k centroid (titik pusat cluster) awal secara acak.

3. Hitung jarak setiap data ke masing-masing centroid menggunakan rumus korelasi antar dua objek (Euclidean Distance).
4. Kelompokkan setiap data berdasarkan jarak terdekat antara data dengan centroidnya.
5. Tentukan posisi centroid baru (k C) dengan cara menghitung nilai rata-rata dari data yang ada pada centroid yang sama.

$$C_k = \left(\frac{1}{n_k}\right) \sum d_i \ldots\ldots\ldots (4)$$

Dimana n k adalah jumlah dokumen dalam cluster k dan d i adalah dokumen dalam cluster k.

6. Kembali ke langkah 3 jika posisi centroid baru dengan centroid lama, tidak sama.



Gambar.II.6 langkah – langkah metode K-Means[4]

II.2.6. Outliers Analysis

Sebuah sumber data atau dataset pada umumnya mempunyai nilai-nilai pada setiap obyek yang tidak terlalu berbeda jauh dengan obyek lain. Akan tetapi terkadang pada data tersebut juga ditemukan obyek - obyek yang mempunyai nilai atau sifat atau karakteristik yang berbeda dibandingkan dengan obyek pada umumnya.

Deteksi oulier adalah suatu teknik untuk mencari obyek dimana obyek tersebut mempunyai perilaku yang berbeda dibandingkan obyek-obyek pada umumnya.

Teknik data mining dapat digunakan untuk mendeteksi adanya suatu outlier pada sebuah dataset. Teknik data mining yang diganakan adalah Clustering-based, Distance-based dan Density-based.

II.2.6.1. Metode Clustering-based

Clustering merupakan salah satu teknik analisis dalam Data Mining dimana clustering melakukan pengelompokan data berdasarkan kesamaan karakteristik data. Dengan kesamaan karakteristik pada sebuah kelompok ini dapat diambil suatu informasi yang mempunyai arti dan berguna.

1. Algoritma *CLAD*

Pada *CLAD* terdapat dua fase utama yaitu pembuatan cluster dan meng-*assign* obyek – obyek data pada data set. Secara sederhana dapat dideskripsikan sebagai berikut:

1) inisialisasi cluster_outier = 0
2) fase_1
3) untuk setiap cluster_outlier hitung jarak centroid cluster dengan setiap obyek data
4) jika jarak obyek data dengan centroid cluster kurang dari lebar_cluster masukkan obyek ke dalam cluster
5) jika jarak obyek data lebih dengan centroid lebih dari lebar_cluster dan obyek data belum menjadi anggota cluster_outlier lain maka buat cluster_outlier baru dengan obyek data sebagai centroid
6) fase_2
7) untuk setiap cluster_outlier hitung jarak centroid cluster dengan setiap obyek data
8) jika jarak centroid cluster_outlier dengan obyek data kurang dari lebar cluster dan obyek data belum menjadi anggota cluster_outlier maka masukkan obyek data ke dalam cluster_outlier

2. Lebar Cluster

Lebar *cluster* dideskripsikan sebagai jangkauan antara centroid cluster_outlier dengan obyek data. Perhitungan parameter lebar cluster_dilakukan dengan mengambil sampel data dari data set kemudian dihitung jarak rata-rata.

3. Fungsi Jarak

Perhitungan jarak antara dua obyek data dilakukan dengan menggunakan fungsi Euclidan dimana fungsi ini dapat digunakan pada dimensi yang tinggi.

$$distance(Z1, Z2) = \sqrt{\sum_{i=1}^{i=|Z1|} [\mathrm{Z1}i - \mathrm{Z2}i]\,2}$$

4.* Analisis *Cluster

Penentuan bahwa suatu cluster merupakan cluster outlier, CLAD menggunakan 2 attribut pada cluster yang telah terbentuk yaitu *distance* dan *density* dari cluster lain. Dikarenakan setiap cluster memiliki lebar cluster yang tetap maka kepadatan (*density*) dari setiap cluster dihitung berdasarkan jumlah obyek yang termasuk dalam cluster tersebut. Jarak (*distance*) antar cluster dihitung dengan menggunakan *average intercluster distance* (ICD)

$$IDCi = [\sum_{j=1, j\neq i}^{j=|c|} distance(ci, cj)] \div (|c| - 1)$$

Standar deviasi yang digunakan adalah *median absolute deviation* (MAD) dikarenakan persebaran jumlah anggota cluster yang tidak merata.

$$\mathrm{MAD}(P) = median(\{p - median(P) : p \in P\})$$

Dengan menggunakan fungsi ICD dan MAD dapat diketahui apakah suatu cluster dikatakan sebagai cluster outlier. Cluster dengan label *sparse* dikatakan sebagai *local* outlier, sedangkan cluster dengan label *distant* dikatakan sebagai *global* outlier. Sebuah cluster dikatakan sebagai cluster outlier apabila cluster tersebut *distant* dan *sparse* yang merupakan gabungan dari *local* outlier dan *global* outlier.

$$\text{C distant} = \{ci \in C \mid ICDi > \text{AVG}(ICD) + \text{SD}(ICD)\}$$

$$\text{Csparse} = \{ci \in C \mid Counti > \text{AVG}(Count) - \text{MAD}(Count)\}$$

$$\text{C demse} = \{ci \in C \mid Counti > \text{AVG}(Count) + \text{MAD}(Count)\}$$

Sebuah cluster diakatakan sebagai cluster_outlier jika memiliki status *distant* dan *sparse.*

II.2.6.2. Metode Distance-based

Sebuah metode pencarian outlier yang popular dengan menghitung jarak pada obyek tetangga terdekat (*nearest neighbor*). Dalam pendekatan ini, satu obyek melihat obyek-obyek *local neighborhood* yang dedefinisikan dengan *k-nearest neighbor*. Jika ketertetanggaan antar obyek relatif dekat maka dikatakan obyek tersebut normal, akan tetapi jika ketertetanggaan antar obyek relatif sangat jauh maka dikatakan obyek tersebut tidak normal.

1. Algoritma Bay's

Algoritma Bay's mencari outlier dengan menghitung jarak antar obyek data pada dataset. Pencarian ini dilakukan dengan membandingkan jarak yang telah dihitung dengan jarak pada *k* tetangga terdekat (*k-nearest neighbor*), kemudian dipilih untuk menjadi tetangga terdekat menggantikan tetangga terdekat yang terjauh.

2. Analisis Obyek Data

Obyek data dikatakan sebagai outlier apabila obyek tersebut memiliki obyek tetangga yang sangat sedikit pada jarak tertentu dan memiliki jarak yang jauh dibandingkan dengan jarak rata-rata obyek-obyek data tetangga terdekat

II.2.6.3. Metode density-based

Metode density-based tidak secara eksplisit mengklasifikasikan sebuah obyek adalah outlier atau bukan, akan tetapi lebih kepada pemberian nilai kepada obyek sebagai derajat kekuatan obyek tersebut dapat dikategorikan sebagai outlier. Ukuran derajat kekuatan ini adalah *local outlier factor* (LOF). Pendekatan untuk pencarian outlier ini hanya membutuhkan satu parameter yaitu *MinPts*, dimana *MinPts* adalah

jumlah tetangga terdekat yang digunakan untuk mendefinisikan *local neighborhood* suatu obyek. *MinPts* diasumsikan sebagai jangkauan dari nilai MinPtsLB dan MinPtsUB Nilai MinPtsLB dan MinPtsUB disarankan bernilai 10 dan 20. Akhirnya semua obyek dalam dataset dihitung nilai LOFnya.

1. Algoritma LOF (Local Outlier Factor)

Secara sederhana algoritma LOF dapat dideskripsikan sebagai berikut:

1. menghitung jumlah tetangga terdekat
2. menghitung kepadatan lokal dari setiap obyek
3. menghitung LOF untuk setiap obyek data
4. me-*maintain* obyek-obyek data dengan nilai LOF yang tinggi

2. Analisis Obyek Data

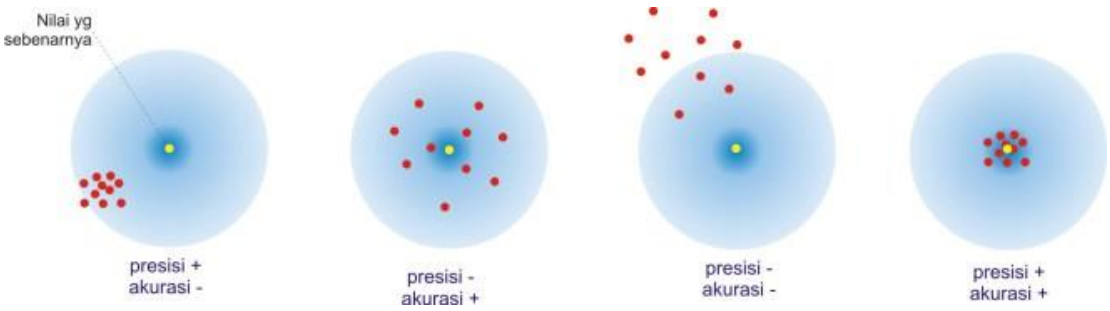
Obyek data akan dianggap memiliki nilai outlier yang tinggi jika pada jarak k tetangga terdekat memiliki kepadatan yang sangat kecil. Semakin banyak obyek – obyek tetangga dalam jarak k-tetangga terdekat obyek ini memiliki nilai LOF mendekati 1 dan tidak seharusnya diberi label sebagai outlier.

3. Pengaruh Nilai Parameter *Minpts*

Algoritma *density-based* hanya membutuhkan satu parameter yaitu *MinPts*, jumlah tetanga terdekat untuk menghitung ketertetanggaan local. obyek – obyek data didistribusikan dengan menggunakan distribusi Gaussian. Untuk setiap nilai MinPts berkisar antara 2 sampai 50, minimum, maksimum dan rata – rata nilai LOF. Karena nilai MinPts dapat berubah secara fluktuatif, maka digunakan jangkauan dari MinPts yaitu *MinPtsLB* dan *MinPtsUB* untuk mendefinisikan jangkauan terendah dan jangkauan tertinggi dari MinPts. Dengan melihat gambar 2.5 standar deviasi dari LOF hanya stabil saat MinPts mulai dari nilai 10 sampai nilai kurang dari 30.

II.2.7. Precission, Recall dan Accuracy

Dalam pengenalan pola (*pattern recognition*) dan temu kembali informasi (*information retrieval*), *precision* dan *recall* adalah dua perhitungan yang banyak digunakan untuk mengukur kinerja dari system yang digunakan. *Precision* adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Sedangkan *recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Sedangkan di dalam statistika dikenal juga istilah *accuray*. Accuracy didefinisikan sebagai tingkat kedekatan antara nilai prediksi dengan nilai aktual. Ilustrasi berikut ini memberikan gambaran perbedaan antara *accuracy* dan *precision.*



Gambar.II.7 Precission, recall dan accuracy

Secara umum precision, recall dan accuracy dapat dirumuskan sebagai berikut:

		Nilai sebenarnya	
		TRUE	FALSE
Nilai predisksi	TRUE	TP (True Positive) *Corect result*	FP (False Positive) *Unexpected result*
	FALSE	FN (False Negative) *Missing result*	TN (True Negative) *Corect absence of result*

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Salah satu penerapan prinsip relevansi yang sejak dahulu digunakan dalam pengembangan sistem IR adalah penggunaan ukuran *recall and precision*. Sejak teori tentang IR berkembang di tahun 1940an, para ilmuan selalu memeras otak, bagaimana caranya membuat sistem IR yang benar-benar handal. Bagaimana mengukur keefektifan sebuah sistem IR dalam memenuhi permintaan informasi? Bagaimana mengukur kemampuan sistem dalam menyediakan dokumen yang relevan dengan kebutuhan pemakai? Nah, recall and precision adalah upaya untuk menjawab persoalan itu.

Precision dapat diartikan sebagai kepersisan atau kecocokan (antara permintaan informasi dengan jawaban terhadap permintaan itu). Jika seseorang mencari informasi di sebuah sistem, dan sistem menawarkan beberapa dokumen, maka kepersisan ini sebenarnya juga adalah relevansi. Artinya, seberapa persis atau cocok dokumen tersebut untuk keperluan pencari informasi, bergantung pada seberapa relevan dokumen tersebut bagi si pencari. [5]

Precision adalah rasio jumlah dokumen relevan yang ditemukan dengan total jumlah dokumen yang ditemukan oleh sistem, dengan rumus precision adalah sebagai berikut **Error! Reference source not found.**:

$$\text{Precision} = \frac{|\{a\} \cap \{b\}|}{|\{b\}|} \quad (7)$$

Diketahui :

a : jumlah dokumen relevan

b : jumlah dokumen yang diuji

Precision mengindikasikan kualitas himpunan jawaban, tetapi tidak memandang total jumlah dokumen yang relevan dalam kumpulan dokumen. Sedangkan

Recall adalah proporsi jumlah dokumen yang dapat ditemukan-kembali oleh sebuah proses pencarian di sistem IR. Rumusnya: Jumlah dokumen relevan yang ditemukan / Jumlah semua dokumen relevan di dalam koleksi. Lalu, *precision* adalah proporsi jumlah dokumen yang ditemukan dan dianggap relevan untuk kebutuhan si pencari informasi. Rumusnya: Jumlah dokumen relevan yang ditemukan / Jumlah semua dokumen yang ditemukan.[6]

Recall adalah rasio jumlah dokumen relevan yang ditemukan kembali dengan total jumlah dokumen dalam kumpulan dokumen yang dianggap relevan, dengan rumus recall adalah sebagai berikut :

$$\text{Recall} = \frac{|\{a\} \cap \{b\}|}{|\{a\}|} \quad (8)$$

Diketahui :

a : jumlah dokumen relevan

b : jumlah dokumen yang diuji

Kedua ukuran di atas biasanya diberi nilai dalam bentuk persentase, 1 sampai 100%. Sebuah sistem informasi akan dianggap baik jika tingkat *recall* maupun *precision*-nya tinggi. Jika ada seseorang mencari dokumen tentang "Pangeran Diponegoro" pada sebuah sistem, dan jika sistem tersebut memiliki 100 buku tentang Pangeran Diponegoro, maka kinerja terbaik adalah jika sistem tersebut berhasil menemukan 100 dokumen tentang Pangeran Diponegoro.

Kalau sistem tersebut memberikan 100 temuan, dan di temuan tersebut ada 50 dokumen tentang "Pangeran Diponegoro", maka nilai *recall*-nya adalah 0,5 (atau 50%) dan nilai *precision*-nya juga 0,5. Kalau sistem tersebut memberikan 1 dokumen saja, dan dokumen tersebut adalah tentang "Pangeran Diponegoro", maka *recall*-nya bernilai 0,01 dan precision-nya bernilai 1. Perhatikan bahwa nilai *precision* yang tinggi ini sebenarnya terjadi karena sistem memberikan hanya 1 jawaban kepada si pencari informasi. Kalau sistem memberikan 100 dokumen, dan hanya 1 yang relevan, maka nilai *recall*-nya tetap 0,01 dan *precision*-nya pun ikut merosot ke 0,01.

II.2.8. Alat – Alat Pemodelan Sistem

Pemodelan sistem merupakan hal yang penting bagi kelangsungan sistem itu sendiri. Pemodelan sistem adalah suatu upaya untuk menjaga efektivitas sistem dalam memenuhi kebutuhan pengguna sistem. Pemodelan sistem dapat bererti menyusun sistem yang baru untuk menggantikan sistem yang lama secara keseluruhan atau memperbaiki sistem yang sudah ada [10].

II.2.8.1. UML

UML adalah singkatan dari *Unified Modeling Language* yang berarti bahasa pemodelan standar. Chonoles mengatakan sebagai bahasa, berarti UML memiliki sintaks dan semantic. Ketika kita membuat model menggunakan konsep UML ada aturan-aturan yang harus diikuti. Bagaimana elemen pada model-model yang kita buat berhubungan satu dengan lainnya harus mengikuti standar yang ada. UML

bukan hanya sekedar diagram, tetapi juga menceritakan konteksnya. Ketika pelanggan memesan sesuatu dari sistem, bagaimana transaksinya? Bagaimana sistem mengatasi *error* yang terjadi? Bagaimana keamanan terhadap sistem yang kita buat? Dan sebagainya dapat dijawab dengan UML. UML diaplikasikan untuk maksud tertentu, biasanya antara lain :

a. Merancang perangkat lunak.
b. Sarana komunikasi antara perangkat lunak dengan proses bisnis.
c. Menjabarkan sistem secara rinci untuk analisa dan mencari apa yang diperlukan sistem.
d. Mendokumentasi sistem yang ada, proses-proses dan organisasinya.

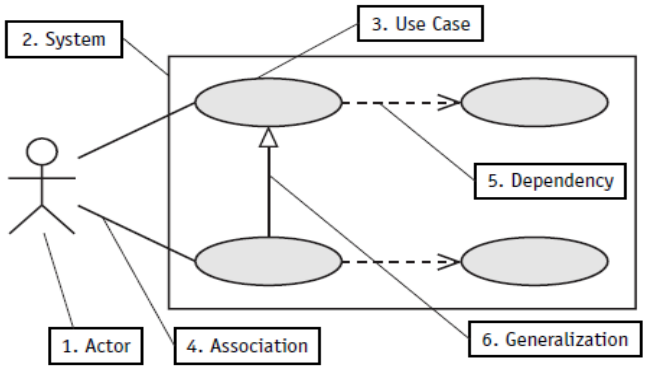
Akan dijelaskan empat diagram yang paling sering digunakan dalam pemodelan UML, yaitu :

1. *Use-Case* Diagram

Salah satu kontributor terhadap diagram *use-case* dalam UML adalah Ivar Jacobsen. *Use case* menggambarkan *external view* dari sistem yang akan kita buat modelnya. Pooley mengatakan bahwa model *use case* dapat dijabarkan dalam diagram *use case*, tetapi yang perlu diingat, diagram tidak identik dengan model karena model lebih luas dari diagram.

Komponen pembentuk diagram *use case* adalah :

a. Aktor (*actor*), menggambarkan pihak-pihak yang berperan dalam sistem.
b. *Use case*, aktivitas/sarana yang disiapkan oleh bisnis/sistem.
c. Hubungan (*link*), aktor mana saja yang terlibat dalam *use case* ini.



Gambar.II.8 UML Weekend Crash Course

Ada 6 elemen yang membangun *use case* diagram : *actor, system, use case, association, dependencies, dan generalization.*

a. *Actor*

Sebuah peran yang dimainkan oleh orang, sistem, atau perangkat yang memiliki saham dalam keberhasilan pengoperasian sistem.

b. *System*

Mengatur batas sistem dalam kaitannya dengan aktor yang menggunakannya (diluar sistem) dan fitur harus memberikan (dalam sistem).

c. *Use case*

Mengidentifikasi fitur kunci dari sistem. Tanpa fitur ini, sistem tidak akan memenuhi pengguna/aktor persyaratan. Setiap *use case* mengungkapkan bahwa tujuan sistem harus tercapai.

d. *Association*

Mengidentifikasi interaksi antara aktor dan *use case*. Setiap asosiasi menjadi sebuah dialog yang harus dijelaskan dalam sebuah narasi *use case*. Setiap narasi

dalam gilirannya memberikan seperangkat skenario yang berfungsi sebagai uji kasus ketika mengevaluasi analisis, desain, dan implementasi *use case*.

e. *Dependencies*

Mengindentifikasi hubungan komunikasi antara dua *use case*.

f. *Generalization*

Mendefinisikan sebuah hubungan antara dua aktor atau dua *use case* yang mana salah satu *use case* mewarisi dan menambah atau menggantikan sifat-sifat yang lain.

a. *Class* Diagram

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

Class diagram menggambarkan struktur dan deskripsi *class*, *package* dan objek beserta hubungan satu sama lain seperti *containment*, pewarisan, asosiasi, dan lain-lain. *Class* memiliki tiga area pokok yaitu :

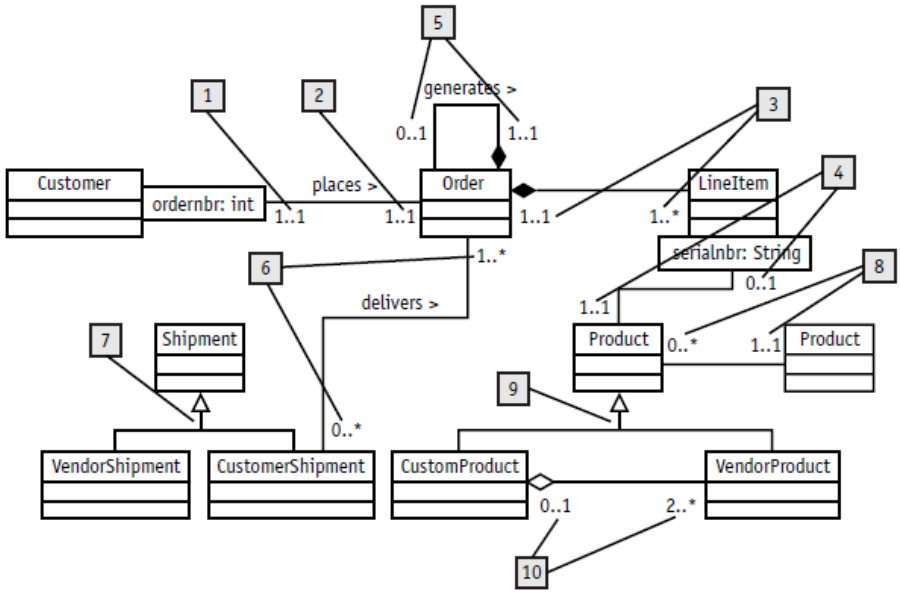
a. Nama (dan *stereotype*)
b. Atribut
c. Metoda

Atribut dan metoda memiliki salah satu dari sifat berikut :

a. *Private*, tidak dapat dipanggil dari luar *class* yang bersangkutan.
b. *Protected*, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
c. *Public*, dapat dipanggil oleh siapa aja.

Class dapat merupakan implementasi dari sebuah *interface*, yaitu *class* abstrak yang hanya memiliki metoda. *Interface* tidak dapat langsung diinstansiasikan, tetapi harus diimplementasikan dahulu menjadi sebuah *class*. Dengan demikian *interface* mendukung resolusi metoda pada saat run-time. Hubungan antar *class* antara lain :

a. Asosiasi, yaitu hubungan statis antar *class*. Umumnya menggambarkan *class* yang memiliki atribut berupa class lain, atau *class* yang harus mengetahui eksistensi lain. Panah *navigability* menunjukkan arah *query* antar class.
b. Agregasi, yaitu hubungan yang menyatakan bagian ("terdiri atas..").
c. Pewarisan, yaitu hubungan hierarki antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari *class* yang diwarisinya. Kebalikan dari pewarisan adalah generalisasi.
d. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-passing dari satu *class* kepada *class* lain. Hubungan dinamis dapat digambarkan dengan menggunakan *sequence* diagram yang akan dijelaskan kemudian.

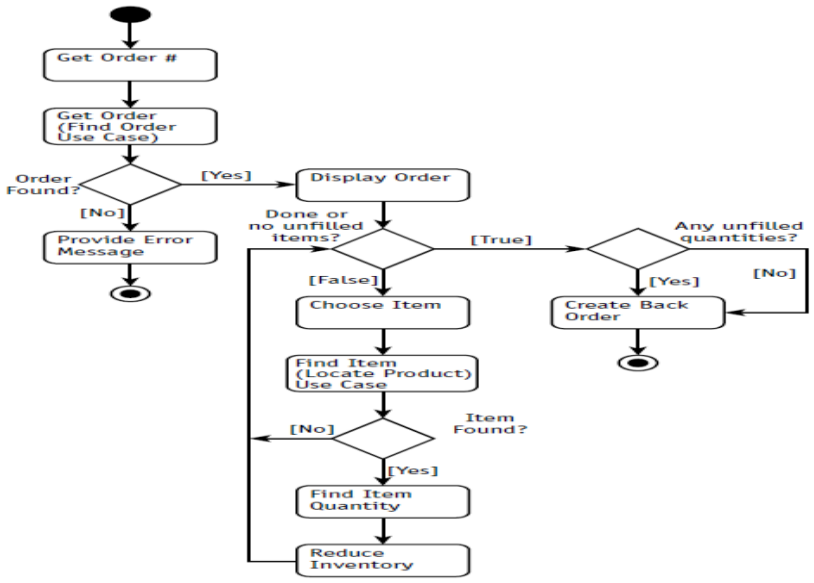


Berdasarkan gambar 3.3 di atas, maka dapat dijelaskan sebagai berikut:

1. Pelanggan menempatkan pesanan untuk satu atau lebih item tidak peduli apakah mereka telah membeli atau belum.
2. Pelanggan menempatkan pesanan untuk satu atau lebih item. Pelanggan dapat menanyakan tentang status pesanannya dengan menggunakan nomor urut.
3. Pelanggan memesan untuk satu atau lebih item.
4. Setiap item sesuai dengan produk. Produk diidentifikasikan dengan menggunakan nomor seri yang unik.
5. Setiap item yang belum dikirim ditempatkan ada backorder dengan referensi untuk urutan asli.
6. Pelanggan pesanan dikirimkan sebagai produk yang sudah tersedia, sehingga mungkin ada lebih dari satu pengiriman untuk memenuhi pesanan pelanggan tunggal.
7. Pengiriman produk dari *vendor* yang diterima dan ditempatkan ke saham.
8. Setiap produk ditugaskan untuk lokasi sehingga nanti kita dapat dengan mudah menemukan ketika mengisi perintah. Setiap lokasi memiliki pengenal lokasi yang unik.
9. Produk ini dapat dibeli langsung dari *vendor* dan dijual kembali.
10. Kita dapat membuat paket produk vendor bersama-sama untuk membuat produk sendiri.

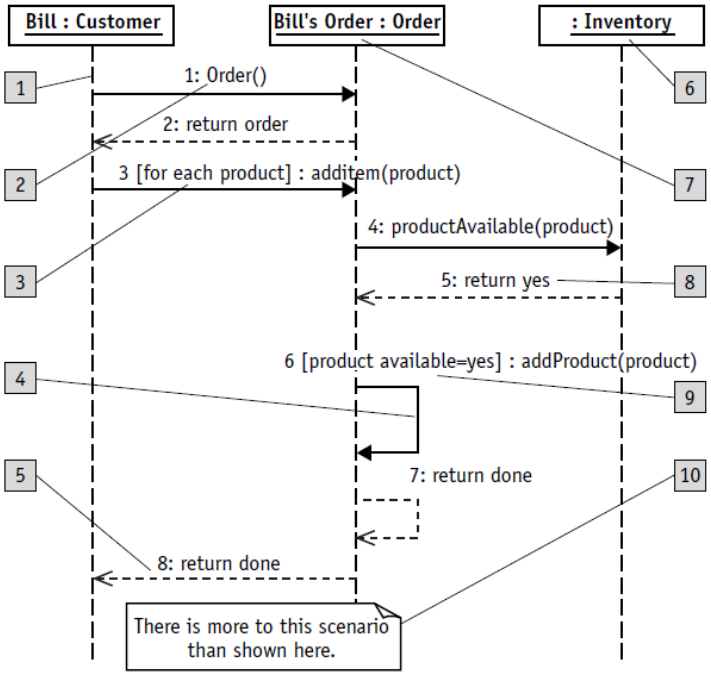
2. *Activity* Diagram

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



1. *Sequence* Diagram

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



II.2.11.1. Java

Java adalah bahasa pemrograman tingkat tinggi yang berorientasi objek dan program java tersusun dari bagian yang disebut kelas. Kelas terdiri atas metode-metode yang melakukan pekerjaan dan mengembalikan informasi setelah melakukan tugasnya. Para pemrograman Java banyak mengambil keuntungan dari kumpulan kelas di pustaka kelas Java, yang disebut dengan *Java Application Programming Interface* (API). Kelas-kelas ini diorganisasikan menjadi sekelompok yang disebut paket *(package)*. Java API telah menyediakan fungsionalitas yang memadai untuk menciptakan *applet* dan aplikasi canggih. Jadi ada dua hal yang harus dipelajari dalam Java, yaitu mempelajari bahasa Java dan bagaimana mempergunakan kelas pada Java API. Kelas merupakan satu-satunya cara menyatakan bagian eksekusi program, tidak ada cara lain. Pada Java program javac untuk mengkompilasi file kode sumber Java menjadi kelas-kelas *bytecode*. File kode sumber mempunyai ekstensi *.java. Kompilator javac menghasilkan file *bytecode* kelas dengan ekstensi *.class.

Interpreter merupakan modul utama sistem Java yang digunakan aplikasi Java dan menjalankan program *bytecode* Java.

Java merupakan bahasa berorientasi objek *(OOP)* yaitu cara ampuh dalam pengorganisasian dan pengembangan perangkat lunak. Pada OOP, program komputer sebagai kelompok objek yang saling berinteraksi. Deskripsi ringkas OOP adalah mengorganisasikan program sebagai kumpulan komponen, disebut objek. Objek-objek ini ada secara independen, mempunyai aturan-aturan berkomunikasi dengan objek lain dan untuk memerintahkan objek lain guna meminta informasi tertentu atau meminta objek lain mengerjakan sesuatu. Kelas bertindak sebagai modul sekaligus tipe. Sebagai tipe maka pada saat jalan, program menciptakan objek-objek yang merupakan instan-instan kelas. Kelas dapat mewarisi kelas lain. Java tidak mengijinkan pewarisan jamak namun menyelesaikan kebutuhan pewarisan jamak dengan fasilitas antarmuka yang lebih elegan.

Seluruh objek diprogram harus dideklarasikan lebih dulu sebelum digunakan. Ini merupakan keunggulan Java yaitu *Statically Typed*. Pemaksaan ini memungkinkan kompilator Java menentukan dan melaporkan terjadinya pertentangan (ketidakkompatibelan) tipe yang merupakan barikade awal untuk mencegah kesalahan yang tidak perlu (seperti mengurangkan variabel bertipe integer dengan variabel bertipe string). Pencegahan sedini mungkin diharapkan menghasilkan program yang bersih. Kebaikan lain fitur ini adalah kode program lebih dapat dioptimasi untuk menghasilkan program berkinerja tinggi.

Java menggunakan model pengamanan tiga lapis *(three-layer security model)* untuk melindungi sistem dari *untrusted Java code.* Pertama, *bytecode verifier* membaca *bytecode* sebelum dijalankan dan menjamin *bytecode* memenuhi aturan-aturan dasar bahasa Java. Kedua, *class loader* menangani pemuatan kelas Java ke *runtime interpreter*. Ketiga, manajer keamanan menangani keamanan tingkat aplikasi

dengan mengendalikan apakah program berhak mengakses sumber daya seperti sistem file, port jaringan, proses eksternal dan sistem *window*.

Java termasuk bahasa *Multithreading*. *Thread* adalah untuk menyatakan program komputer melakukan lebih dari satu tugas di satu waktu yang sama. Java menyediakan kelas untuk menulis program *multithreaded*, program mempunyai lebih dari satu *thread* eksekusi pada saat yang sama sehingga memungkinkan program menangani beberapa tugas secara konkuren.

Program Java melakukan *garbage collection* yang berarti program tidak perlu menghapus sendiri objek-objek yang tidak digunakan lagi. Fasilitas ini mengurangi beban pengelolaan memori oleh pemrogram dan mengurangi atau mengeliminasi sumber kesalahan terbesar yang terdapat di bahasa yang memungkinkan alokasi dinamis.