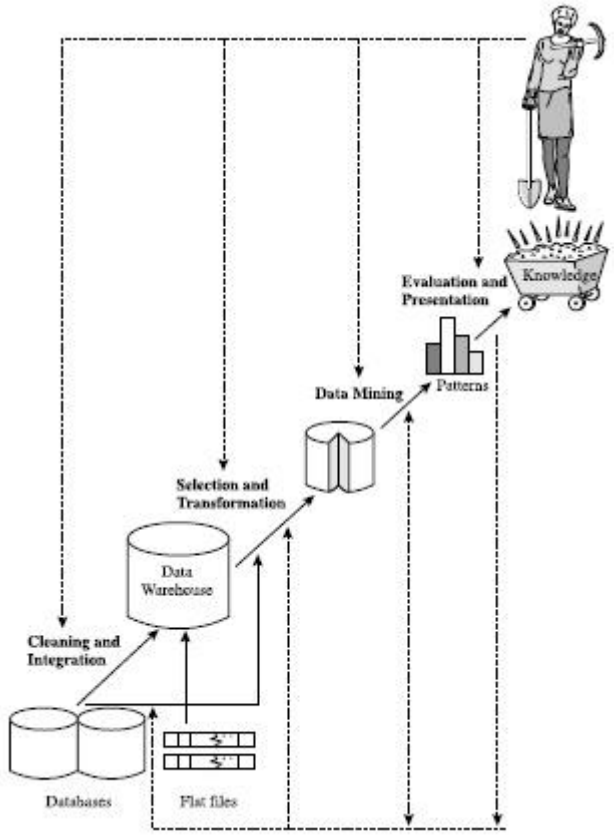
BAB II

LANDASAN TEORI

2.1. Data Mining

Definisi yang paling umum diterima dari "data mining" adalah penemuan "Model" untuk data (Leskovec, Rajaraman, & Ullman, 2014). Secara sederhana, data mining merujuk pada ekstraksi atau "pertambangan" pengetahuan dari sejumlah besar data (Han & Kamber, 2006). Data mining merupakan proses iteratif dan interaktif untuk menemukan pola atau model baru yang dapat digeneralisasi untuk masa yang akan datang, bermanfaat dan dapat dimengerti dalam suatu *database* yang sangat besar (*massive databases*). 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 lainnya (Hermawati, 2013). Berikut ini adalah proses data mining yang dapat dilihat pada gambar 2:



Gambar 1 Proses Data Mining (Han & Kamber, 2006)

1. Pembersihan Data (*data cleaning*)

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

2. Integrasi Data (*data integration*)

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

3. Seleksi Data (*data selection*)

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

4. Transformasi Data (*data transformation*)

   Data diubah atau digabung ke dalam format yang sesuai untuk diproses dalam data mining. Beberapa metode dari data mining membutuhkan format data yang khusus sebelum bisa diaplikasikan. Sebagai contoh beberapa metode standar seperti analisis asosiasi dan *clustering* hanya

bisa menerima input data kategorikal. Karenanya data berupa angka numerik yang berlanjut perlu dibagi-bagi menjadi beberapa interval. Proses ini sering disebut transformasi data.

5. Penambangan Data (*data Mining*)

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

6. Evaluasi Pola (*pattern evaluation*)

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

7. Presentasi Pengetahuan (*knowledge presentation*)

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

tahapan yang diperlukan dalam proses data mining. Dalam presentasi ini, visualisasi juga bisa membantu mengkomunikasikan hasil data mining.

Langkah 1 sampai 4 adalah bentuk-bentuk yang berbeda dari *preprocessing* data, dimana data disusun untuk pertambangan. Langkah data mining dapat berinteraksi dengan pengguna atau basis pengetahuan. Itu pola yang menarik disajikan kepada pengguna dan dapat disimpan sebagai pengetahuan baru dalam basis pengetahuan. Perhatikan bahwa menurut pandangan ini, data mining hanya satu langkah dalam seluruh proses, meskipun salah satu yang penting karena mengungkap pola tersembunyi untuk evaluasi.

2.4.1. *Normalization*

Normalisasi adalah proses penskalan nilai atribut dari data sehingga bisa jatuh pada range tertentu. Contoh, misalnya berkenaan dengan pencatatan tingkat kematian penduduk di Indonesia per bulannya berdasarkan jenis umur. Secara sederhana, disana ada 3 dimensi data, yaitu bulan (1-12), umur (0-150) misalnya, dan jumlah kematian (0-jutaan). Kalau dibentangkan range masing-masing dimensi, maka akan mendapatkan ketidakseimbangan range pada dimensi yang ketiga, yaitu jumlah kematian. Salah satu metode normalisasi adalah z-score. Z-Score merupakan metode normalisasi yang berdasarkan mean (nilai rata-rata) dan standard deviation (standar deviasi dari data). Normalisasi z-score sangat berguna apabila

tidak mengetahui nilai aktual minimum dan maksimum dari data. Berikut ini adalah persamaan normalisasi z-score:

$$newdata = \frac{data - \bar{x}}{s} \quad \text{..... 1}$$

Dimana: $s = \sqrt{\frac{\sum(x_i - \bar{x})^2}{n-1}}$ 2

newdata = data hasil normalisasi

$\bar{x}$ = rata-rata

s = standar deviasi

n = banyak data

2.2. Klastering

Clustering adalah proses pemeriksaan beberapa "objek," dan pengelompokan objek ke "cluster" menurut beberapa ukuran jarak (Leskovec, Rajaraman, & Ullman, 2014). Tujuannya agar poin di cluster yang sama memiliki jarak kecil dari satu sama lain, sementara poin dalam kelompok yang berbeda berada pada jarak yang besar dari satu sama lain. Proses pengelompokan satu set objek fisik atau abstrak ke dalam kelas objek serupa disebut *clustering* (Han & Kamber, 2006). *Clustering* adalah sebuah proses untuk mengelompokan data ke dalam beberapa cluster atau kelompok sehingga data dalam satu *cluster* memiliki tingkat kemiripan yang maksimum dan data antar *cluster* memiliki kemiripan yang minimum (Tan, Steinbach, & Kumar, 2006). Pengelompokan (*clustering*) merupakan teknik yang sudah cukup dikenal dan banyak digunakan untuk mengelompokkan data/objek ke dalam kelompok data (cluster) sehingga setiap cluster memiliki data yang mirip dan berbeda dengan data yang berada dalam cluster lain. Jika diberikan

himpunan data yang berjumlah terhingga, yaitu X, maka permasalahan *clustering* dalam X adalah mencari beberapa pusat cluster yang dapat memberikan ciri kepada masing-masing *cluster* dalam X (Arief, 2017). Sebuah cluster adalah kumpulan objek data yang mirip satu sama lain dalam cluster yang sama dan berbeda dengan benda-benda di cluster lainnya. Sekelompok data benda dapat diperlakukan secara kolektif sebagai satu kelompok dan dapat dianggap sebagai data yang formof kompresi. Keuntungan dari proses berbasis *cluster* adalah dapat beradaptasi perubahan dan membantu satu fitur yang berguna yang membedakan kelompok yang berbeda. *Clustering* juga disebut segmentasi data dalam beberapa aplikasi karena mengelompokkan partisi data besar set ke dalam kelompok sesuai dengan kesamaan mereka. *Clustering* juga bisa digunakan untuk deteksi *outlier*, di mana *outlier* (nilai-nilai yang "jauh" dari kelompok manapun) mungkin lebih menarik daripada kasus umum.

Dalam mengukur jarak dalam klastering dapat dilakukan dengan menggunakan *Manhattan Distance* ataupun *Euclidean Distance*. *Manhattan Distance* merupakan pengukuran jarak objek dan pusat klaster yang banyak digunakan dalam kasus PAM. *Manhattan distance* dinyatakan dengan persamaan:

$$d(o,m) = \sum_{i=1}^{n} |m_i - o_i| \qquad \text{..... 3}$$

Berikut ini adalah persamaan *Euclidean Distance:*

$$d(o,m) = \sqrt{\sum_{i=1}^{n} (m_i - o_i)^2} \qquad \text{..... 4}$$

Dimana: n = banyak data

i = indeks data

o_i = nilai atribut ke–i dari o

m_i = nilai atribut ke-i dari m

2.3. *Partitioning Around Medoids* (PAM)

Algoritma *Partitioning Around Medoids* atau dikenal juga K-Medoids adalah algoritma pengelompokan yang berkaitan dengan algoritma K-Means dan algoritma medoidshift. Algortima K-Medoids ini diusulkan pada tahun 1987.

Algoritma *Partitioning Around Medoids* (PAM) dikembangkan oleh Leonard Kaufman dan Peter J. Rousseeuw. Algoritma ini sangat mirip dengan algoritma K-Means, terutama karena algoritma ini partitional. Dengan kata lain, kedua algoritma ini memecah dataset menjadi kelompok-kelompok dan kedua algoritma ini berusaha untuk meminimalkan kesalahan. Tetapi algoritma *Partitioning Around Medoids* bekerja dengan menggunakan Medoids, yang merupakan entitas dari dataset yang mewakili kelompok dimana ia dimasukkan.

Algoritma *Partitioning Around Medoids* menggunakan metode partisi klastering untuk mengelompokkan sekumpulan n objek menjadi sejumlah k klaster. Algoritma ini menggunakan objek pada kumpulan objek untuk mewakili sebuah klaster. Objek yang terpilih untuk mewakili sebuah klaster disebut medoid. Klaster dibangun dengan menghitung kedekatan yang dimiliki antara medoid dengan objek non-medoid.

Algoritma dari *Partitioning Around Medoids* atau K-Medoids adalah sebagai berikut (Han & Kamber, 2006):

1. Secara acak pilih *k* objek pada sekumpulan *n* objek sebagai medoid.
2. Tempatkan objek non-medoid ke dalam klaster yang paling dekat dengan medoid.
3. Secara acak pilih o_{random} (sebuah objek nom-medoid).
4. Hitung total biaya, S, dari pertukaran *medoid* o_j dengan o_{random}.
5. Jika S<0 maka tukar o_j dengan o_{random} untuk membentuk sekumpulan *k* objek baru sebagai medoid.
6. Ulangi langkah 2 hingga langkah 5.
7. Hingga tidak ada perubahan.

Nilai total biaya/*cost* dinyatakan dengan persamaan:

$$Total\ Biaya = \sum_{j=1}^{k} \sum_{p \in cj} |p - o_j| \quad \text{..... 5}$$

Nilai S dinyatakan dengan persamaan:

$$S = Total\ cost\ baru - Total\ cost\ lama \quad \text{..... 6}$$

Dimana:

Total *cost* baru = jumlah biaya/*cost* non-medoids

Total *cost* lama = jumlah biaya/*cost* medoids

Berikut ini adalah beberapa penelitian mengenai *Partitioning Around Medoids* (*K-Medoids*). Dalam penenilitan "*An Efficient Density Based Improved K-Medoids Clustering Algorithm"* mereka membandingkan algoritma DBSCAN dengan K-Medoids, dan hasil yang didapatkan adalah algoritma K-Medoids bekerja lebih baik dibandingkan dengan algoritma DBSCAN (A, Devi, Vani, & Rao, 2011). Dalam penelitian "Penerapan Algoritma *Paritioning Around Medoids* (PAM) Clustering untuk Melihat

Gambaran Umum Kemampuan Akademik Mahasiswa" memiliki kesimpulan bahwa algoritma yang digunakan memiliki mutu klastering yang baik, dimana setiap objek telah dikelompokkan sesuai dengan tingkat kesamaan yang tinggi (Chrisnanto & Abdillah, 2015). Dalam penelitian " *A Hybrid Algorithm for K-Medoid Clustering of Large Data Sets*" menghasilkan bahwa algoritma K-Medoids berhasil di hibridisasi untuk data set yang lebih besar yaitu menghasilkan algoritma HKA (Sheng & Liu, 2004). Dalam penelitian "*Scalling K-Medoid Algorithm for Clustering Large Categorical Dataset and Its Performance Analysis*" mencoba menganalisa k-medoid untuk dataset yang jumlahnya lebih besar (Joshi, Patidar, & Mishra, 2011). Dalam penelitian "Clustering of Scale Free Network Using a K-medoid Framework" membuktikan bahwa algoritma K-medoid dapat mengklaster jaringan dalam skala kecil (Ray & Pakhira, 2011). Dalam penelitian "*Medoid selection from sub-tree leaf nodes for k-medoid clustering-based hierarchical template tree construction*" menggunakan k-medoid untuk deteksi kontur pejalan kaki (Jung, 2013). Dalam penelitian "*Image Segmentation using Rough-Fuzzy K-medoid Algorithm*" menggunakan k-medoid untuk segmentasi gambar (Halder, Dasgupta, & Ghosh, 2012). Dalam penelitian "*MapReduce Model For K-Medoid Clustering*" mencoba menggabungkan model *map reduce* dengan k-medoid (Harikumar & Thaha, 2016). Dalam penelitian "*Hierarchical and Non-hierarchical Medoid Clustering Using Asymmetric Similarity Measures*" mencoba mengembangkan algoritma k-medoid (Miyamoto, Kaizu, & Endo, 2016). Dalam penelitian "*Breast Cancer Symptom Clusters Derived From Social Media and Research Study Data Using Improved K-Medoid*

Clustering" mengklaster populasi kanker payudara (Ping, Yang, Marshall, Avis, & Ip, 2016). Dalam penelitian "*Two-Stage Clustering Using One-Pass Medoids and Medoid-Based Agglomerative Hierarchical Algorithms"* mencoba melakukan dua pendekatan medoid (Tamura & Miyamoto, 2014). Dalam penelitian "*Finding Good XML Fragments based on k-medoid Cluster Number Optimization and Ranking Model for Feedback*" mengklaster xml terbaik.

.K-Medoids sangat mirip dengan K-Means, perbedaan utama diantara dua algoritma tersebut adalah jika pada K-Means klaster diawali dengan pusat dari klaster, sedangkan pada K-Medoids klaster diawali oleh objek terdekat dari pusat klaster.

2.4. Validasi Klaster

Validasi merupakan proses untuk menilai hasil algoritma klaster. Oleh karena itu, proses ini bertujuan untuk menjamin bahwa solusi klaster yang dihasilkan dalam analisis klaster dapat menggambarkan populasi sebenarnya.

Menurut Gordon (Yatskiv & Gusarova, 2004), mengatakan bahwa terdapat 3 pendekatan utama dalam melakukan validasi klaster yaitu:

1. *External Test*, dalam uji ini data dibagi menjadi dua bagian. Solusi klaster dari data hasil analisis klaster dibandingkan dengan solusi klaster dari data yang tidak diikutsertakan dalam analisis klaster tersebut.
2. *Internal Test*, dalam uji ini solusi klaster digunakan untuk melihat kualitas klaster dengan cara membandingkan solusi klaster hasil metode hirarki dan metode iterative (non-hirarki).

3. *Relative Test*, dalam uji ini beberapa solusi klaster yang berbeda dari data dibandingkan menggunakan algoritma yang sama dengan parameter yang berbeda.

Pada dasarnya, validasi memberikan informasi tentang ketepatan jumlah klaster yang dipilih. Dalam hal ini, jumlah klaster yang terbentuk dikatakan baik apabila solusi klaster yang dihasilkan tidak jauh berbeda berdasarkan pendekatan yang digunakan. Validasi dengan pendekatan *internal test* lebih sering digunakan dalam praktik analisis klaster disebabkan pendekatan ini lebih sederhana dan mudah.

Dalam penelitian ini, validasi lebih ditekankan pada pendekatan *relative test* dengan statistic uji sebagai berikut:

2.7.1. Davies Bouldien Index (IDB)

IDB bertujuan untuk memaksimumkan jarak antara klaster (*inter cluster*) yang satu dengan klaster yang lain (*separation value*) dan meminimumkan jarak antara titik (*intra cluster*) dalam sebuah klaster (*compactness value*). IDB didefinisikan sebagai (Rahmawati, Cahyani, & Putro):

..... 7

$$\text{IDB} = \frac{1}{k}\sum_{i=1}^{k} R_i$$

Dimana $Ri = \frac{Var\,(ci)+Var\,(cj)}{|\bar{x}ci-\bar{x}cj|}$ 8

Perhatikan bahwa *Variance* (Var), didefinisikan sebagai:

$$Var = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$$ 9

Sedangkan rata-rata, didefinisikan sebagai:

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \text{..... 10}$$

Dimana: IDB = validasi Davies Bouldin

k = jumlah klaster

R = jarak antar cluster

Var = variance dari data

N = banyak data

x_i = data ke-i

$\bar{x}$ = rata-rata dari tiap cluster

n = banyak data

Nilai IDB berada pada interval (0,1), nilai minimum dari IDB akan menunjukkan jumlah klaster optimal.

2.5. PHP

PHP (*Hypertext Preprocessor*) adalah bahasa pemrograman yang digunakan untuk membuat aplikasi *website*, jika kode HTML diproses oleh web browser di komputer *user*, maka PHP diproses pada server web dan hasilnya baru dikirim ke web browser. Pengertian lain dari PHP adalah bahasa pemrograman yang dapat digunakan untuk tujuan umum, sama seperti bahasa pemrograman lain: C, C++, Pascal, Phyton, Perl, Ruby, dan sebagainya, meskipun demikian PHP lebih popular digunakan untuk pengembangan aplikasi *web*. Selain itu, ada pendapat lain yang mengemukakan bahwa PHP merupakan secara umum dikenal sebagai bahasa pemrograman script script yang membuat dokumen HTML secara *on the fly* yang dieksekusi di server *web*. PHP Pertama kali ditemukan pada 1995 oleh seorang *Software*

Developer bernama Rasmus Lerdrof. Ide awal PHP adalah ketika itu Radmus ingin mengetahui jumlah pengunjung yang membaca *resume online*. *Script* yang dikembangkan baru dapat melakukan dua pekerjaan, yakni merekam informasi *visitor*, dan menampilkan jumlah pengunjung dari suatu *website*. Dan sampai sekarang kedua tugas tersebut masih tetap populer digunakan oleh dunia *web* saat ini. Kemudian, dari situ banyak orang di milis mendiskusikan *script* buatan Rasmus Lerdrof, hingga akhirnya rasmus mulai membuah sebuah *tool/script*, bernama *Personal Home Page* (PHP).

2.6. *Database*

Basis data terdiri atas 2 kata, yaitu Basis dan Data. Basis kurang lebih dapat diartikan sebagai markas atau gudang, tempat bersarang/berkumpul. Sedangkan Data adalah representasi fakta dunia nyata yang mewakili suatu objek seperti manusia (pegawai, siswa, pembeli, pelanggan), barang, hewan, peristiwa, konsep, keadaan, dan sebagainya, yang diwujudkan dalam bentuk angka, huruf, simbol, teks, gambar, bunyi, atau kombinasinya (Fathansyah, 2015). Basis data (*database*) adalah kumpulan informasi yang disimpan di dalam komputer secara sistematik sehingga dapat diperiksa menggunakan suatu program komputer untuk memperoleh informasi dari basis data tersebut. Pengertian lain dari *database* adalah kumpulan data yang terintegrasi dan diatur sedemikian rupa sehingga data tersebut dapat dimanipulasi, diambil dan dicari secara cepat (Raharjo, 2011). *Database* digunakan untuk menyimpan informasi atau data yang terintegrasi dengan baik di dalam komputer. Untuk mengelola *database* diperlukan suatu perangkat lunak yang disebut DBMS (*Database Management System*).

DBMS merupakan suatu sistem perangkat lunak yang memungkinkan *user* (pengguna) untuk membuat, memelihara, mengontrol, dan mengakses *database* secara praktis dan efisien. Dengan DBMS, *user* akan lebih mudah mengontrol dan memanipulasi data yang ada (Solichin, 2010).

2.7. MySQL

MySQL merupakan *software* RDBMS (atau *server database*) yang dapat mengelola *database* dengan sangat cepat, dapat menampung data dalam jumlah sangat besar, dapat diakses oleh banyak *user* (*multi-user*), dan dapat melakukan suatu proses secara sinkron atau berbarengan (*multi-threaded)* (Raharjo, 2011). Pengertian lain dari *MySQL* adalah sebuah perangkat lunak sistem manajemen basis data SQL (bahasa Inggris: *database management system*) atau DBMS yang *multithread*, *multi-user*, dengan sekitar 6 juta instalasi di seluruh dunia (Solichin, 20010). Selain itu ada yang berpendapat lain yang mengatakan bahwa *MySQL* merupakan *Database Management System* (DBMS) yang didistribusikan secara gratis di bawah lisensi dari *General Public License* (GPL), dimana setiap orang bebas untuk mengunduh tetapi tidak boleh untuk dijadikan program induk turunannya yang bersifat komersial (Darmayuda, 2014). *MySQL* AB membuat *MySQL* tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual dibawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL. Tidak seperti *PHP* atau *Apache* yang merupakan *software* yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing, *MySQL* dimiliki dan disponsori oleh sebuah perusahaan

komersial Swedia yaitu *MySQL* AB. *MySQL* AB memegang penuh hak cipta hampir atas semua kode sumbernya. Kedua orang Swedia dan satu orang Finlandia yang mendirikan *MySQL* AB adalah: David Axmark, Allan Larsson, dan Michael "Monty" Widenius.

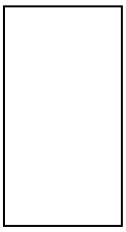
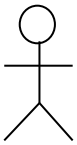
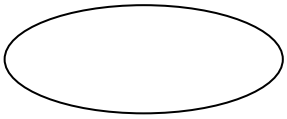
2.8. *Unified Modelling Language* (UML)

UML menyediakan bahasa untuk menggambarkan interaksi sistem, yang didukung oleh seperangkat kohesif yang dikelola oleh kelompok resmi, Manajemen Obyek Group (OMG) (Eriksson, Penker, Lyons, & Fado, 2004). Unified Modeling Language (UML) adalah keluarga notasi grafis, didukung oleh satu meta-model, yang membantu dalam menjelaskan dan merancang perangkat lunak sistem, khususnya sistem perangkat lunak yang dibangun menggunakan berorientasi objek (00) (Fowler, 2004). *Unified Modeling Language* (UML) adalah notasi grafis untuk menggambar diagram konsep *software* (Martin, 2002).

2.10.1. *Use Case Diagram*

Use case diagram menunjukkan sejumlah aktor eksternal dan hubungan mereka dengan kasus penggunaan yang sistem sediakan (Eriksson, Penker, Lyons, & Fado, 2004). Notasi *use case diagram* seperti terlihat pada tabel 1.

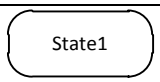
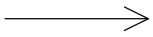
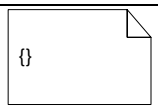
Tabel 1 Notasi Use Case Diagram

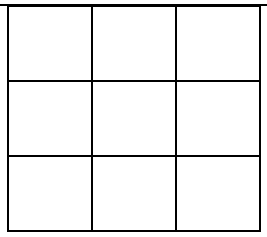
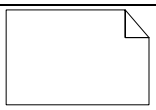
Bentuk Simbol	Nama	Keterangan
	*System Boundary*	Simbol ini digunakan untuk membatasi antara sistem yang satu dengan sistem lainnya atau dengan lingkungan luarnya.
	*Actor*	Simbol ini digunakan sebagai penggambaran subjek yang berhubungan langsung dengan sistem.
	*Use Case*	Digunakan untuk menggambarkan interaksi atau *interface* yang terjadi antara *actor* dengan sistem.
	Relationship	Simbol ini digunakan untuk menghubungkan antara actor dan use case yang saling berhubungan di dalam sistem tersebut.

2.10.2. *Activity Diagram*

Activity Diagram menunjukkan aliran sekuensial tindakan, *activity diagram* biasanya digunakan untuk menggambarkan kegiatan yang dilakukan dalam proses alur kerja umum, meskipun juga dapat digunakan untuk menggambarkan aktivitas lainnya mengalir, seperti kasus penggunaan atau aliran kontrol rinci (Eriksson, Penker, Lyons, & Fado, 2004). *Activvity Diagram* adalah teknik untuk menggambarkan logika prosedural, proses bisnis,dan alur kerja (Fowler, 2004). Notasi *activity diagram* seperti terlihat pada tabel 2.

Tabel 2 Notasi Activicty Diagram

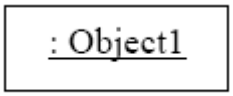
Bentuk Simbol	Nama	Keterangan
	*Solid Circle*	Untuk memulai proses
	*Rounded Rectangle*	aktivitas di pemicu
	*Continous Line*	Urutan dari satu peristiwa atau aktivitas ke aktivitas berikutnya
	Dofted Line	Aliran informasi antara peristiwa
	*Document*	Merupakan dokumen sumber atau laporan
	*Diamond*	Cabang

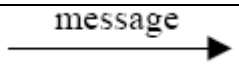
Bentuk Simbol	Nama	Keterangan
	*Table*	Sebuah file komputer dari mana data dapat dibaca dari atau direkam selama acara bisnis
	*Bull's Eye*	Akhir proses
	*Note*	Mengacu pembaca untuk diagram lain atau dokumen untuk rincian

2.10.3. *Sequence Diagram*

Sequence Diagram menunjukkan kolaborasi dinamis antara sejumlah objek, aspek penting dari diagram ini adalah bahwa hal itu menunjukkan urutan pesan yang dikirim antara objek (Eriksson, Penker, Lyons, & Fado, 2004). Sequence diagram menggambarkan bagaimana kelompok-kelompok objek berkolaborasi dalam beberapa perilaku (Fowler, 2004). Notasi *sequence diagram* seperti terlihat pada tabel 3.

Tabel 3 Notasi Sequence Diagram

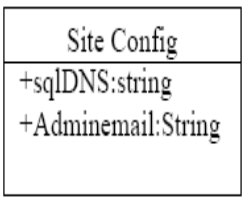
SIMBOL	NAMA	KETERANGAN
	**Object**	Object merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah class (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma

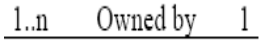
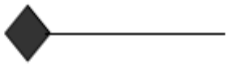
SIMBOL	NAMA	KETERANGAN
	**Actor**	Actor juga dapat berkomunikasi dengan object, maka actor juga dapat diurutkan sebagai kolom. Simbol Actor sama dengan simbol pada Actor Use Case Diagram.
	Lifeline	Lifeline mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk Lifeline adalah garis putus-putus vertikal yang ditarik dari sebuah obyek.
	**Activation**	Activation dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah lifeline. Activation mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.
	**Message**	Message, digambarkan dengan anak panah horizontal antara Activation. Message mengindikasikan komunikasi antara object-object.

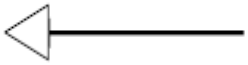
2.10.4. *Class Diagram*

Class Diagram memperlihatkan struktur statis dari kelas dalam sistem, kelas mewakili "hal-hal" yang ditangani dalam sistem (Eriksson, Penker, Lyons, & Fado, 2004). Notasi *class diagram* seperti terlihat pada tabel 4.

Tabel 4 Notasi Class Diagram

SIMBOL	NAMA	KETERANGAN
	**Class**	*Class* adalah blok - blok pembangun pada pemrograman berorientasi obyek. Sebuah class digambarkan sebagai sebuah kotak yang

SIMBOL	NAMA	KETERANGAN
		terbagi atas 3 bagian. Bagian atas adalah bagian nama dari *class*. Bagian tengah mendefinisikan property/atribut *class*. Bagian akhir mendefinisikan method method dari sebuah *class*.
	**Assosiation**	Sebuah asosiasi merupakan sebuah *relationship* paling umum antara 2 class, dan dilambangkan oleh sebuah garis yang menghubungkan antara 2 *class*. Garis ini bisa melambangkan tipe-tipe *relationship* dan juga dapat menampilkan hukum-hukum multiplisitas pada sebuah *relationship* (Contoh: One-to-one, one-to-many, many-to-many).
	**Composition**	Jika sebuah *class* tidak bisa berdiri sendiri dan harus merupakan bagian dari *class* yang lain, maka *class* tersebut memiliki relasi *Composition* terhadap *class* tempat dia bergantung tersebut. Sebuah *relationship composition* digambarkan sebagai garis dengan ujung berbentuk jajaran genjang berisi/solid.
	**Dependency**	Kadangkala sebuah *class* menggunakan *class* yang lain. Hal ini disebut *dependency*. Umumnya penggunaan *dependency* digunakan untuk menunjukkan operasi

SIMBOL	NAMA	KETERANGAN
		pada suatu *class* yang menggunakan *class* yang lain. Sebuah *dependency* dilambangkan sebagai sebuah panah bertitik-titik.
	**Aggregation**	*Aggregation* mengindikasikan keseluruhan bagian *relationship* dan biasanya disebut sebagai relasi "mempunyai sebuah" atau "bagian dari". Sebuah *aggregation* digambarkan sebagai sebuah garis dengan sebuah jajaran genjang yang tidak berisi/tidak solid.
	**Generalization**	Sebuah relasi *generalization* sepadan dengan sebuah relasi *inheritance* pada konsep berorientasi obyek. Sebuah *generalization* dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas "*paren*t"-nya/induknya.

2.9. Pengujian

Pengujian ini dimaksudkan untuk menunjukkan bahwa program melakukan apa yang dimaksudkan untuk dilakukan dan untuk menemukan cacat program sebelum digunakan (Sommerville, 2011). Ketika menguji perangkat lunak, saat mengeksekusi program menggunakan data buatan. Kemudian memeriksa hasil uji coba untuk kesalahan, anomali, atau informasi

tentang atribut non-fungsional program. Proses pengujian memiliki dua tujuan yang berbeda (Sommerville, 2011):

1. Untuk menunjukkan kepada pengembang dan pelanggan bahwa perangkat lunak memenuhi persyaratan. Untuk perangkat lunak kustom, ini berarti bahwa harus ada setidaknya satu tes untuk setiap kebutuhan dalam dokumen persyaratan. Untuk perangkat lunak generic produk, itu berarti bahwa harus ada tes untuk semua fitur sistem, ditambah kombinasi dari fitur ini, yang akan dimasukkan dalam rilis produk.
2. Untuk menemukan situasi di mana hasil dari software ini tidak benar, tidak diinginkan, atau tidak sesuai dengan spesifikasinya. Ini adalah konsekuensi dari cacat perangkat lunak. Pengujian cacat berkaitan dengan membasmi sistem yang tidak diinginkan. Hasil seperti sistem crash, interaksi yang tidak diinginkan dengan sistem lain, perhitungan yang salah, dan korupsi data.

2.11.1. *Black Box Testing*

Black box testing merupakan pengujian yang dilakukan hanya mengamati hasil eksekusi melalui data uji dan memeriksa fungsional dari perangkat lunak (Taslim, 2013). Pengujian *black box*, mengevaluasi hanya dari tampilan luarnya (*interface* nya), dan fungsionalitasnya tanpa mengetahui apa sesungguhnya yang terjadi dalam proses detailnya