BAB II

TINJAUAN PUSTAKA

II.1. Sistem Informasi

II.1.1. Sistem

Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan / berinteraksi dan saling memengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu.

Menurut Jerry FithGerald, Sistem adalah suatu jaringan kerja dari prosedur-prosedur yang saling berhubungan dan berkumpul bersama-sama untuk melakukan suatu kegiatan atau menyelesaikan suatu sasaran tertentu.

1. Syarat-Syarat Sistem
    a. Sistem harus dibentuk untuk menyelesaikan tujuan.
    b. Elemen sistem harus mempunyai rencana yang ditetapkan.
    c. Adanya hubungan di antara elemen sistem.
    d. Unsur dasar dari proses (arus informasi, energi dan material) lebih penting dari pada elemen sistem.
    e. Tujuan organisasi lebih penting dari pada tujuan elemen.
2. Karakteristik Sistem
    a. Komponen (*Component*)

    Suatu sistem terdiri dari jumlah komponen yang saling berinteraksi, bekerja sama membentuk satu kesatuan. Komponen-komponen sisten dapat berupa suatu subsistem atau bagian-bagian sistem. Setiap subsistem

mempunyai sifat-sifat dari sistem untuk menjalankan suatu fungsi tertentu dan memengaruhi proses sistem secara keseluruhan. Suatu sistem dapat mempunyai proses sistem yang lebih besar yang disebut *supra sistem*.

b. Batas Sistem (*Boundary*)

Batas sistem merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lain atau dengan lingkungan luarnya. Batas sistem ini memungkinkan suatu sistem dipandang sebagai suatu kesatuan, karena dengan batas sistem ini fungsi dan tugas dari subsistem yang satu dengan yang lain berbeda tetapi tetap saling berinteraksi. Batas suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

c. Lingkungan Luar Sistem (*Environment*)

Environment merupakan segala sesuatu yang berada di luar batas sistem yang memengaruhi operasi dari suatu sistem. Lingkungan luar sistem ini dapat bersifat menguntungkan atau merugikan. Lingkungan luar yang menguntungkan harus dipelihara dan dijaga agar tidak hilang pengaruhnya, sedangkan lingkungan luar yang merugikan harus dimusnahkan atau dikendalikan agar tidak mengganggu operasi sistem.

d. Penghubung Sistem (*Interface*)

Merupakan media penghubung antara satu subsistem dengan subsistem yang lainnya untuk membentuk satu kesatuan sehingga sumber-sumber daya mengalir dari subsistem yang satu ke subsistem yang lainnya. Dengan kata lain, *output* dari suatu subsistem akan menjadi *input* dari subsistem yang lainnya.

e. Masukan Sistem (*Input*)

Merupakan energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*Maintenance Input*) adalah energi yang dimasukkan supaya sistem tersebut dapat beroperasi. Masukan sinyal (*Signal Input*) adalah energi yang diproses untuk didapatkan keluaran.

f. Keluaran Sistem (*Output*)

Merupakan hasil dari energi yang diolah oleh sistem, meliputi *output* yang berguna contohnya informasi yang dikeluarkan oleh komputer. Dan *output* yang tidak berguna dikenal sebagai sisa pembuangan, contohnya panas yang dikeluarkan komputer.

g. Pengolah Sistem (*Process*)

Merupakan bagian yang memproses masukan untuk menjadi keluaran yang diinginkan contoh CPU pada komputer.

h. Tujuan Sistem (*Goal*)

Setiap sistem mempunyai tujuan ataupun sasaran yang memengaruhi *input* yang dibutuhkan dan *output* yang dihasilkan. Dengan kata lain, suatu sistem itu mengenai sasaran atau tujuannya. Jika sistem tidak mempunyai sasaran, maka operasi sistem tidak akan ada gunanya.

(Asbon Hendra; 2012 : 157)

II.1.2. Informasi

Informasi adalah data yang telah diklasifikasikan atau diolah atau diinterpretasikan untuk digunakan dalam proses pengambilan keputusan.

Informasi dapat mengenai data mentah, data tersusun, kapasitas sebuah saluran komunikasi dan lain sebagainya. (Tata Sutabri ; 2012 : 29).

II.1.3. Sistem Informasi

Sistem informasi adalah suatu sistem didalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi harian yang mendukung fungsi operasi organisasi yang bersifat manajerial dengan kegiatan strategi dari suatu organisasi untuk dapat menyediakan kepada pihak luar dengan laporan-laporan yang diperlukan (Tata Sutabri ; 2012 : 46).

II.2. Penerapan Data Mining

II.2.1. Data Mining

Data Mining merupakan proses yang memperkerjakan satu atau lebih teknik pembelajaran komputer (*machine learning*) untuk menganalisis dan mengekstraksi pengetahuan (*knowledge*) secara otomatis. Definisi lain diantaranya adalah pembelajaran berbasis induksi *(induction-based learning)* adalah proses pembentukan definisi – definisi konsep umum yang dilakukan dengan cara mengobservasi contoh – contoh spesifik dari konsep – konsep yang akan dipelajari. Sistem data mining berdasarkan pada basis data yang cenderung dinamis, tidak lengkap, ber-noise dan besar. Permasalahan lain muncul sebagai akibat dari kecukupan dan relevansi dari informasi yang disimpan.

Basis data sering kali didesain untuk tujuan yang berbeda dari data mining dan kadangkala properti atau atribut yang akan menyederhanakan pekerjaan

pembelajaran tidak tersedia atau tidak dapat dimintai dari dunia nyata. Data yang tidak meyakinkan menyebabkan permasalahan karena jika ada atribut – atribut esensial bagi pengetahuan tentang domain aplikasi tidak ada dalam data tidak memungkinkan untuk menemukan pengetahuan yang tepat mengenai domain yang diberikan. Sebagai contoh, kita tidak dapat mendiagnosa malaria dari basis data pasien jika basis data tersebut tidak mengandung jumlah sel darah merah pasien. (Fajar Astuti Hermawati / Andi ; 2011 : 03).

II.2.2. Penerapan Data Mining

Data Mining berisi pencarian *trend* atau pola yang diinginkan dalam database yang besar untuk membantu pengambilan keputusan di waktu yang akan datang. Harapanya, perangkat *data mining* mampu mengenali pola – pola ini dalam data dengan masukan yang minimal. Pola – pola ini dikenali oleh perangkat tertentu yang dapat memberikan suatu analisa data yang berguna dan berwawasan yang kemudian dapat dipelajari dengan lebih teliti, yang mungkin saja menggunakan perangkat pendukung keputusan yang lainya.

II.2.3. Tujuan Data Mining

Lingkup *Data Mining* dapat dijelaskan merupakan proses iteratif dan interaktif untuk menemukan pola atau model baru yang sahih (sempurna), bermanfaat dan dapat dimengerti dalam suatu *database* yang sangat besar *(massive databases*). manfaat yang didapat dari penerapan *data mining*. Manfaat atau tujuan Penerapan Data Mining adalah sebagai berikut :

1. Sahih : Dapat digeneralisasi untuk masa yang akan datang.
2. Baru : Apa yang sedang tidak diketahui.
3. Bermanfaat : Dapat digunakan untuk suatu tindakan.
4. Iteratif : Memerlukan sejumlah proses yang diulang.
5. Interaktif : Memerlukan interaksi manusia dalam prosesnya.

(Fajar Astuti Hermawati / Andi ; 2011 : 03)

II.3. Bagian Data Mining

II.3.1 Data Warehouse

Jurnal khusus Penerapan *Data Mining* yang berpotensi tinggi jika data yang tepat dikumpulkan dan disimpan dalam sebuah gudang data (*data warehouse*). Sebuah gudang data merupakan suatu sistem manajemen basis data relasional (RDMS) yang didesain khusus untuk memenuhi kebutuhan akan sistem pengolahan transakasi. *Data Warehouse*, secara bebas dapat didefinisikan sebagai tempat penyimpanan data terpusat yang dapat di-*query* untuk manfaat bisnis. (Fajar Astuti Hermawati / Andi ; 2011 : 06)

II.3.2 Operasi Data Mining

Operasi *data mining* menurut sifatnya dibedakan menjadi dua, yaitu bersifat (1) prediksi (*prediction driven)* untuk menjawab pertanyaan apa dan sesuatu yang bersifat remang – remang atau transparan. Operasi prediksi digunakan untuk validasi hipotesis, *querying* dan pelaporan (misal : *spreadsheet* dan *pivot tabel*), analisis multidimensi (*dimensional summary*); OLAP (*Online*

Analytic Processing) serta analisis statistik. (2) Penemuan (discovery driven) bersifat transparan dan unuk menjawab pertanyaan ”mengapa?”. Operasi penemuan digunakan untuk analisis data eksplorasi, pemodelan prediktif, segmentasi database, analisis keterkaitan (*link analysis*) dan deteksi deviasi. (Fajar Astuti Hermawati / Andi ; 2011 : 05)

II.3.3. Karakteristik Data Warehouse

Menurut Bill Inmon, pemilik Building the Data Warehouse dan ahli yang mendalami konsep data warehouse, ada empat karakteristik data warehouse yaitu :

1. *Subject-oriented* : Data diorganisaikan menurut subyek dari aplikasi, misalnya sebuah perusaan asuransi menggunakan data *warehouse* yang mengorganisasi data mereka sebagai kastemer, premi dan klaim, dari pada dengan produk – produk berbeda (otomotif, jiwa, dsb).
2. *Integrated* : Ketika data menempati aplikasi – aplikasi yang terpisah dalam lingkungan operasional, pengkodean data seringkali tidak konsisen.
3. *Time Variant* : Data *Warehouse* terdiri dari suatu tempat untuk menyimpan data yang berusia 5 sampai 10 tahun atau lebih lama, untuk digunakan sebagai komparasi, *trend* dan peramalan. Data – data ini tidak di-*update.*
4. *Non-volatile :* Data yang tidak di-*update* sesudah mereka memasukkan data *warehaouse*, tetapi hanya dimuat dan diakses.

   (Fajar Astuti Hermawati / Andi ; 2011 : 07)

II.3.4. Proses Dalam Data Warehouse

Tahap pertama dalam data *warehousing* adalah menyekat informasi operasional sekarang. Misalnya menjaga keamanan dan integrasi aplikasi OLTP *mission-critical* saat kita mengakses basis data yang lebih luas. Hasil basis data atau data *warehouse* mungkin menghabiskan ratusan *gigabytes* dari ruan disk. Apa yang diperlukan kemudian adalah teknik efisien untuk menyimpan dan mengambil kembali sejumlah informasi secara besar – besaran. Organisasi – organisasi yang besar menentukan bahwa hanya sistem pengolahan paralel memberikan *bandwidth* yang cukup.

(Fajar Astuti Hermawati / Andi ; 2011 : 07-08)

II.3.5. Data Warehouse dan Sistem OLTP

Sebuah basis data yang dibangun untuk pengolahan transaksi secara *online,* OLTP, secara umum dipandang tidak cocok untuk *data warehouse* dikarenakan mereka didesain dengan suatu kumpulan kebutuhan yang berbeda, yaitu memaksimalkan kapasitas transaksi dan secara khusus mempunyai ratusan tabel dalam urutan yang tidak membatasi *user, data warehouse* dipandang dalam proses *query* sebagi lawan dari proses transaksi. Sistem OLTP tidak dapat menjadi tempat penyimpanan dari data fakta dan histori untuk analisa bisnis. Sistem ini tidak dapat menjawab secara cepat *query* dan pengambilan kembali secara cepat hampir tidak mungkin. Data yang tidak konsisten dan berubah, duplikasi masukan yang ada, masukan yang hilang yang dan tidak adanya data histori yang diperlukan untuk menganalisa *trend.* Pada dasarnya OLTP menawarkan sejumlah

besar data mentah yang tidak mudah dipahami. *Data warehouse* menawarkan kemampuan untuk mengambilan kembali dan menganalisa informasi secara cepat mudah. (Fajar Astuti Hermawati / Andi ; 2011 : 09-10)

Data Warehouse mempunyai kesamaan dengan OLTP seperti terliha pada Tabel II.1 berikut :

Tabel II.1. Persamaan dan Perbedaan OLTP dan Data Warehose

	OLTP	Data Warehouse
Purpose	Ran day-to-day operations	Information retrieval and analysis
Structure	RDMS	RDMS
Data Model	Normalised	Multi -dimensional
Access	MySQL	MySQL plus data analysis extensions
Type of Data	Data that runs the bussiness	Data that analysis the business
Condition of Data	Changing, incomplete	Historical, descriptive

(Sumber : Data Mining ; 2013 : 10)

II.4. MySQL

II.4.1. Database MySQL

Database didefinisikan sebagai kumpulan data yang terintegrasi dan diatur sedemikian rupa sehingga data tersebut dapat dimanipulasi, diambil, dan dicari secara cepat. Selain berisi data, database juga berisi metadata. Metadata adalah data yang dijelaskan tentang struktur dari data itu sendiri. Database memiliki beberapa model, diantaranya model rasional. Dalam model rasional, tabel-tabel yang terdapat dalam suatu database idealnya harus saling berelasi. (Budi Raharjo (2011 : 3-4).

MySQL merupakan software RDBMS (atau server database) yang dapat mengelolah database dengan sangat cepat, dapat menampung data dalam jumlah yang sangat besar, dapat diakses oleh banyak user (multi-user), dan dapat melakukan suatu proses secara singkron atau berbarengan (multi-threaded). MySQL banyak digunakan diberbagai kalangan untuk melakukan penyimpanan dan pengolahan data, mulai dari kalangan akademis sampai ke industri. MySQL sebagai produk open source di bawah GNU General Public License (gratis) atau dapat membeli lisensi dari versi komersialnya. (Budi Raharjo (2011 : 21-22).

II.4.2 Basis Data

Basis data adalah kumpulan data yag saling berhubungan yang disimpan/ diorganisasi secara bersama, dalam bentuk sedemikian rupa, dan tanpa redudansi (pengulangan) tidak perlu supaya dapat dimanfaatkan kembali dengan cepat dan mudah untuk memenuhi berbagai kebutuhan. (Ema Utami, Anggit Dwi Hartanto (2012 : 3)

Sistem basis data dapat terbagi dalam beberapa komponen, yaitu :

1. Data

   Merupakan informasi yang disimpan dalam suatu struktur tertentu yang terintegrasi.

2. *Hardware*

   Merupakan perangkat keras berupa komputer dengan media penyimpanan sekunder yang digunakan untuk menyimpan data karena pada umumnya basis data memiliki ukuran yang besar.

3. Sistem Operasi

   Program yang mengaktifkan/memfungsikan sistem komputer, mengendalikan seluruh sumber daya dalam komputer dan melakukan operasi-operasi dasar dalam komputer yang meliputi operasi *Input Output* (IO), pengelolaan *file* dan sebagainya.

4. Basis Data

   Basis data sebagai inti dari sistem basis data. Basis data menyimpan data serta struktur sistem basis data baik untuk entitas maupun objek-objeknya secara detail.

5. *Database Management System (DBMS)*

   Merupakan perangkat lunak yang digunakan untuk melakukan pengelolaan basis data.

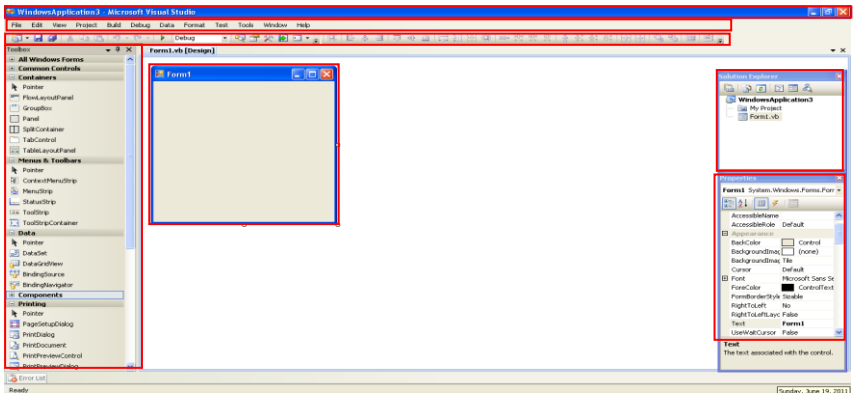
II.4.3 Sekilas Tentang Microsoft Visual Studio 2010

Bahasa Pemrograman **Microsoft Visual Basic.Net** adalah sebuah bahasa pemrograman tingkat tinggi untuk **Microsoft.NET Framework.** Walaupu**n VB.NET** ini memang dibuat supaya mudah dipahami dan dipelarajari, namun bahasa pemrograman ini juga cukup powerful untuk memenuhi kebutuhan dari programmer yang berpengalaman.Bahasa pemrograman**Visual Basic .NET** mirip dengan bahasa pemrograman **Visual Basic**, namun keduanya tidak sama.

Bahasa pemrograman **Visual Basic .NET** memiliki struktur penulisan yang mirip dengan Bahasa Inggris, dimana hal ini juga menyebabkan kemudahan dalam membaca dan mengerti dari sebuah kode **Visual Basic .NET**. Dimana dimungkinkan, kata ataupun frasa yang memiliki arti digunakan, dan bukanya menggunakan singkatan, akronim ataupun special characters.

Pada intinya, **Visual Basic .Net** ini adalah sebuah bahasa pemrograman yang berorientasi pada object, yang bisa dianggap sebagai evolusi selanjutnya dari bahasa pemrograman **Visual Basic** sederhana.

Adapun gambar Interface **Microsoft Visual Basic.Net 2010** dapat dilihat pada Gambar II.1. berikut ini :



Gambar II.1. Interface Microsoft Visual Basic.Net 2010

(Sumber : Buku Pintar VB.NET)

Berikut penjelasan tentang komponen-komponen **Microsoft Visual Basic .Net** :

1. *Title Bar*, berfungsi untuk menampilkan nama *project* yang sedang aktif atau sedang dikembangkan.
2. *Menu Bar*, berfungsi untuk pengelolaan fasilitas yang dimiliki oleh Visual Basic .Net 2010
3. *Tool Bar*, berfungsi untuk menampilan perintah khusus secara cepat.
4. *Form*, berfungsi untuk meletakkan objek yang berfungsi untuk meletakkan objek-objek yang terdapat pada *Toolbox*, yang digunakan dalam merancang sebuah tampilan program aplikasi.
5. *Tool Box*, berfungsi untuk menyediakan objek-objek atau komponen yang digunakan untuk merancang sebuah *form* pada program aplikasi.
6. *Solutation Explorer*, digunakan untuk menampilkan project beserta file-file pendukung yang terdapat pada sebuah program aplikasi.

7. *Properties Windows*, berfungsi untuk mengatur properties-properties pada objek (*setting object*) yang diletakkan pada sebuah *form*.

II.5. Konsep UML (*Unified Modelling Language*)

Unified Modelling Language (UML) merupakan alat komunikasi yang konsisten dalam mensupport para pengembang sistem saat ini. Sebagai perancang sistem, mau tidak mau pasti akan menjumpai *UML*, baik kita sendiri yang membuat atau sekedar membaca diagram *UML* buatan orang lain. (Prabowo Pudjo Widodo ; 2011 : 7)

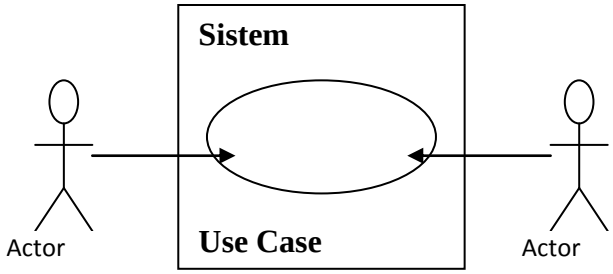
II.6. Diagram – diagram Pada Metode UML

1. Use Case Diagram

Use case adalah deskripsi fungsi dari sebuah sistem dari *perspektif* pengguna. *Use case* bekerja dengan cara deskripsikan tipikal interaksi antara *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem dipakai. Urutan langkah-langkah yang menerangkan antara pengguna dan sistem yang disebut *scenario*. Setiap *scenario* mendeskripsikan urutan kejadian. Setiap urutan diinisialisasi oleh orang, sistem yang lain, perangkat keras dan urutan waktu. Dengan demikian secara singkat bisa dikatakan *use case* adalah serangkaian *scenario* yang digabungkan bersama-sama oleh pengguna tujuan umum pengguna.

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

Model *use case* adalah bagai dari model *requirement*. Termasuk disini adalah problem domain object dan penjelasan tentang *user interface*. *Use case* memberikan spesifikasi fungsi-fungsi yang ditawarkan oleh sistem dari *persfectif user*. Notasi *use case* menunjukkan 3 aspek dari sistem yaitu *actor use case* dan *system / sub system boundary*. *Actor* mewakili peran orang, *system* yang lain atau alat ketika berkomunikasi dengan *use case*. Ilustrasi *actor*, *use case* dan *system* ditunjukkan pada gambar II.2. berikut ini :



Gambar II.2. *Use Case* Diagram

(Sumber : Prabowo Pudjo Widodo ; 2011 : 17)

Untuk mengidentifikasi *actor*, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. *Actor* adalah *abstraction* dari orang dan sistem yang lain mengaktifkan fungsi dari target sistem. Orang atau sistem bila muncul dalam beberapa peran. Perlu dicatat bahwa *actor* berinteraksi dengan *use case*, tetapi tidak memiliki kontrol atas *use case*.

Use case adalah abstraksi dari interaksi antara sistem dan *actor*. Oleh kerena itu sangat penting untuk memilih abstraksi yang cocok. *Use case* dibuat berdasarkan keperluan *actor*. *Use case* harus merupakan ’apa’ yang dikerjakan *software* aplikasi, bukan ’bagaimana’ *software* aplikasi mengerjakannya. Setiap use case harus diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan *actor*. Namun *use case* boleh terdiri dari beberapa kata dan tidak boleh ada dua *use case* yang memiliki nama yang sama.

2. Activity diagram

Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. *Activity diagram* mempunyai peran seperti halnya *flowchart*, akan tetapi perbedaanya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku *paralel* sedangkan *flowchart* tidak bisa

3. Class Diagram

Diagram kelas atau class diagram adalah inti dari proses pemodelan objek baik *forward engineering* maupun *reverse engineering* memanfaatkan diagram ini. *Forward engineering* adalah proses perubahan model menjadi kode program sedangkan *reverse engineering* sebaliknya merubah kode program menjadi model. (Prabowo Pudjo Widodo ; 2011 : 37)

Kelas memiliki apa yang disebut *Atribut* dan metode atau operasi :

a. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas

b. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas

Susunan kelas suatu sistem yang baik pada diagram kelas sebaiknya memiliki jenis-jenis kelas berikut:

a. Kelas main, kelas yang memiliki fungsi awal dieksekusi ketika sistem dijalankan.
b. Kelas yang menangani tampilan sistem, kelas yang mendefinisikan dan mengatur tampilan ke pemakai.
c. Kelas yang diambil dari pendefinisian *use case*, kelas yang menangani fungsi-fungsi yang harus ada diambil dari pendefinisian *use case*.
d. Kelas yang diambil dari pendefinisian data, kelas yang digunakan untuk memegang atau membungkus data menjadi sebuah kesatuan yang diambil maupun akan disimpan ke basis data.

Jenis-jenis kelas diatas juga dapat digabungkan satu sama lain sesuai dengan pertimbangan yang dianggap baik asalkan fungsi-fungsi yang sebaiknya ada pada struktur kelas tetap ada. Susunan kelas juga dapat ditambahkan kelas utilitas seperti koneksi ke basis data, membaca *file* teks, dan lain sebagainya sesuai kebutuhan.

Dalam mendefinisikan metode yang ada di dalam kelas perlu memperhatikan apa yang disebut dengan *cohension* dan *coupling*. *Cohension* adalah ukuran seberapa dekat keterkaitan instruksi di dalam sebuah metode terkait satu sama lain sedangkan *coupling* adalah ukuran seberapa dekat keterkaitan instruksi antara metode yang satu dengan yang lain dalam sebuah kelas.

Berikut adalah simbol-simbol yang ada pada diagram kelas Tabel II.2. berikut ini :

Tabel II.2. Simbol-Simbol *Class Diagram*

Simbol	Deskripsi
Kelas **nama_kelas** +atribut +operasi()	Kelas Pada struktur system
Antarmuka / *interface* nama_interface	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi / *association*	Relasi antar kelas dengan maksna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *directed association*	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya disertai dangan *multiplicity*
generalisasi	Relasi antar kelas dengan makna generalisasi – spesialisasi (umum khusus)
Keberuntungan /*dependency*	Relasi antar kelas dengan makna kebergantungan antar kelas
Agregasi /*aggregation*	Relasi antar kelas dengan makna

(Sumber : Prabowo Pudjo Widodo ; 2011 : 37)

4. *Squence Diagram*

Sequence Diagram digunakan untuk menggambarkan perilaku pada sebuah sekenario. Diagram ini menunjukkan sebuah contoh objek dan pesan yang diletakkan diantara objek-objek ini didalam *use case*.

Komponen utama *Squence diagram* terdiri dari atas objek yang dituliskan dengan kotak segiempat bernama. *Messege* diwakili oleh garis dengan tanda panah dan waktu yang ditunjukkan dengan *progress vertical.*

a. Objek / *participant*

Objek diletakkan di dekat bagian atas diagram dengan urutan dari kiri ke kanan. mereka diatur dalam urutan guna menyederhanakan diagram. Setiap *participant* dihubungkan garis titik-titik yang disebut *lifeline.* Sepanjang *lifeline* ada kotak yang disebut *activation*. *Activation* mewakili sebuah eksekusi operasi dari *participant*. Panjang kotak ini berbanding lurus dengan durasi *activation*. *Activation* mewakili sebuah eksekusi operasi dari participant. Panjang kotak ini berbanding lurus dengan durasi *activation.* Bentuk *participant* dapat dilihat pada Gambar II.3. berikut ini :

Gambar II.3. Bentuk *Participant*

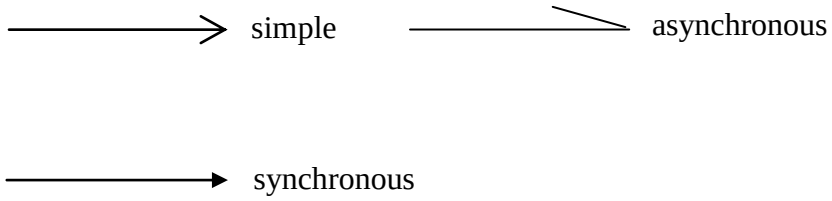
(Sumber : Prabowo Pudjo Widodo ; 2011 : 175)

b. *Messege*

Sebuah *messege* bergerak dari suatu *participant* ke *participant* yang lain dan dari *lifeline* ke *lifeline* yang lain. Sebuah *participant* bisa mengirim sebuah *message* kepada dirinya sendiri.

Sebuah *message* bisa jadi *simple*, *synchronous* atau *asynchoronous*. *message* yang *simple* adalah sebuah perpindahan (transfer), contoh dari satu *participant* ke *participant* yang lainnya.

Simbol *message* pada *squnence diagram* dapat dilihat pada gambar II.4. berikut ini :



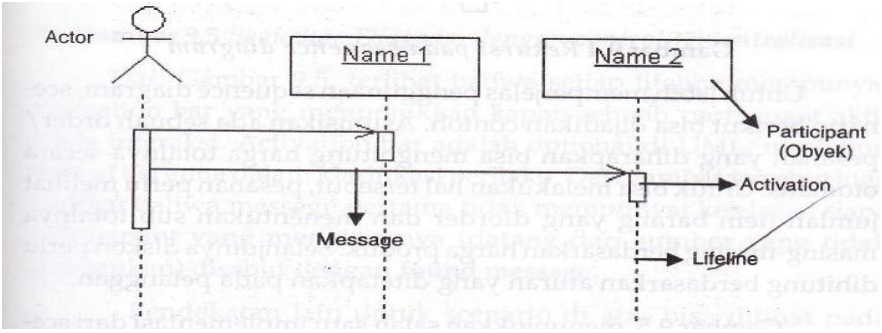
Gambar II.4. Bentuk *Messege*

(Sumber : Prabowo Pudjo Widodo ; 2011 : 179)

c. *Time*

Time adalah diagram yang mewakili waktu pada arah vertikal. Waktu dimulai dari atas ke bawah. *Message* yang lebih dekat dari atas akan dijalankan terlebih dahulu dibanding *message* yang lebih dekat kebawah.

Terdapat dua dimensi pada *squence diagram* yaitu dimensi dari kiri ke kanan menunjukkan tata letak participant dan dimensi dari atas ke bawah menunjukkan lintasan waktu. Simbol-simbol yang ada pada *squence diagram* ditunjukkan pada gambar II.5. berikut ini :



Gambar II.5. Bentuk *Time*

(Sumber : Prabowo Pudjo Widodo ; 2011 : 189)

II.7. Konsep Sistem Database

Database adalah kumpulan data terhubung *(interrelated data)* yang disimpan secara bersama-sama pada suatu media, tanpa mengatap satu sama lain atau tidak perlu suatu kerangkapan data *(controlled redundancy)* dengan cara tertentu sehingga mudah digunakan atau ditampilkan kembali; dapat digunakan oleh satu atau lebih program aplikasi secara optimal; data disimpan tanpa mengalami ketergantungan pada program yang akan menggunakannya; data disimpan sedemikian rupa sehingga penambahan, pengambilan, dan modifikasi dapat dilakukan dengan mudah dan terkontrol. Sementara itu, sistem *database* adalah sekumpulan database yang dapat dipakai secara bersama-sama, *personal-personal* yang merancang dan mengelola *database*, teknik-teknik untuk merancang dan mengelola *database*, serta *computer* untuk mendukungnya.

II.8. Normalisasi

Normalisasi merupakan proses pengelompokan elemen data menjadi tabel yang menunjukkan entitas sekaligus relasinya. (Ema Utami dan Anggit Dwi Hartanto ; 2012 : 40).

Tahap normalisasi terdiri dari beberapa bentuk :

1. Bentuk Tidak Normal (*Unnormalized Form*)

Bentuk ini merupakan kumpulan data yang akan direkam, tidak ada keharusan mengikuti suatu format tertentu, dapat saja data tidak lengkap atau terduplikasi. Data dikumpulkan apa adanya sesuai dengan kedatangannya.

2. Bentuk Normal Kesatu (1NF/ *First Normal Form*)

Bentuk normal kesatu mempunya ciri : setiap data dibentuk dalam *file file* (*file* datar/rata), data dibentuk dalam satu *record* demi *record* dan nilai dari *field* berupa ”atomic value”. Tidak ada set atribut yang berulang atau atribut bernilai ganda (*multivalue*). Tiap *field* hanya satu pengertian, bukan merupakan kumpulan kata yang mempunyai arti mendua, hanya satu arti saja dan juga bukan pecahan kata sehingga artinya lain. Atom adalah zat terkecil yang masih memiliki sifat induknya, bila dipecah lagi, maka ia tidak memiliki sifat induknya.

3. Bentuk Normal Kedua (2NF/ *Second Normal Form*)

Bentuk normal kedua memiliki syarat : bentuk data telah memenuhi kriteria bentuk normal kesatu. Atribut bukan kunci haruslah bergantung fungsi pada kunci utama/*primary key* sehingga untuk membentuk normal kedua haruslah sudah ditentukan kunci *field*. Kunci *field* haruslah unik dan dapat mewakili atribut lain yang menjadi anggotanya. Tahap normalisasi terdiri dari beberapa bentuk.

4. Bentuk Normal Ketiga (3NF/ *Third Normal Form*)

Untuk menjadi bentuk normal ketiga, relasi haruslah dalam bentuk normal kedua dan semua atriut dalam primer tidak punya hubungan yang transif. Dengan kata lain, setiap atribut bukan kunci haruslah bergantung hanya pada *primary key* dan pada *primary key* secara menyeluruh. Contoh pada bentuk kedua di atas termasuk juga bentuk normal ketiga seluruh atribut yang ada disitu bergantung penuh pada kunci primernya.

5. Bentuk Normal *Boyce Codd* (BCNF/ *Boyce Codd Normal Form*)

Boyce Codd Normal *Form* mempunyai paksaan yang lebih kuat dari pada bentuk normal ketiga. Untuk menjadi BCNF, relasi harus dalam bentuk normal kesatu dan setiap atribut harus bergantung fungsi pada atribut *superkey*.

II.9. *Entity Relationship Diagram* (ERD)

ERD merupakan notasi grafis dalam pemodelan data *konseptual* yang mendeskripsikan hubungan antar penyimpanan. *ERD* juga merupakan gambaran yang menghubungkan antara objek satu dengan objek yang lainnya dalam dunia nyata. (Ema Utami dan Anggit Dwi Hartanto ; 2012 : 18)

ERD menggunakan sejumlah notasi dan simbol untuk menggambarkan struktur dan hubungan antar dua data. Pada dasarnya ada 3 macam simbol yang digunakan, yaitu :

1. *Entity*

Entity adalah suatu objek yang dapat diidentifikasikan dalam lingkungan pemakai, sesuatu yang penting bagi pemakai dalam konteks sistem yang akan dibuat. Sebagai contoh adalah barang, pemasok, pekerja dan lain-lain.

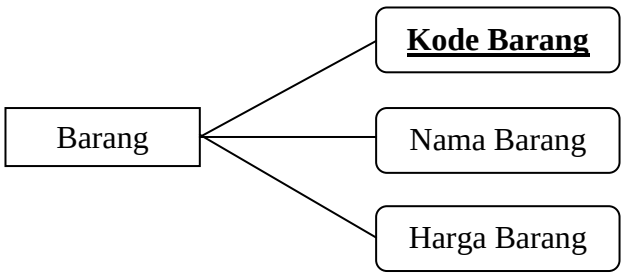
Seandainya adalah A maka barang A adalah isi dari barang, sedangkan jika B adalah seorang pelanggan maka B adalah isi dari pelanggan. Karena itu harus dibedakan antara entitas sebagai bentuk umum dari deskripsi tertentu dan isi entitas seperti A dan B dalam contoh diatas. Entitas dapat digambarkan dalam bentuk persegi empat. Gambar II.6. berikut ini :

Gambar II.6. Entitas

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 19)

2. *Atribut*

Entitas mempunyai elemen yang disebut *atribut* dan berfungsi mendeskripsikan karakter *entitas*, misalnya atribut nama barang dari *entitas* barang. Setiap *ERD* bisa berisi lebih dari satu *atribut*. *Entitas* digambarkan dalam bentuk elips Gambar II.7. berikut ini :

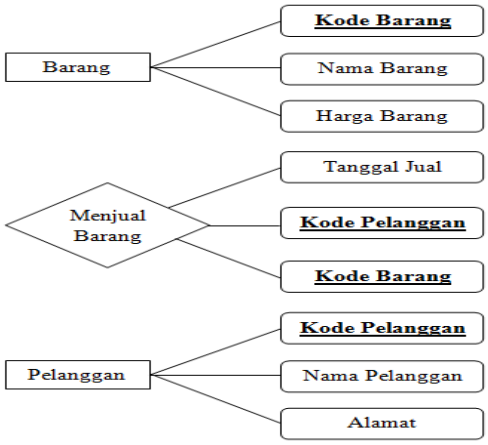


Gambar II.7. Atribut

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 20)

3. Hubungan / *relationship*

Sebagimana halnya *entitas*, hubungan pun harus dibedakan antara hubungan atau bentuk hubungan antar entitas dengan isi dari hubungan itu sendiri. Misalnya dalam kasus hubungan antar entitas barang dan entitas pelanggan adalah menjual barang, sedangkan isi hubungannya dapat berupa tanggal jual atau yang lainnya. *Relationship* digambarkan dalam bentuk intan (*diamons*) pada Gambar II.8. berikut ini :

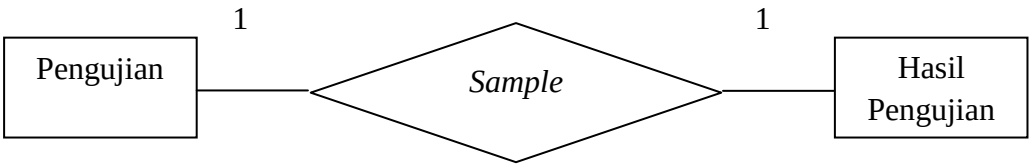


Gambar II.8. *Relationship*

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 21)

Jenis-jenis hubungan:

a. Satu ke satu, misalnya suatu perusahaan mempunyai aturan saperti pengujian *sample* dimana mempunyai proses – proses pengaturan pengujian seperti Gambar II.9. berikut ini :



Gambar II.9. *Relational* 1 to 1

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 24)

b. Satu ke banyak atau banyak ke satu, misalya suatu perusahaan selalu berasumsi bahwa satu proses pengujian dapat menghasilkan hasil pengujian yang dimana *sample* beras tersebut di ketahui nilai kadar klorinnya, dan di ketahui dalam proses threshold tersebut Seperti Gambar II.10. berikut ini :

Gambar II.10. *Relational* 1 to *Many*

(Sumber : Ema Utami dan Anggit Dwi Hartanto ; 2012 : 25)

II.10. Pohon Keputusan *(Decision Tree)*

Salah satu teknik klasifikasi yang akan dipelajari pada pohon keputusan *(decision tree).* Pohon *(tree)* adalah sebuah struktur data yang terdiri dari simpul *(node)* dan rusuk *(edge).* Simpul pada sebuah pohon dibedakan menjadi tiga, yaitu

simpul akar *(root node)*, simpul percabangan / *internal (branch / internal node)* dan simpul daun *(leaf node).*

Pohon keputusan merupakan representasi sederhana dari teknik klasifikasi untuk sejumlah kelas berhingga, dimana simpul internal maupun simpul akar ditandai dengan nama atribut, rusuk – rusuknya diberi label nilai atribut yang mungkin dan simpul daun ditandai dengan kelas – kelas yang berbeda. Contoh pohon keputusan yang digunakan untuk mengklasifikasikan pengujian *sample* beras dapat dilihat pada Tabel II.3 berikut ini :

Tabel II.3. Pohon Keputusan untuk Pengujian Sample Beras

NO	Id Beras	Kadar Nilai Klorin	Proses Masuk
1	B001	A1	Penggilingan Beras
2	B002	A2	Penggilingan Beras
3	B003	A1	Pedagang Beras
4	B004	A3	Penggilingan Bears
5	B005	B2	Importir Beras
6	B006	A3	Importir Beras
7	B007	A3	Penggilingan Beras
8	B008	A2	Penggilingan Beras
9	B009	A2	Importir Beras
10	B010	A2	Penggilingan Beras
11	B011	B2	Penggilingan Beras

(Sumber : Fajar Astuti Hermawati / Andi ; 2011 : 58)

II.11. Klarifikasi *(Decision Tree)*

Pengklarifikasian adalah suatu Proses *Decision Tree* dengan pemilihan atribut pada algoritma induksi pohon keputusan menggunakan ukuran berdasarkan *entropy* yang dikenal dengan *information gain* sebagai sebuah *heuristic* untuk memilih atribut yang merupakan bagian terbaik dari contoh ke dalan kelas. Semua atribut adalah bersifat katagorikal yang bernilai diskrit. Atribut dengan nilai kontinyu harus didiskritkan.

Ada beberapa Contoh Model Klarifikasi *Decision Tree* antara lain adalah :

Model Overfitting

yaitu merupakan data-set dua dimensi yang titik-titiknya berada pada dua kelas yang berbeda yaitu kelas 0 sebanyak 1200 titik dan kelas = sebanyak 1800.30% dari titik-titik tersebut digunakan sebagai data *training* dan 70% titik digunakan sebagai data tes.

Model Noise Overfitting

yaitu merupakan data training dan data test yang digunakan untuk mengklarifikasi objek menjadi dua kelas, yaitu kelas mamalia dan non mamalia dimana *Bat* dan *Whale* masuk kelas non-mamalia padahal seharusnya merupakan mamalia yang mempunyai sifat yang kontradiktif dengan sifat – sifat dalam data training.

Model Overfitting Karena Data Yang Tidak Mencukupi

Yaitu Jumlah *Record* yang tidak mencukupi dari data *training* menyebabkan *decision tree* memprediksi contoh test menggunakan *training records* yang lain yang tidak relevan untuk proses klarifikasi.