BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Adapun penelitian terkait yang akan digunakan sebagai sumber acuan yang relevan dan terkini yaitu:

Berdasarkan penelitian Roy Sumaryono (2014) dengan judul "Penerapan Metode *Trend Moment* dalam *Forecast* Penjualan Beton *Ready mix* Di PT. X, Mojokerto" Dalam penjualan beton *ready mix* sering terjadi fluktuasi penjualan beberapa mutu beton yang dipengaruhi oleh faktor bahan baku terutama semen dan pesaing. Sehingga menyebabkan perusahaan khususnya PT. X tidak bisa meramalkan penjualan beton *ready mix* dimasa yang akan datang. Penelitian ini bertujuan untuk mengidentifikasi dan menganalisis hasil ramalan penjualan beton *ready mix* dengan menggunakan metode ramalan *Trend Moment* dan merancang aplikasi peramalan penjualan beton *ready mix* dengan menggunakan metode *Trend Moment* untuk mengatasi kerugian dan tidak tercapainya target perusahaan. Hasil akhir dari peramalan penjualan beton *ready mix* PT. X dengan menggunakan metode *Trend Moment* pada bulan Januari 2015 yaitu cenderung meningkat atau mengalami *Trend* Positif dibandingkan dengan penjualan tahun lalu.

Menurutt Aan Suhartri Wahyono (2016) dengan judul Implementasi Metode *Trend Moment* Untuk Peramalan Penjualan Kubis" Permasalahan yang umum dihadapi penjual kubis adalah bagaimana meramalkan atau memperkirakan

penjualan kubis di masa mendatang berdasarkan data yang telah direkam sebelumnya. Peramalan tersebut sangat berpengaruh pada penjual kubis untuk sebagai bahan acuan memperkirakan penjualan kubis yang harus disediakan guna memenuhi permintaan pasaryang ada. Metode peramalan *Trend Moment* digunakan dalam peramalan penjualan kubis. Peramalan menggunakan data penjualan yang sudah ada selama 3 bulan, dengan data tersebut bisa diperoleh siklus penjualan yang akan dijadikan acuan sistem peralaman ini.

II.2. Uraian Teoritis

II.2.1. Sistem

Sistem adalah kumpulan dari bagian-bagian yang bekerja sama untuk mencapai tujuan yang sama. Definisi sistem adalah sekumpulan objek-objek yang saling berelasi dan berinteraksi serta hubungan antar objek bisa dilihat sebagai suatu kesatuan yang dirancang untuk mencapai suatu tujuan. Sistem adalah penggabungan dari bagian-bagian atau komponen-komponen yang terpisah-pisah dan disatukan menjadi satu rangkaian dan menjadi suatu fungsi yang baru (Aris, 2015).

II.2.1.1. Karakteristik Sistem

Untuk memahami atau mengembangkan suatu sistem, maka perlu membedakan unsur-unsur dari sistem yang membentuknya. Berikut adalah karakteristik sistem yang dapat membedakan suatu sistem dengan sistem lainnya yaitu:

1. Batasan (*boundary*) adalah Penggambaran dari suatu elemen atau unsur mana yang termasuk di dalam sistem dan mana yang di luar sistem.
2. Lingkungan (*environment*) adalah Segala sesuatu di luar sistem, lingkungan yang menyediakan asumsi, kendala, dan input terhadap suatu sistem.
3. Masukan (*input*) adalah Sumber daya (data, bahan baku, peralatan, energi) dari lingkungan yang dikonsumsi dan dimanipulasi oleh suatu sistem.
4. Keluaran (*output*) adalah Sumber daya atau produk (informasi, laporan, dokumen, tampilan layer Komputer, barang jadi) yang disediakan untuk lingkungan sistem oleh kegiatan dalam suatu sistem.
5. Komponen (*component*) adalah Kegiatan-kegiatan atau proses dalam sistem yang mentransformasikan input menjadi bentuk setengah jadi (output). Komponen ini bisa merupakan subsistem dari sebuah sistem.
6. Penghubung (*Interface*) adalah Tempat di mana komponen atau sistem dan lingkungannya bertemu atau berinteraksi.
7. Penyimpanan (*storage*) adalah Area yang dikuasai dan digunakan untuk penyimpanan sementara dan tetap dari informasi, energi, bahan baku, dan sebagainya. Penyimpanan merupakan suatu media penyangga di antara komponen tersebut bekerja dengan berbagai tingkatan yang ada dan memungkinkan komponen yang berbeda dari berbagai data yang sama (Aris : 2015).

II.2.2. Informasi

Informasi adalah data yang diolah menjadi bentuk yang lebih berguna dan lebih berarti bagi yang menerimanya.

1. Akurat (*Accurate*)

   Berarti informasi harus bebas dari kesalahan-kesalahan dan tidak bias atau menyesatkan. Akurat juga berarti informasi harus jelas mencerminkan maksudnya.

2. Tepat waktu (*Timelines*)

   Berarti informasi yang datang pada si penerima tidak boleh terlambat. Informasi yang sudah usang tidak akan mempunyai nilai logika karena informasi merupakan landasan dalam pengambilan keputusan.

3. Relevan (*Relevance*)

   Berarti informasi tersebut bermanfaat bagi pemakainya (Deppi Linda, 2016 : 62-63).

II.2.3. Sistem Informasi

Sistem informasi adalah suatu kumpulan dari komponen – komponen dalam perusahaan atau organisasi yang berhubungan dengan proses penciptaan dan pengaliran informasi. System informasi dapat juga didefinisikan sebagai suatu system yang menerima sumber data sebagai *input* dan mengolahnya menjadi produk informasi sebagai *output*. System informasi merupakan suatu system yang terdiri dari beberapa subsistem atau komponen *hardware, software* dan *brainware*, data dan prosedur untuk menjalankan *input,* proses, *output*,

penyimpanan dan pengontrolan yang mengubah sumber data menjadi informasi. Atau dapat juga didefinisikan sebagai suatu system di dalam suatu organisasi yang mempertemukan kebutuhan pengolahan transaksi, mendukung oprasi, bersifat manajerial, dan kegiatan strategi dari suatu organisasi dan menyediakan pihak luar tertentu dengan laporan – laporan yang diperlukan (Marimin ; 2012 : 18).

II.2.4. Peramalan

Ramalan (*forecast*) merupakan dugaan atau perkiraan mengenai terjadinya suatu kejadian atau peristiwa di waktu yang akan datang. Ramalan ini sangat berguna dalam berbagai bidang kehidupan, terutama dalam rangka perencanaan untuk mengantisipasi berbagai keadaan yang terjadi pada masa yang akan datang. Ramalan bisa dilakukan secara kualitatif maupun kuantitatif. Terkait dengan ramalan kuantitatif, metode peramalannya pada dasarnya dapat dibedakan atas:

1. Metode peramalan melalui analisis suatu variabel yang akan diperkirakan dengan variabel waktu, yang dikenal dengan metode hubungan deret waktu. Data yang digunakan adalah dataderet waktu (*time series*).
2. Metode peramalan melalui analisis pola hubungan antara variabel yang akan diperkirakan dengan variabel-variabel lain yang mempengaruhinya (waktu dan/ serta bukan waktu). Metode ini sering disebut metode hubungan sebab akibat (*causal method*). Data yang digunakan dapat berupa data time series maupun data *cross section*.

Terdapat berbagai jenis model/metode peramalan hubungan deret waktu. Diantaranya adalah: 1) *Model Linear*; 2) *Model Quadratic*; 3) *Model Exponential Growth*; 4) *Model S-Curve* (*PearlReed Logistic*); 5) *Model Moving Average*; 6) *Model Single Exponential Smoothing*; 7) *Model Double Exponential Smoothing*; 8) *Metode Winter*; 9) Model ARIMA (Junaidi : 2014).

II.2.5. Analisis Time Series

Suatu data *time series* dapat dilihat sebagai suatu representasi dari realisasi suatu variabel random yang biasanya mempunyai interval waktu yang sama dan diamati pada suatu periode tertentu. Data *time series* ini merupakan suatu deskripsi masa lampau dan digunakan untuk meramalkan masa depan, artinya kita berharap masa depan dapat dijelaskan dengan informasi yang ada pada masa lampau.kalau memang hal ini yang terjadi, kita dapat menawarkan suatu model matematik yang mampu merepresentasikan proses terjadinya data *time series* tersebut. Kemudian, kita gunakan model matematik ini untuk membuat suatu ramalan tentang masa depan. Kenyataannya, dalam kehidupan sehari-hari, sering kali kita dihadapkan pada keterbatasan informasi masa lalu sehingga kita tidak dapat membuat model yang dapat menceritakan masa lalu secara tepat. Oleh sebab itu, biasanya, yang dapat dilakukan hanyalah membuat model yang dekat dengan model yang sebenarnya. Sering kali pendekatan ini berdasarkan pada pengamatan terhadap data *time series*. (Ashari : 2016)

Ada 2 hal pokok yang harus diperhatikan dalam proses pembuatan peramalan yang akurat dan bermanfaat. Pertama ialah pengumpulan data yang

relevan berupa informasi yang dapat menghasilkan peramalan yang akurat. Kedua ialah pemilihan teknik peramalan yang tepat yang akan memanfaatkan informasi data yang diperoleh seoptimal mungkin. Setiap variabel yang terdiri dari data yang dikumpulkan, dicatat atau diobservasi sepanjang waktu yang berurutan disebut data runtut waktu (*time series*). Analisis runtut waktu dilakukan untuk menemukan pola pertumbuhan atau perubahan masa lalu, yang dapat digunakan untuk memperkirakan pola pada masa yang akan datang. Analisis ini cukup penting dalam proses peramalan dan membantu mengurangi kesalahan dalam peramalan tersebut. Dalam analisis runtut waktu terdapat 4 komponen yaitu:

a. *Trend*

Trend ialah perkembangan jangka panjang dalam suatu runtut waktu yang dapat digambarkan dengan sebuah garis lurus atau sebuah kurva kekuatan-kekuatan dasar yang menghasilkan atau mempengaruhi *trend* dari suatu seri adalah perubahan populasi, perubahan harga, perubahan teknologi, dan peningkatan produktivitas. Dalam analisis *trend* variabel bebasnya adalah waktu. Seorang peneliti harus memetakan data dalam bentuk aritmatika dan semilogaritma sebelum memilih persamaan *trend* berdasarkan bentuk umum dari grafik yang tampak. Jika grafik berbentuk garis lurus dalam skala aritmatika, maka peneliti akan menggunakan persamaan linier dalam analisis datanya. Jika data dinyatakan dalam bentuk semilogaritma dan terbentuk grafik dengan garis lurus, maka peneliti akan memilih model eksponensial dalam datanya. Metode untuk menjelaskan *trend linier* ialah metode kuadrat terkecil.

b. Variasi Siklis

Komponen siklis ialah suatu seri fluktuasi seperti gelombang atau siklus yang mempengaruhi keadaan ekonomi selama lebih dari satu tahun. Hal tersebut dapat dilihat dari perbedaan antara nilai yang diharapkan (*trend*) dengan nilai yang sebenarnya yaitu variasi residual yang berfluktusi sekitar *trend*. Komponen siklis dan tak beraturan dari data runtut waktu dapat diidentifikasi dengan cara menghilangkan pengaruh *trend*, metode ini disebut metode residual (*residual method*). Tahap metode residual tergantung pada dimulainya menganalisis dengan data tahunan, bulanan, atau kuartalan. Jika data yang digunakan ialah data bulanan atau kuartalan, maka pengaruh *trend* dan komponen-komponen musiman harus dihilangkan. Jika datanya ialah data tahunan, maka pengaruh *trend* yang dihilangkan.

c. Musiman

Fluktuasi musiman biasanya dijumpai pada data yang dikelompokan secara kuartalan, bulanan, atau mingguan. Variasi musiman ini menggambarkan pola perubahan yang berulang secara teratur dari waktu ke waktu. Komponen musiman runtut waktu diukur dalam bentuk angka indeks. Interpretasi angka indeks ini, yang mencerminkan besarnya pengaruh musiman untuk suatu segmen tahun tertentu, berkaitan dengan perbandingan nilai terhitung atau nilai yang diharapkan dari segmen tersebut (bulan, kuartal, dan sebagainya).

d. Fluktuasi tak beraturan

Komponen tidak beraturan terbentuk dari fluktuasi-fluktuasi yang disebabkan oleh peristiwa-peristiwa yang tidak terduga seperti perubahan cuaca,

pemogokan, perang, pemilihan umum, rumor perang, dan lain-lain. (Sri Nawangwulan : 2016).

II.2.6. *Trend Moment*

Metode *trend moment* adalah merupakan metode untuk mencari garis *trend* dengan perhitungan statistika dan matematika tertentu guna mengetahui fungsi garis lurus sebagai pengganti garis patah-patah yang dibentuk oleh data historis. Dalam penerapan metode *Trend Moment* dapat di lakukan dengan menggunakan data historis dari satu variabel, adapun rumus yang di gunakan dalam penyusunan dari metode ini (Muthia : 2016)

Dengan demikian pengaruh unsur subjektif dapat dihindarkan. Persamaan trend dengan metode *trend moment* adalah sebagai berikut (Imam Wahyudi : 2016) :

$Y = a + bX$

Dimana :

Y : nilai trend (Peramalan)

a : bilangan konstant

b : *slope* atau koefisien kecondongan garis *trend*

X : indeks waktu (x = 0, 1, 2, 3, …, n)

Untuk mencari nilai *a* dan *b* pada rumus diatas, digunakan dengan cara matematis dengan penyelesaiannya menggunakan metode subtitusi dan metode

eliminasi. Sedangkan untuk menghitung nilai a dan b digunakan rumus pada persamaan 1 dan persamaan 2. (Astuti, 2014) :

$$b = \frac{n\,(\sum XY) - (\sum X)(\sum Y)}{(\sum X^2) - (\sum X^2)} \quad \text{.....(1)}$$

$$a = \frac{\sum Y - b\,(\sum X)}{n} \quad \text{.....(2)}$$

Dimana :

ΣX : Jumlah kumulatif dari periode waktu

ΣY : Jumlah kumulatif data penjualan

Σ XY : Jumlah kumulatif dari jumlah periode dikalikan jumlah penjualan

n : Banyaknya periode waktu (bulan)

Setelah nilai ramalan yang telah diperoleh dari hasil peramalan dengan metode *Trend Moment* akan dikoreksi terhadap pengaruh musiman dengan menggunakan indeks musim dengan rumus :

Rata – Rata Permintaan Tertentu / rata – rata permintaan perbulan

Untuk mendapatkan hasil ramalan akhir setelah dipengaruhi indek musim maka akan menggunakan perhitungan sebagai berikut (Muthia. 2014) :

$$Y^* = \text{Indeks Musim} \times Y$$

Dimana :

Y* = Hasil ramalan dengan menggunakan metode *Trend Moment* yang telah dipengaruhi oleh indeks musim.

Y = Hasil ramalan dengan menggunakan *Trend Moment*.

II.2.7. Basis Data (*Database*)

Secara sederhana *Database* (basis data/ pangkalan data) dapat diungkapkan sebagai suatu pengorganisasian data dengan bantuan komputer yang memungkinkan data dapat diakses dengan mudah dan cepat (Kadir, 2004). Pengertian akses dapat mencakup pemerolehan data maupun pemanipulasian data seperti menambah serta menghapus data. Dengan memanfaatkan komputer, data dapat disimpan dalam media pengingat yang disebut *hard disk*. Dengan menggunakan media ini, keperluan kertas untuk menyimpan data dapat dikurangi. Selain itu, data menjadi lebih cepat untuk diakses terutama jika dikemas dalam bentuk *Database*.

Pengaplikasian *Database* dapat kita lihat dan rasakan dalam keseharian kita. *Database* ini menjadi penting untuk mengelola data dari berbagai kegiatan. Misalnya, kita bisa menggunakan mesin ATM (anjungan tunai mandiri/ *automatic teller machine*) bank karena bank telah mempunyai *Database* tentang nasabah dan rekening nasabah. Kemudian data tersebut dapat diakses melalui mesin ATM ketika bertransaksi melalui ATM. Pada saat melakukan transaksi, dalam konteks *Database* sebenarnya kita sudah melakukan perubahan (*update*) data pada *Database* di bank.

Ketika kita menyimpan alamat dan nomor telepon di HP, sebenarnya juga telah menggunakan konsep *Database*. Data yang kita simpan di HP juga mempunyai struktur yang diisi melalui formulir (*form*) yang disediakan. Pengguna dimungkinkan menambahkan nomor HP, nama pemegang, bahkan kemudian dapat ditambah dengan alamat *email*, alamat *web*, nama kantor, dan sebagainya.

Pemahaman tentang *Database* ini dapat didekatkan pada konsep akuntansi. Kita bisa umpamakan bahwa ketika kita melakukan proses akuntansi secara manual, kita menuliskan suatu catatan ke dalam lajur dan kolom buku. Mulai dari jurnal, buku besar, buku pembantu kita memasukkan catatan satu demi satu. Melihat buku akuntansi tersebut, sebenarnya kita sudah melihat konsep *Database*, yang jika dikelola dengan komputer masih diperlukan penyesuaian dalam membentuk kolom-kolomnya. (Mujilan : 2012:23)

II.2.7.1. Model *Database*

Model *Database* yang saat ini banyak digunakan adalah model *Database* relational. Imam (2008) menyebutkan "Model *Database* ini disusun dalam bentuk tabel dua dimensi yang terdiri dari baris (*record*) dan (*field*), pertemuan antara baris dengan kolom disebut item data (*data value*). Tabel-tabel yang ada dihubungkan (*relationship*) sedemikian rupa menggunakan *field-field* kunci (*key field*) sehingga dapat meminimalkan duplikasi data."

Model *Database relational* ini dapat kita kenal konsepnya mulai dari yang paling sederhana misalnya dengan penerapan program aplikasi *excel*. Meskipun untuk pengelolaan *Database* secara luas *excel* jarang digunakan dan kurang

mencukupi, namun untuk melihat konsep *Database* dan konsep membangunnya program ini dapat dimanfaatkan. *Excel* mempunyai baris yang disebut *raw* dan mempunyai kolom. Kemudian item data merupakan sel atau pertemuan antara baris dan kolom. Tabel-tabel dapat diumpakan apabila kita menggunakan tabel dalam suatu *sheet* tertentu. Data dari berbagai tabel dapat diambil dari tabel lain menggunakan perintah *look up* yang berdasarkan kode kunci tertentu. Kode kunci tersebut berada pada suatu kolom tertentu, yang dalam konsep *Database relational* disebut sebagai *key field* tadi. (Mujilan : 2012:24)

II.2.7.2. Struktur *Database*

Untuk memahami konteks *database* kita perlu memahami istilah dan hal-hal yang terkait dengan *database*. Dalam berbagai program aplikasi *database* terdapat kesamaan ataupun sedikit perbedaan di dalamnya. Seseorang yang mempelajari *database* dengan program aplikasi tertentu harus memperhatikan struktur dan karakteristik sesuai dengan bahasa dalam aplikasi tersebut. Namun demikian, secara umum terdapat karakteristik sebagai berikut:

1. Nama *file*

Nama *file* adalah nama yang digunakan untuk mengidentifikasi adanya data yang disimpan dalam komputer dan digunakan untuk pemanggilan data. *File* yang dikelola akan muncul dalam komputer dengan ekstensi sesuai dengan program aplikasinya. *File* tersebut dapat digunakan untuk menandakan adanya *file database*, ataupun *file table*. *Database* dan table akan saling terkait, meskipun cara menyimpan dalam komputer akan mengalami sedikit perbedaan pada

beberapa aplikasi. Misalnya Ms. *Access* akan menyimpan dengan *file* yang dapat kita lihat adalah *file Database*nya. Di program *MySql* nama *database* ini akan menjadi *folder*. Sementara di *FoxPro* nama *database* dapat menjadi *file* tersendiri. *Table* cara menyimpannya juga berbeda, dalam *Ms. Access* mungkin kita tidak melihat nama *table* secara kasat mata karena akan dikelola di dalam *file database*. Di dalam *MySql* kita bisa melihat beberapa nama *file* terkait dengan pengelolaan *table*. Dan di dalam *FoxPro table* ini dapat menjadi nama *file* terpisah dan dapat dikenali pula sebagai *free table*.

2. *Database*

Database sebenarnya merupakan nama untuk menampung berbagai *table* di dalamnya. Konsep ini akan sama dalam berbagai program aplikasi. Misalnya kita membangun *database* akuntansi dengan nama *database* "*akun_base*". Di dalam *akun_base* akan diorganisasi berbagai *table* yang terkait dengan kegiatan akuntansi misalnya *tabel* rekening, pelanggan, jurnal, buku induk, dan administrator program, dan sebagainya. Setiap data yang masuk tidaklah dicatat dalam *database*, namun di dalam masing-masing *table* yang sesuai.

3. *Table*

Table merupakan tempat untuk menyimpan data sesuai dengan kelompok data. Setiap isi table mengandung data yang mempunyai karakteristik dalam penggunaannya. Untuk mempermudah pengolahan biasanya pembangun *Database* mengkategorikan *table* sesuai dengan data isinya sebagai berikut :

a. *Master table*

Master table berisi data tentang hal-hal utama dalam kegiatan *database*. Table ini berisi *record* yang relatif permanen atau seringkali menjadi acuan ketika mengoperasikan transaksi. Dalam master tabel identitas *record* menjadi penting dan diusahakan merupakan data atau kode yang bersifat unik. Unik dapat diartikan bahwa tidak ada dalam satu table berisi kode yang sama. Disain kode menjadi penting di sini.

b. *Transaction table*

Tabel transaksi digunakan untuk menyimpan data dalam menjalankan suatu kegiatan atau bisnis. Data ini seringkali akan bertambah dalam kesehariannya ketika terjadi transaksi yang sesuai dengannya. Secara lebih mudah dapat dipahami dalam akuntansi seringkali mencatat transaksi dalam jurnal.

c. *Tabulation table*

Tabulasi data dapat digunakan untuk menyimpan data seperti halnya master data namun bersifat sebagai data pembantu ketika menginput formulir baik untuk data master maupun transaksi.

d. *Temporary table*

Temporary adalah data sementara yang digunakan untuk membantu ketika terjadi proses transaksi. Data ini dapat saja langsung dihapus ketika transaksi selesai terproses.

4. *Field*

Field adalah penanda untuk kolom data. Jika dalam *excel* penanda tersebut adalah kolom A, B, dan seterusnya, sementara dalam konsep *table* dalam

Database maka nama *field* memegang peranan penting. Dalam konsep table dalam *Database*, ketika memanggil dengan nama *field* tertentu maka data-data di dalamnya akan muncul. Pengolahan dapat dilakukan dengan membuat *filter*, misalnya berdasarkan kode tertentu, berdasarkan *record* tertentu.

Dalam mengatur setting *field*, biasanya akan terkait hal-hal sebagai berikut :

a. *Field type* : tipe *field* ini dapat terkait apakah *field* tersebut akan berisi data berupa *key field* (*primary key, secondary key*), atau *descriptor*. *Primary key* akan berisi ID atau kode pokok yang akan digunakan dalam mengidentifikasi *record*, sehingga data di dalam *field* tersebut tidak diijinkan untuk memiliki lebih dari satu data yang sama. *Secondary* adalah subset dari *key* utama. Misalnya saja kode mata kuliah dalam satu semester tidak boleh terdapat lebih dari satu pada ID atau NIM yang sama. *Descriptor* adalah *field* berisi data yang akan merupakan satu kesatuan dengan yang lainnya sebagai penjelasan akan adanya *record* atau ID tertentu.

b. Data *type*: tipe data merupakan jenis data yang dapat dimasukkan dalam *field*. Hal ini dapat dibagi secara umum sebagai karakter/ *text*, numerik, tanggal dan sebagainya.

c. *Field Size*. Penting untuk memahami ukuran *field* yang akan digunakan dalam menampung data. Dalam pengembangan sistem harus dapat memperkirakan berapa lebar ukuran *field* yang efektif. Apabila terlalu lebar akan terjadi banyak spasi kosong dan berpengaruh pada ukuran *file* yang

disimpan. Sementara apabila terlalu sempit akan terdapat data yang tidak tersimpan.

5. *Records*

Records merupakan baris data. Karena satu baris data biasanya mengindikasikan satu kesatuan data tertentu, maka satu *record* ada yang menyebut satu data. Misalnya keterangan mengenai biodata Andi disimpan dalam satu *record* beridentitas ID 11001, maka untuk menyebutkan satu kesatuan data seputar Andi dalam baris tertentu ada yang menyebutkan sebagai *record* data Andi. Dalam konsep *Database* masing-masing *record* memiliki nomor identitas tersendiri baik itu identitas yang diberikan komputer ataupun yang diinputkan secara manual. Sehingga dalam konteks tertentu dapat digunakan konsep nomor *record*, ID otomatis, ID *primary key*. (Mujilan : 2012:31)

Normalisasi merupakan sebuah teknik dalam logical desain sebuah basis data / *database*, teknik pengelompokkan atribut dari suatu relasi sehingga membentuk struktur relasi yang baik tanpa redudansi. Tujuan normalisasi adalah mengorganisasikan data kedalam tabel-tabel untuk memenuhi kebutuhan pemakai, menghilangkan kerangkapan data, mengurangi kompleksitas, mempermudah modifikasi data. (Mukhlisulfatih Latief : 2016)

1. Proses Normalisasi

   a. Data diuraikan dalam bentuk tabel, selanjutnya dianalisis berdasarkan persyaratan tertentu kebeberapa tingkat.

b. Apabila tabel yang diuji belum memenuhi persyaratan tertentu maka tabel tersebut pelu dipecah menjadi beberapa tabel yang lebih sederhana sampai memenuhi bentuk yang optimal.

2. Tahapan Normalisasi :

1) Bentuk tidak normal : Menghilangkan perulangan grup.

Tabel II.1 Contoh bentuk tidak normal (Unnormal)

No-Mhs	Nama Mhs	Jurusan	Kode-MK	Nama-MK	Kode Dosen	Nama Dosen	Nilai
2683	Welli	MI	M1350	Manajemen DB	B104	Ati	A
			M1465	Analisis Perc. Sistem	B317	Dita	B
5432	Bakti	AK	M1350	Manajemen DV	B104	Ati	C
			Akn201	Akuntansi	D310	Lia	B
			MKT300	Dasar Pemasaran	B212	Lola	A

Sumber : Mukhlisulfatih Latief : 2016

2) Bentuk Normal pertama (1NF) : Menghilangkan ketergantungan sebagian.

Yaitu : suatu relasi dikatakan sudah memenuhi bentuk normal kesatu bila setiap data bersifat atomik yaitu setiap irisan baris dan kolom hanya mempunyai satu nilai data.

Tabel II.2. Contoh Bentuk Normal Pertama (1NF)

No-Mhs	Nama Mhs	Jurusan	Kode-MK	Nama-MK	Kode Dosen	Nama Dosen	Nilai
2683	Welli	MI	M1350	Manajemen DB	B104	Ati	A
2683	Welli	MI	M1465	Analisis Perc. Sistem	B317	Dita	B
5432	Bakti	AK	M1350	Manajemen DV	B104	Ati	C
5432	Bakti	AK	Akn201	Akuntansi	D310	Lia	B

5432	Bakti	AK	MKT300	Dasar Pemasaran	B212	Lola	A

Sumber : Mukhlisulfatih Latief : 2016

3) Bentuk Normal kedua (2NF) : Menghilangkan ketergantungan transitif.

Yaitu : suatu relasi dikatakan sudah memenuhi bentuk normal kedua bila relasi tersebut sudah memenuhi bentuk normal kesatu dan atribut yang bukan *key* sudah tergantung penuh terhadap *key*-nya.

Tabel II.3. Contoh Bentuk Normal Kedua (2NF)

Kode-MK	Nama-MK	Kode Dosen	Nama Dosen
M1350	Manajemen DB	B104	Ati
M1465	Analisis Perc. Sistem	B317	Dita
M1350	Manajemen DV	B104	Ati
Akn201	Akuntansi	D310	Lia
MKT300	Dasar Pemasaran	B212	Lola

Sumber : Mukhlisulfatih Latief : 2016

4) Bentuk Normal ketiga (3NF) : Menghilangkan anomali-anomali hasil dari ketergantungan fungsional. Yaitu : suatu relasi dikatakan sudah memenuhi bentuk normal ketiga bila relasi tersebut sudah memenuhi bentuk normal kedua dan atribut yang bukan *key* tidak tergantung transitif terhadap *key*-nya.

Tabel II.4. Contoh Tabel Mahasiswa Dan Tabel Kuliah (3NF)

No_Mhs	Nama Mhs	Jurusan
2683	Welli	MI
5432	Bakti	AK

Sumber : Mukhlisulfatih Latief : 2016

II.2.8. *SQL Server* 2008

SQL (*Structured Query Language*) adalah bahasa non procedural untuk mengakses data pada *Database* relasional. SQL adalah bahasa *database* yang dipergunakan dalam menyelesaikan permasalahan dalam *database* serta mempunyai kelebihan dalam mengolah data. Standar SQL mula-mula didefenisikan oleh ISO *(International Standards Organization)* dan ANSI *(the American National Standards Institute)* yang dikenal dengan sebutan SQL86. (Eka Iswandy : 2015 : 73)

Dengan menggunakan SQL, kita dapat melakukan hal-hal berikut:

1. Memodifikasi struktur *database* .
2. Mengubah, mengisi, menghapus isi *Database*.
3. Mentransfer data antara *Database* yang berbeda. SQL ada yang dikembangkan untuk PC dan ada juga yang dikembangkan untuk dapat mengakomodasi *Database* yang sangat besar.

Beberapa contohnya antara lain:

1. *Microsoft Accsess*

Digunakan untuk PC, sangat mudah dipakai dimana perintah SQL dapat langsung dimasukkan atau melalui fasilitas yang telah digunakan.

2. *Microsoft Query*

SQL yang dipaket dengan produk lain dari *Microsoft Windows*, yaitu *Microsoft Visual Studio* seperti *Visual Basic* dan *Visual* C++. Untuk terhubung dengan *Database* lain menggunakan ODBC.

3. *Oracle*

Digunakan untuk perusahaan yang mengggunakan *database* besar. (Eka Iswandy : 2015 : 73)

II.2.9. Visual Basic

Microsoft Visual Studio merupakan sebuah perangkat lunak lengkap yang dapat digunakan untuk melakukan pengembangan aplikasi, baik itu aplikasi bisnis, aplikasi personal, ataupun komponen aplikasi lainnya dalam bentuk aplikasi *console*, aplikasi *Windows,* ataupun aplikasi Web. Kompiler yang dimasukkan ke dalam paket *Visual Studio* antara lain *Visual C++, Visual C#, Visual Basic, Visual Basic .NET, Visual InterDev, Visual J++, Visual J#, Visual FoxPro*, dan *Visual SourceSafe*. *Microsoft Visual Studio* dapat digunakan untuk mengembangkan aplikasi dalam *native code* (dalam bentuk bahasa mesin yang berjalan di atas Windows) ataupun *managed code* (dalam bentuk *Microsoft Intermediate Language* di atas *.NET Framework*).

Selain itu, *Visual Studio* juga dapat digunakan untuk mengembangkan aplikasi *Silverlight*, aplikasi *Windows Mobile* (yang berjalan di atas .NET *Compact Framework*).*Visual Studio* kini telah menginjak versi *Visual Studio* 12.0 atau dikenal dengan sebutan *Microsoft Visual Studio* 2013 yang diluncurkan pada 17 Oktober 2013 yang ditujukan untuk *platform Microsoft .NET Framework* 4.5.1. Versi sebelumnya Visual Studio 2012 ditujukan untuk platform 4.5, Visual Studio 2010 ditujukan untuk *.NET Framework* 4.0, *Visual Studio* 2008 ditujukan untuk *platform .NET Framework* 3.5, *Visual Studio* 2005 ditujukan untuk *platform .NET Framework* 2.0 dan 3.0. *Visual Studio* 2003 ditujukan untuk *.NET*

Framework 1.1, dan *Visual Studio* 2002 ditujukan untuk *.NET Framework* 1.0. Versi-versi tersebut di atas kini dikenal dengan sebutan *Visual Studio.NET*, karena memang membutuhkan *Microsoft .NET Framework*. Sementara itu, sebelum muncul *Visual Studio .NET*, terdapat *Microsoft Visual Studio* 6.0. *(Ahmad Rasi* Ruli*; 2017 : 10)*

II.2.10. UML (*Unified Modeling Language*)

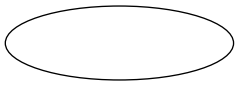
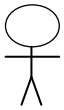
Menurut Gata (2013 : 4) Hasil pemodelan pada OOAD terdokumentasikan dalam bentuk *Unified Modeling Language* (UML). UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak. UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem. UML saat ini sangat banyak dipergunakan dalam dunia industri yang merupakan standar bahasa pemodelan umum dalam industri perangkat lunak dan pengembangan sistem. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use case* Diagram

   Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi

antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram, yaitu :

Tabel II.5. Simbol *Use Case*

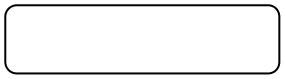
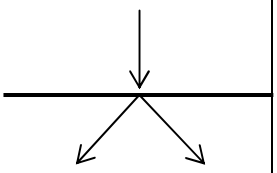
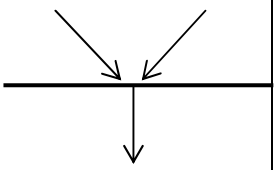
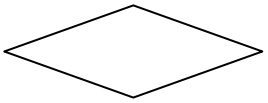
Gambar	Keterangan
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*.
	Aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi dari target sistem. Untuk mengidentifikasikan aktor, harus ditentukan pembagian tenaga kerja dan tugas-tugas yang berkaitan dengan peran pada konteks target sistem. Orang atau sistem bisa muncul dalam beberapa peran. Perlu dicatat bahwa aktor berinteraksi dengan *use case*, tetapi tidak memiliki control terhadap *use case*.
	Asosiasi antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengidikasikan aliran data.
	Asosiasi antara aktor dan *use case* yang menggunakan panah terbuka untuk mengidinkasikan bila aktor berinteraksi secara pasif dengan sistem.
	Include, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.
	Extend, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

(Sumber : Gata, 2013 : 4)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram,* yaitu :

Tabel II.6. Simbol *Activity Diagram*

Gambar	Keterangan
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activites*, menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan pararel menjadi satu.
	*Join* (penggabungan) atau rake, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points,* menggambarkan pilihan untuk pengambilan keputusan, *true, false.*
	*Swimlane*, pembagian *activity* diagram untuk menunjukkan siapa melakukan apa.

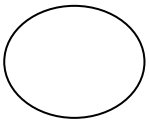
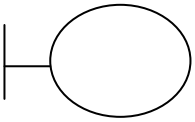
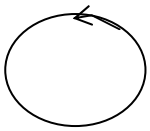
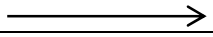
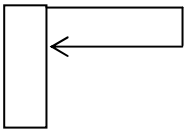
(Sumber : Gata, 2013 : 6)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan

diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram*, yaitu :

Tabel II.7. Simbol *Sequence Diagram*

Gambar	Keterangan
	*Entity Class,* merupakan bagian dari sistem yang berisi kumpulan kelas berupa entitas-entitas yang membentuk gambaran awal sistem dan menjadi landasan untuk menyusun basis data.
	*Boundary Class,* berisi kumpulan kelas yang menjadi *interface* atau interaksi antara satu atau lebih aktor dengan sistem, seperti tampilan *formentry* dan *form* cetak.
	*Control class,* suatu objek yang berisi logika aplikasi yang tidak memiliki tanggung jawab kepada entitas, contohnya adalah kalkulasi dan aturan bisnis yang melibatkan berbagai objek.
	*Message*, simbol mengirim pesan antar *class*.
	*Recursive,* menggambarkan pengiriman pesan yang dikirim untuk dirinya sendiri.
	*Activation, activation* mewakili sebuah eksekusi operasi dari objek, panjang kotak ini berbanding lurus dengan durasi aktivitas sebuah operasi.
	Lifeline, garis titik-titik yang terhubung dengan objek, sepanjang *lifeline* terdapat *activation*.

(Sumber : Gata, 2013 : 7)

4. *Class Diagram* (Diagram Kelas)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggng jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan

constraint yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/ Method*), *Visibility,* tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti.

Tabel II.8. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber : Gata, 2013 : 9)