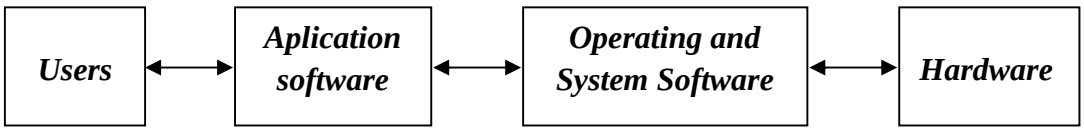
BAB II

TINJAUAN PUSTAKA

II.1. Perangkat Lunak

Perangkat lunak komputer atau sering disebut *software* komputer adalah bagian dari sistem komputer yang sifatnya non-fisik yang berupa suatu modul, perintah, prosedur, pengendali, pendukung, dan aktifitas-aktifitas pengolahan perintah pada sistem komputer. Jadi intinya komputer bisa digunakan atau bisa hidup karena menggunakan *software*.

Software komputer secara garis besar dibagi menjadi 2 yaitu *software* sistem operasi (*operating System*) dan *software* sistem aplikasi (*application software*). (Aji Supriyanto; 2005 : 32)



Gambar II.1 Model Hubungan Antar Sistem Komputer

(Sumber : Aji Supriyanto; 2005 : 32)

II.1.1. Sistem Operasi

Sistem operasi merupakan perangkat lunak yang berfungsi melakukan operasi yang mengurusi tentang segala aktifitas komputer seperti mendukung operasi sistem aplikasi dan mengendalikan semua perangkat komputer agar dapat berjalan selaras dengan fungsinya.

Tugas dari sistem operasi diantaranya :

1. Melakukan fungsi manajemen sistem berkas.
2. Mengendalikan berbagai sumber pada sistem seperti disk dan printer.
3. Mengatur sejumlah pemakai yang menggunakan sistem bersamaan.
4. Membentuk penjadwalan proses-proses di dalam sistem.

Tujuan utama sistem operasi yaitu untuk memaksimalkan produktivitas sistem komputer dengan pengoperasian yang efisien. Operasi sistem yang dilakukan berupa pengaturan sistem program (*system management programs*) dan pengaturan pengembangan sistem (*system development programs*).

Gambar II.2 Lingkup Konfigurasi Sistem Komputer

(Sumber : Aji Supriyanto; 2005 : 33)

Fungsi system operasi antara lain :

1. Menyediakan antarmuka pengguna (*user interface*), terdapat 3 jenis antarmuka pengguna, yaitu :
   a. Melakukan perintah (*command-based user interface)* dalam bentuk teks
   b. Mengarahkan Menu (menu *driven*)

c. Antarmuka unit grafik (*graphical user interface-GUI*)

d. Kombinasi ikon dan menu untuk menerima dan melaksanakan perintah

2. Menyediakan informasi yang berkaitan dengan *hardware*, yaitu berupa perangkat yang aktif dan pasif, informasi penambahan atau pencopotan *hardware.*

3. Menangani memori, tujuannya yaitu untuk memberikan informasi awal bagaimana memori bisa dibaca serta memaksimalkan memori yang ada dan penyimpanan.

4. Melakukan tugas pengolahan dalam sebuah proses sebagai berikut :

   a. *Multitasking,* bisa melakukan tugas serentak atau sekaligus pada aplikasi yang sama maupun berbeda.

   b. *Multiprocessing,* penggunaan atau pemprosesan sebuah program secara serentak oleh beberapa unit CPU.

   c. *Time-sharing,* menggunakan sistem komputer yang sama pada banyak pengguna.

   d. *Multithreading,* memproses aktivitas pada bentuk yang sama dengan *multitasking* tetapi pada aplikasi tunggal.

   e. *Scalability and network,* upaya komputer dalam mengendalikan dan meningkatkan kewaspadaan dan keamanan jumlah pengguna dan memperluas pelayanan.

5. Pengolahan *file* data, memastikan *file-file* dalam penyimpanan sekunder tersedia bila diperlukan, dan mengamankan dari pengguna yang tidak dibenarkan mengaksesnya. (Aji Supriyanto; 2005 : 33-34)

II.1.2. *Software* Aplikasi

Software aplikasi atau program aplikasi adalah program yang memiliki aktifitas pemprosesan perintah yang diperlukan untuk melaksanakan permintaan pengguna.

Program aplikasi adalah sebuah program komputer yang ditujukan untuk pemakai agar bisa berinteraksi dengan suatu *database* (Abdul Kadir; 2010 : 4).

Jenis – jenis dan fungsi-fungsi program aplikasi adalah :

1. Paket Aplikasi, merupakan *software* yang sifatnya instan dan telah dirancang oleh pembuat *software* tersebut untuk siap digunakan dalam paket yang telah ditentukan. Contohnya :
   a. Word Processing (misalnya: *Microsoft Word* dan *Star Offoce*)
   b. Value Processing(misalnya: *Microsoft Excel* dan SPSS)
   c. Desain Processing(misalnya: *Visio* dan *Corel Draw*)
   d. Image Processing(misalnya: *adobe photoshop*), dan lain sebagainya.
2. Program Aplikasi, merupakan *software* yang pemanfaatannya dengan cara mendesain program untuk keperluan pengembangan sistem informasi. Contohnya : *Pascal, Delphi, Visual Basic, Oracle*, C++ dan lain sebagainya. Bahasa pemrograman ini biasanya dibagi atas 3 tingkatan yaitu :
   a. Bahasa pemrograman tingkat rendah (*Low Level Language*), bahasa pemrograman generasi pertama, bahasa pemrograman jenis ini sangat sulit dimengerti karena instruksinya menggunakan bahasa mesin.
   b. Bahasa pemrograman tingkat menengah (*Midle Level Language*), bahasa pemrograman tingkat menengah dimana penggunaan instruksi sudah

mendekati bahasa sehari-hari, walaupun begitu masih sulit di mengerti karena banyak menggunakan singkatan-singkatan seperti STO artinya simpan(singkatan dari *STORE*) dan MOV artinya pindah (singkatan dari *MOVE*).

c. Bahasa pemrograman tingkat tinggi (*High Level Language*), merupakan bahasa tingkat tinggi yang mempunyai cirri mudah dimengerti, karena menggunakan bahasa sehari-hari, contoh bahasa yang disebut diatas merupakan bahasa yang digunakan pada level ini.

(Aji Supriyanto; 2005 : 35-36)

II.2. Kamus

Kamus adalah sejenis buku rujukan yang menerangkan makna kata-kata. Ia berfungsi untuk membantu seseorang mengenal perkataan baru. Selain menerangkan maksud kata, kamus juga mungkin mempunyai pedoman sebutan, asal-usul (etimologi) sesuatu perkataan dan juga contoh penggunaan bagi sesuatu perkataan. Untuk memperjelas kadangn kala terdapat juga ilustrasi di dalam kamus. Biasanya hal ini terdapat dalam kamus bahasa Perancis.

Kata kamus diserap dari bahasa Arab *qamus* (قاموس), dengan bentuk jamaknya *qawamis*. Kata Arab itu sendiri berasal dari kata Yunani Ωκεανός (*okeanos*) yang berarti'samudra'. Sejarah kata itu jelas memperlihatkan makna dasar yang terkandung dalam kata kamus, yaitu wadah pengetahuan, khususnya pengetahuan bahasa, yang tidak terhingga dalam dan luasnya.

Dewasa ini kamus merupakan khazanah yang memuat perbendaharaan kata suatu

bahasa, yang secara ideal tidak terbatas jumlahnya.

II.2.1. Jenis-jenis Kamus

1. Berdasarkan penggunaan bahasa

   Kamus bisa ditulis dalam satu atau lebih dari satu bahasa. Dengan itu kamus bisa dibagi menjadi beberapa jenis yaitu:

   a. Kamus Ekabahasa
   b. Kamus Dwibahasa
   c. Kamus Aneka bahasa

2. Berdasarkan isi

   a. Kamus mini
   b. Kamus kecil
   c. Kamus besar

3. Kamus Istilah

   a. Kamus Etimologi
   b. Kamus Tesaurus (perkataan searti)
   c. Kamus Peribahasa/Simpulan Bahasa
   d. Kamus Kata Nama Khas
   e. Kamus Terjemahan
   f. Kamus Kolokasi

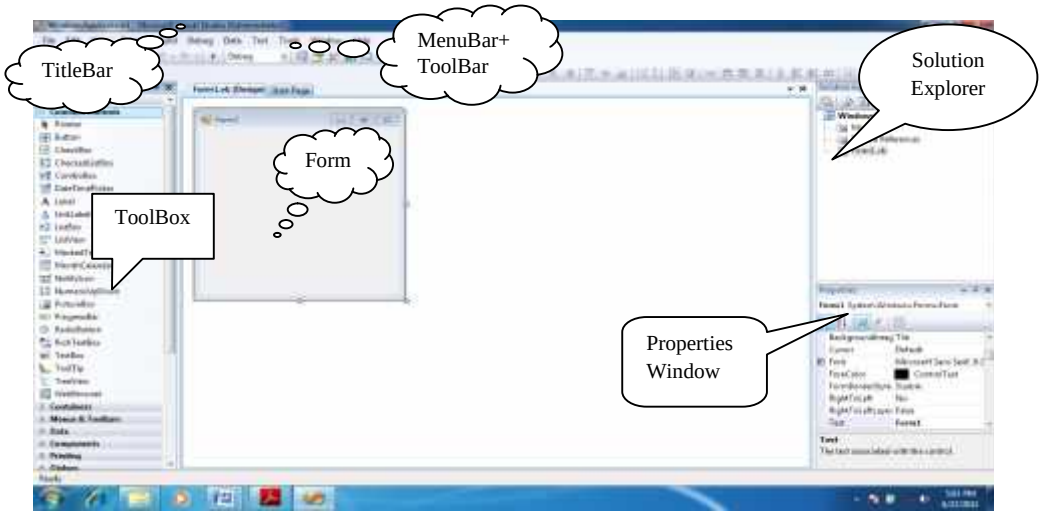
(Racang Bangun Perangkat Lunak Aplikasi Kamus Istilah Dunia Komputer dan Internet Bergambar ; Sugiono, Muhammad Muhibbudin Roziq ; KONVERGENSI Volume 4, Nomor 2, Juli 2008)

II.3. Mengenal *Microsoft Visual Basic .Net* 2008

Pemrograman *Microsoft Visual Studio .Net* 2008 adalah sebuah *platform* untuk membangun, menjalankan, dan meningkatkan generasi lanjut dari aplikasi terdistribusi. *.Net Framework* merupakan *platform* terbaru untuk pemrograman aplikasi *window* dari *Microsoft* dalam upaya meningkatkan produktivitas pembuatan sebuah program aplikasi dan memungkinkan terbukanya peluang untuk menjalankan program pada *multi sistem operasi* serta dapat memperluas pengembangan aplikasi *Client-Server* (Ketut Darmayuda; 2010 : 3)

II.3.1. Antarmuka *Microsoft Visual Basic .Net IDE* 2008

Antarmuka atau lingkungan dari *Visual Basic .Net* IDE 2008 tidak jauh berbeda dengan *Visual Basic* 6.0 IDE, kelebihannay memiliki IDE (*Interface Development Environment*) yang lebih lengkap dan terorganisasi, sehingga mudah bagi pengembang untuk mencari objek-objek atau komponen yang terdapat pada *toolbox* yang kita inginkan, untuk ditempatkan pada objek *form*, dengan meng-klik sebuah objek dan kemudian diletakkan di atas *form.* Berikut adalah tampilan lingkungan dari *Visual Basic .Net* 2008 (Ketut Darmayuda; 2010 : 13).



Gambar II.3 *Interface Microsoft Visual Basic .NET* IDE 2008

(Sumber: Ketut Darmayuda; 2010 : 13)

Lingkungan kerja pada *Microsoft Visual Basic .NET* 2008 umumnya memiliki lingkungan sebagai berikut :

1. *Title Bar*, berfungsi untuk menampilkan nama *project* yang aktif atau sedang dikembangkan.

Gambar II.4 *Title Bar Visual Basic .NET* 2008

(Sumber: Ketut Darmayuda; 2010 : 14)

2. *Menu Bar*, berfungsi untuk pengolahan fasilitas yang dimiliki oleh *Visual Basic .NET* 2008, sedangkan *Tool Bar*, berfungsi untuk melakukan perintah khusus secara cepat.

Gambar II.5 *Menu Bar* dan *Tool Bar Visual Basic .NET* 2008

(Sumber: Ketut Darmayuda; 2010 : 14)

3. *Form*, adalah objek utama berfungsi untuk meletakkan objek-objek yang terdapat pada *Toolbox* yang digunakan dalam melakukan perancangan sebuah tampilan program aplikasi, contohnya sebagai berikut :

Gambar. II.6 Contoh penggunaan objek pada *form*

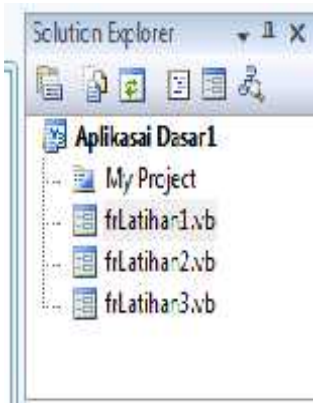
(Sumber: Ketut Darmayuda; 2010 : 57)

4. *ToolBox*, berfungsi untuk menyediakan objek-objek atau komponen yang digunakan dalam merancang sebuah *form* pada program aplikasi.

Gambar II.7 *ToolBox Visual Basic .NET* 2008

(Sumber: Ketut Darmayuda; 2010 : 15)

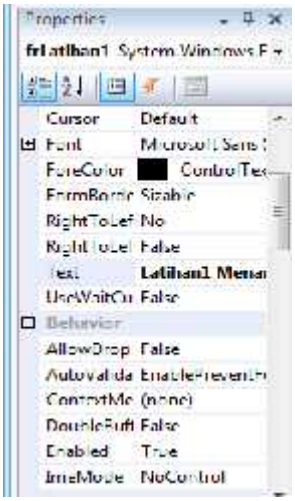
5. *Solution Explorer*, berfungsi untuk menampilkan *project* beserta *file-file* pendukung yang terdapat pada sebuah program aplikasi.



Gambar II.8 *Solution Explorer* VB *.NET* 2008

(Sumber: Ketut Darmayuda; 2010 : 16)

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



Gambar II.9 *Properties Windows* VB *.NET* 2008

(Sumber: Ketut Darmayuda; 2010 : 16)

II.4. *Database* dan *MySQL*

Database sering didefinisikan sebagai kumpulan data yang terkait. Secara teknis, yang berada dalam sebuah *database* adalah sekumpulan tabel atau objek

lain (*indeks, view*, dll). Tujuan utama pembuatan *database* adalah untuk memudahkan dalam mengakses data. Data dapat ditambahkan, diubah, dihapus, atau dibaca dengan relatif mudah dan cepat.

Saat ini tersedia banyak perangkat lunak yang ditujukan untuk mengelola *database*. Perangkat lunak seperti itu biasa dinamakan DBMS (*database management system*). *Access, MS SQL Server*, dan *MySQL* merupakan contoh produk pengelola *database.*

MySQL merupakan *software* yang tergolong *database server* dan bersifat *Open Source*. *Open Source* menyatakan bahwa *software* ini dilengkapi dengan *source code* (kode yang dipakai untuk membuat *MySQL*), selain tentu saja bentuk *executeable*-nya atau kode yang dapat dijalankan secara langsung dalam system operasi, dan bisa diperoleh dengan cara mengunduh di internet secara gratis. Pengaksesan data dalam *database* dapat dilakukan dengan mudah melalui SQL(*Structured Query Language*). (Abdul Kadir ;2009 : 14 - 15)

II.5. Konsep Algoritma

Algortima merupakan fondasai yang harus dikuasai oleh setiap mahasiswa yang ingin menyelesaikan sesuatu masalah secara terstruktur, efektif, dan efisien, teristimewa lagi bagi mahasiswa yang ingin menyusun program komputer untuk menyelesaikan suatu persoalan. (Drs Suarga; 2012 : 1).

II.5.1 Algoritma *Boyer-Moore*

Algoritma *Boyer-Moore* adalah salah satu algoritma pencarian *string*, dipublikasikan oleh Robert S. Boyer, dan J. Strother Moore pada tahun 1977.

Algoritma ini dianggap sebagai algoritma yang paling efisien pada aplikasi umum. Tidak seperti algoritma pencarian *string* yang ditemukan sebelumnya, algoritma *Boyer-Moore* mulai mencocokkan karakter dari sebelah kanan pattern. Ide dibalik algoritma ini adalah bahwa dengan memulai pencocokkan karakter dari kanan, dan bukan dari kiri, maka akan lebih banyak informasi yang didapat.

Algortima *Boyer moore* melakukan perbandingan dimulai dari kanan ke kiri, tetapi pergeseran *window* tetap dari kiri ke kanan. Jika terjadi kecocokan maka dilakukan perbandingan karakter teks dan karakter pola yang sebelumnya, yaitu dengan sama – sama mengurangi indeks teks dan pola masing – masing sebanyak satu. Jika terjadi ketidakcocokan maka ada beberapa kondisi untuk melakukan pergeseran. Kemungkinan ini berdasarkan karakter pada teks yang menyebabkan terjadi ketidakcocokan.

Contoh :

Langkah - langkah untuk mencari kata artist pada the-greatest- artist

a. Langkah pertama menyamakan posisi huruf terakhir dari *input* dengan kata yang dicari

   N : | a | r | t | i | s | t |

   M : | t | h | e | - | g | r | e | a | t | e | s | t | - | a | r | t | i | s | t |

b. Kemudian diamati pada posisi paling akhir input, karena posisi ke 6 N tidak sama dengan posisi ke 6 M, maka posisi artist digeser ke kanan sebanyak 4 posisi, sampai huruf r dari kata artist sesuai dengan huruf r pada kata greatest.

   N : | a | r | t | i | s | t |

   M : | t | h | e | - | g | r | e | a | t | e | s | t | - | a | r | t | i | s | t |

c. Karena pada posisi ke 10 M (huruf e) tidak sama dengan huruf akhir kata artist, dan huruf tersebut tidak sama dengan semua huruf yang ada pada kata artist maka kata artist digeser ke kanan sebanyak 5 posisi.

N : a r t i s t

M : t h e - g r e a t e s t - a r t i s t

d. Pada posisi baru ini ini posisi terakhir dari kata artist sama dengan posisi urutannya, namun posisi kedua dari akhir tidak sama , maka kata artist digeser 3 posisi lagi. Akhirnya semua huruf yang ada pada kata artist sesuai dengan huruf pada teks.

N : a r t i s t

M : t h e - g r e a t e s t - a r t i s t

II.5.2 Algoritma *Knuth-Morris-Pratt*

Algoritma *Knuth-Morris-Pratt* (KMP) dikembangkan oleh E.D.Knuth, bersama dengan J.H Morris dan V.R.Pratt. untuk pencariang *string* dengan menggunakan algoritma *Bruto Force*, setiap kali ditemukan ketidakcocokan *pattern* dengan teks, maka *pattern* akan digeser satu karakter ke kanan.

Algoritma *Knuth-Morris-Pratt* juga dilakukan perbandingan karakter di teks dan karakter di pola kiri ke kanan, tetapi bedanya dengan algoritma *bruto force* adalah pada pergeseran. Ide dari algoritma ini adalah bagaimana dapat dimanfaatkan karakter-karakter pola yang sudah diketahui ada di dalam teks sampai terjadi ketidakcocokan untuk melakukan pergeseran.

Contoh :

Langkah pertama :

N : a r t i s t

M : : t h e - g r e a t e s t - a r t i s t

Langkah kedua :

N : a r t i s t

M : : t h e - g r e a t e s t - a r t i s t

Langkah ketiga :

N : a r t i s t

M : : t h e - g r e a t e s t - a r t i s t

Langkah keempat :

N : a r t i s t

M : : t h e - g r e a t e s t - a r t i s t

Langkah kelima :

N : a r t i s t

M : : t h e - g r e a t e s t - a r t i s t

II.6. ERD (*Entity Relationship Diagram*)

ERM digambarkan dalam bentuk diagram yang disebut ER (ER_Diagram/ERD). Untuk menggambarkan ERD digunakan simbol-simbol grafis tertentu. Bagi perancang/analisis sistem, ERD berguna untuk memodelkan sistem yang nantinya basis datanya akan dikembangkan. Model ini juga membantu perancang/analis sistem pada saat melakukan analisis dan perancangan basis data karena model ini dapat menunjukkan macam data yang dibutuhkan darn kerelasian antar data di dalamnya. Bagi pengguna, model sangat membantu dalam hal pemahaman model sistem dan rancangan basis data yang akan dikembangkan oleh perancangan/analis sistem (sutanta, 2004).

Sebuah diagram ER/ERD tersusun atas tiga komponen yaitu : entitas, atribut, dan kerelasian antar entitas. Secara garis besar, entitas merupakan obyek dasar yang terlibat dalam sistem. Atribut berperan sebagai penjelas entitas, sedangkan kerelasian menunjukkan hubungan yang terjadi di antara dua entitas (Silberschatz, dkk, 2001).

II.6.1 Entitas (*Entity*)

Entitas menunjukkan obyek-obyek dasar yang terkait di dalam sistem.obyek dasar dapat berupa orang, benda, atau hal yang keterangannya perlu disimpan di dalam basis data. Untuk menggambarkan sebuah entitas digunakan aturan sebagai berikut (Sutanta, 2004) :

1. Entitas dinyatakan dengan simbol persegi panjang
2. Nama entitas dituliskan di dalam simbol persegi panjang.
3. Nama entitas berupa kata benda, tunggal

4. Nama entitas sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

II.6.2 Atribut (*Attribute*)

Atribut sering disebut sebagai property (*property*), merupakan keterangan-keterangan yang terkait pada sebuah entitas yang perlu disimpan dalam basis data. Untuk menggambarkan atribut digunakan aturan sebagai berikut :

1. Atribut dinyatakan dengan simbol elips.
2. Nama atribut dituliskan di dalam simbol elips
3. Nama atribut berupa kata benda, tunggal
4. Nama atribut sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.
5. Atribut dihubungkan dengan entitas yang bersesuaian dengan menggunakan sebuah garis (seyogianya menggunakan garis lurus, namun dalam kondisi yang tidak memungkinkan dapat juga tidak menggunakan garis lurus)

II.6.3 Kerelasian Antar Entitas (*Relationship*)

Kerelasian antar entitas mendefinisikan hubungan antara dua buah entitas. Kerelasian adalah kejadian atau transaksi yang terjadi di antara dua buah entitas yang keterangannya perlu disimpan dalam basis data (Martin, 1975). Aturan dalam penggambaran kerelasian antar entitas adalah :

1. Kerelasian dinyatakan dengan simbol belah ketupat
2. Nama kerelasian ditulis dalam simbol belah ketupat
3. Kerelasian menghubungkan dua entitas
4. Nama kerelasian berupa kata kerja aktif(diawali dengan awalan me-), tunggal .

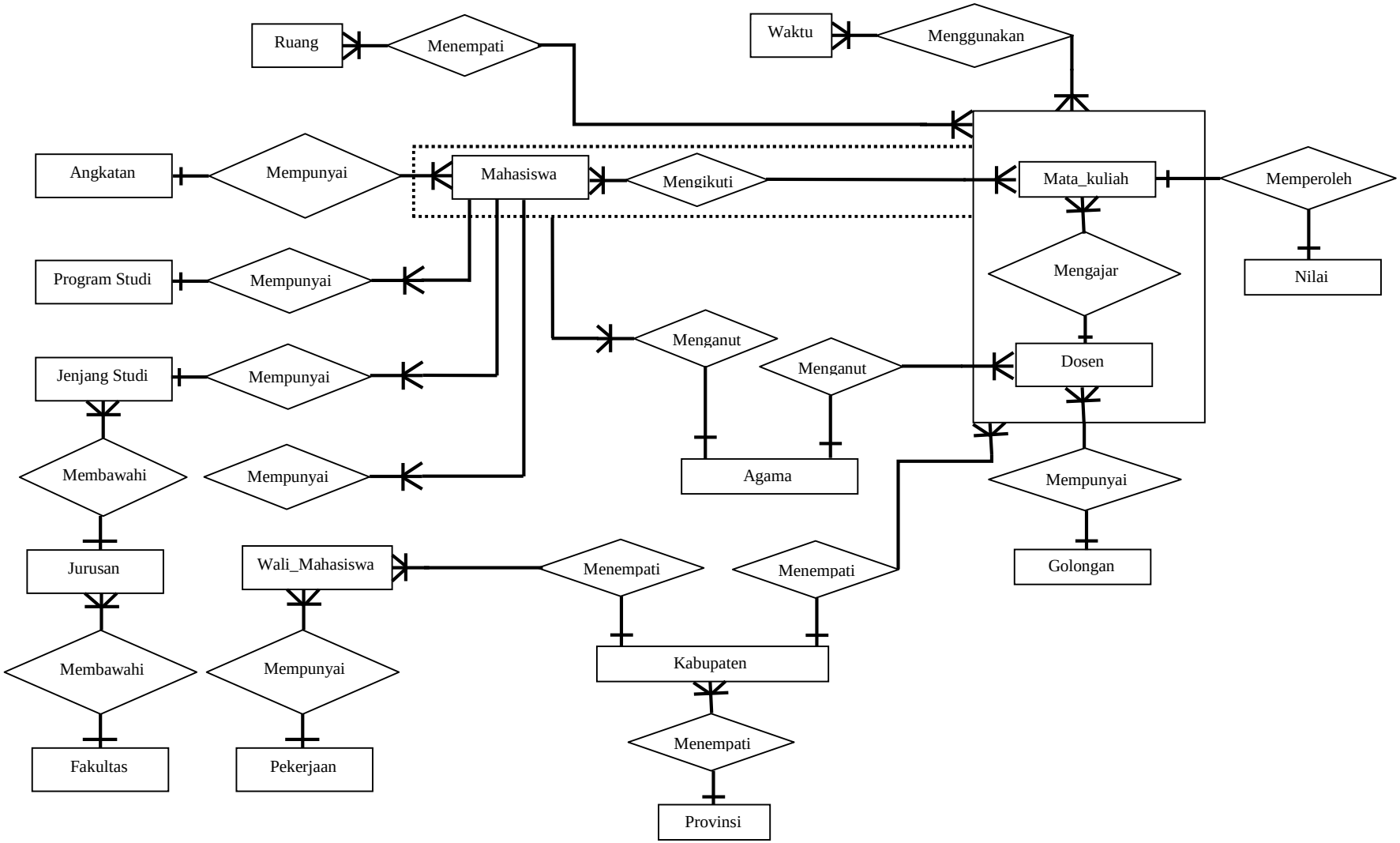
5. Nama kerelasian sedapat mungkin menggunakan nama yang mudah dipahami dan dapat menyatakan maknanya dengan jelas.

II.6.4 Menggambar ERD

Untuk menggambarkan ERD secara lengkap dapat dilakukan dengan mengikuti serangkaian langkah berikut (Sutanta, 2004):

1. Identifikasikan setiap entitas yang terlibat
2. Identifikasikan setiap atribut pada setiap entitas
3. Identifikasikan setiap kerelasian berikut jenisnya yang terjadi di antara entitas
4. Gambarkan simbol-simbol entitas, atribut, dan kerelasian antar entitas sedemikian sehingga simbol kerelasian dapat digambarkan dengan jelas dan tidak saling bertabrakan.
5. Cek ERD yang terbentuk dalam hal :
   a. Kelengkapan entitas
   b. Kelengkapan atribut
   c. Kelengkapan kerelasian antar entitas
   d. Jenis kerelasian antar entitas

   (Edhy Sutanta; 2011 : 91-113)



Gambar II.10 ***Entity Relationship Diagram (ERD)***

(Sumber : Edhy Sutanta; 2011 : 119)

II.7. UML (*Unified Modelling Language*)

UML (*Unified Modelling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipahami. (Adi Nugroho; 2010 : 6 - 7)

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual. Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem *software* yang terkait dengan objek. (Memahami Pengunaan UML (*Unified Modelling Language*) ; Haviluddin ; Jurnal Informatika Mulawarman ; Vol 6 No.1 Februari 2011)

II.7.1 Konsep dan Diagram UML

Konsep dan diagram dalam UML dibagi menjadi beberapa *view*. Suatu *view* sendiri pada dasarnya merupakan sejumlah konstruksi pemodelan UML yang merepresentasikan suatu aspek tertentu dari sistem / perangkat lunak yang sedang dikembangkan. Hubungan antara *view-view* dan diagram-diagram yang menampilkannya, serta konsep-konsep utama yang relevan untuk masing-masing *view* ditunjukkan pada tabel II.1 dibawah ini :

Tabel II.1 View dan Diagram Dalam UML

Major Area	*View*	*Diagrams*	*Main Concepts*
structural	*Static view*	*Class diagram*	*Class, association, generalization, dependency, realization, interface*
	Use case view	*Use case diagram*	*Use case, actor, association, extend, include, use case generalization*
	Implementation view	*Component diagram*	*Component, interface,dependency, realization*
	Deployment view	*Deployment diagram*	*Node, component, dependency, location*
Dynamic	*State machine view*	*Statechart diagram*	*State, event, transition, action*
	Activity view	*Activity diagram*	*State, activity, completion transition, fork, join*
	Interaction view	*Sequence diagram*	*Interaction, object, message, activation*
		Collaboration diagram	*Collaboration, interaction, collaboration role, message*
Model management	*Model management view*	*Class diagram*	*Package, subsystem, model*
extensibility	*all*	*all*	*Constraint, stereotype, tagged values*

(Sumber : Adi Nugroho; 2010 : 9-10)

II.7.2 Pengertian *Class diagram*

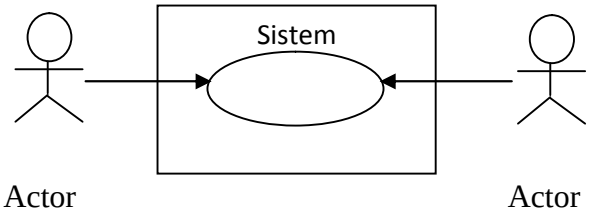
Class dalam notasi *UML* digambarkan dengan kotak. nama *class* menggambarkan huruf besar diawal kalimatnya dan diletakkan di atas kotak (Munawar; 2005: 35).

Gambar II.11 Notasi *class* di UML

(Sumber : Munawar; 2005: 35)

II.7.3. Pengertian *Use Case diagram*

Use Case adalah deskripsi fungsi dari sebuah sistem dari *perspektif* pengguna. *Use case* bekerja dengan cara mendeskripsikan tipikal interaksi antar *user* (pengguna) sebuah sistem dengan sistemnya sendiri melalui sebuah cerita bagaimana sebuah sistem yang dipakai. (Munawar. 2005: 63).



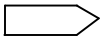
Gambar II.12 *Use Case* Model

(Sumber : Munawar; 2005: 64)

II.7.4. Pengertian *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 perbedaannya dengan *flowchart* adalah *activity diagram* bisa mendukung perilaku *pararel* sedangkan *flowchart* tidak bisa. Berikut ini adalah simbol-simbol yang sering digunakan pada activity diagram :

Tabel II.2. Simbol-simbol yang sering digunakan pada *activity* diagram

Simbol	Keterangan
	Titik Awal
	Titik Akhir
	Activity
	Pilih untuk pengambilan keputusan
	Fork; digunakan untuk menunjukkan kegiatan yang dilakukan sevara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu
	Rake; menunjukkan adanya dekomposisi
	Tanda waktu
	Tanda pengiriman
	Tanda penerimaan
	Aliran akhir (flow final)

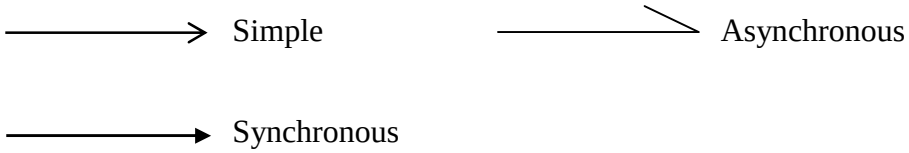
(Sumber : Munawar; 2005: 109)

II.7.5. Pengertian *Sequence diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah skenario. Diagram ini menunjukkan sejumlah contoh objek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini diluar *use case* (Munawar; 2005: 87).

Gambar II.13 Participant pada sebuah *sequence diagram*

(Sumber : Munawar; 2005: 88)



Gambar II.14 Simbol-simbol *message*

(Sumber : Munawar; 2005: 88)

II.8. Normalisasi (*Normalization*)

Perancangan basis data menghasilkan sekumpulan relasi yang saling berkerelasian dalam lingkup sebuah sistem. Untuk memenuhi batasan dalam definisi basis data maka setiap rancangan relasi perlu diuji untuk menentukan apakah relasi tersebut telah optimal. Perwujudan normalisasi adalah dekomposisi relasi menjadi relasi-relasi baru yang sederhana.

Normalisasi diartikan sebagai suatu teknik yang menstrukturkan/ mendekomposisi data dalam cara-cara tertentu untuk mencegah timbulnya permasalahan pengolahan data dalam basis data. Permasalahan yang dimaksud adalah berkaitan dengan penyimpangan-penyimpangan (*anomallies*) yang terjadi

akibat adanya kerangkapan data dalam relasi dan in-efisensi pengolahan (Martin,1975).

Proses normalisasi menghasilkan relasi yang optimal, yaitu (Martin,1975) :

1. Memiliki struktur *record* yang konsisten secara logic
2. Memiliki struktur *record* yang mudah untuk dimengerti
3. Memiliki struktur *record* yang sederhana dalam pemeliharaan
4. Memiliki struktur *record* yang mudah ditampilkan kembali untuk memenuhi kebutuhan pengguna
5. Minimalisasi kerangkapan data guna meningkatkan kinerja sistem.

Level normalisasi atau sering disebut sebagai bentuk normal suatu relasi dijelaskan berdasarkan criteria tertentu pada bentuk normal. Bentuk normal yang dikenal hingga saat ini meliputi bentuk 1NF, 2NF, 3NF, BCNF, 4NF, 5NF, DKNF, dan RUNF. Secara berturut-turut masing-masing level normal tersebut dibahas berikut ini, dimulai dari bentuk tidak normal.

1. Relasi bentuk tidak normal (*un normalized form*/ UNF)

   Relasi UNF mempunyai kriteria sebagai berikut :

   a. Jika relasi mempunyai bentuk *non flat file* (dapat terjadi akibat data disimpan sesuai dengan kedatangannya, tidak memiliki struktur tertentu, terjadi duplikasi atau tidak lengkap).
   b. Jika relasi menurut *set* atribut berulang (*non single value*)
   c. Jika relasi memuat atribut *non single value*

2. Relasi bentuk normal pertama (*first norm form*/INF)

   Relasi disebut sebagai 1NF jika memenuhi kriteria sebagai berikut :

a. Jika seluruh atribut dalam relasi bernilai atomic (*atomic value*).
b. Jika seluruh atribut dalam relasi bernilai tunggal (*single value*).
c. Jika relasi tidak memuat *set* atribut berulang.
d. Jika semua *record* mempunyai sejumlah atribut yang sama.

Permasalahan dalam 1NF adalah sebagai berikut :

a. Tidak dapat menyisipkan informasi parsial
b. Terhapusnya informasi ketika menghapus sebuah *record*
c. Pembaruan atribut *non* kunci mengakibatkan sejumlah *record* harus diperbaharui

3. Bentuk normal kedua (*second norm form*/2NF)

Relasi disebut sebagai 2NF jika memenuhi kriteria sebagai berikut :

a. Jika memenuhi kriteria 1NF
b. Jika semua atribut *non* kunci FD dan PK

Permasalahan dalam 2NF adalah sebagai berikut :

a. Kerangkapan data (data *redundancy*
b. Pembaharuan yang tidak benar dapat menimbulkan inkonsistensi data (data *inconsistency*).
c. Proses pembaharuan data tidak efisien
d. Penyimpanan pada saat penyisipan, penghapusan, dan perubahan.

4. Bentuk normal ketiga (*third norm form*/3ND)

Suatu relasi disebut sebagai 3NF jika memenuhi kriteria sebagai berikut :

a. Jika memenuhi kriteria 2NF

b. Jika setiap atribut *non* kunci tidak TDF (*non transitive dependeny*) terhadap PK

5. Bentuk normal *Boyce-Codd* (*Boyce-Codd norm form*/BCNF)

   Suatu relasi disebut sebagai BCNF jika memenuhi kriteria sebagai berikut :

   a. Jika memenuhi kriteria 3NF
   b. Jika semua atribut penentu (determinan) merupakan CK

6. Bentuk normal keempat (*forth norm form*/4NF)

   Relasi disebut sebagai 4NF jika memenuhi kriteria sebagai berikut :

   a. Jika memenuhi kriteria BCNF
   b. Jika setiap atribut di dalamnya tidak mengalami ketergantungan pada banyak nilai. Atau dengan kalimat lain, bahwa semua atribut yang mengalami ketergantungan pada banyak nilai adalah bergantung secara fungsional (*functionally dependency*).

7. Bentuk normal kelima (*fifth norm form*/5NF)

   Suatu relasi memenuhi kriteria 5NF jika kerelasian antardata dalam relasi tersebut tidak dapat direkonstruksi dari struktur relasi yang sederhana.

8. Bentuk normal kunci *domain* (*domain key norm form*/DKNF)

   Suatu relasi disebut DKNF jika setiap batasan dapat disimpulkan secara sederhana dengan mengetahui sekumpulan nama atribut dan domainnya selama menggunakan sekumpulan atribut pada kuncinya. Sangat spesifik, artinya tidak semua relasi dapat mencapai level ini

   (Edhy Sutanta; 2011 : 174-179).