BAB II

TINJAUAN PUSTAKA

II.1 Sistem

Sistem merupakan kumpulan dari unsur atau elemen - elemen yang saling berkaitan atau berinteraksi dan saling mempengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu.(Asbon Hendra ; 2012 : 167).

II.1.1. Sistem Pakar

Dalam ilmu komputer, banyak ahli yang berkonsentrasi pada pengembangan kecerdasan buatan atau *Artificial Inteligence* (AI). AI adalah suatu studi khusus di mana tujuannya adalah membuat komputer berpikir dan bertindak seperti manusia. Banyak implementasi AI dalam bidang komputer, misalnya *Decision Support System* (Sistem Pendukung Keputusan), *Robotic*, *Natural Language* (Bahasa Alami), *Neural Network* (Jaringan Saraf), dan lain-lain.

Contoh bidang lain pengembangan kecerdasan buatan adalah sistem pakar yang menggabungkan pengetahuan dan penulusuran data untuk memecahkan masalah yang secara normal memerlukan keahlian manusia. Tujuan pengembangan sistem pakar sebenarnya bukan untuk menggantikan peran manusia, tetapi untuk mensubstitusikan pengetahuan manusia ke dalam bentuk sistem, sehingga dapat digunakan oleh orang banyak.

Ada banyak manfaat yang dapat diperoleh dengan mengembangkan system pakar, antara lain :

1. Masyarakat awam non-pakar dapat memanfaatkan keahlian di dalam bidang tertentu tanpa kehadiran langsung seorang pakar.
2. meningkatkan produktivitas kerja, yaitu bertambah efisiensi pekerjaan tertentu serta hasil solusi kerja.
3. penghematan waktu dalam meyelesaikan masalah yang kompleks.
4. Memberikan penyederhanaan solusi untuk kasus-kasus yang kompleks dan berulang-ulang.
5. pengetahuan dari seorang pakar dapat didokumentasikan tanpa ada batas waktu.
6. Memungkinkan penggabungan berbagai bidang pengetahuan dari berbagai pakar untuk dikombinasikan.

Berikut ini merupakan perbandingan antara kemampuan pakar manusia dan sistem komputer yang menjadi pertimbangan pengembangan sistem pakar (Tim Penerbit Andi : 2009 : 3) adalah terdapat dalam Tabel II.1 berikut :

Tabel II.1. Perbandingan kemampuan pakar manusia dan sistem komputer

Pakar Manusia	Sistem Pakar
Terbatas waktu karena manusia membutuhkan istirahat.	Tidak terbatas karena dapat digunakan kapanpun juga.
Tempat akses bersifat lokal pada suatu tempat saja dimana pakar berada.	Dapat digunakan diberbagai tempat.
Pengetahuan bersifat variabel dan dapat berubah-ubah tergantung situasi.	Pengetahuan bersifat konsisten.
Kecepatan untuk menentukan solusi sifatnya bervariasi.	Kecepatan untuk memberikan solusi konsisten dan lebih cepat daripada manusia.

Biaya yang harus dibayar untuk konsultasi biasanya sangat mahal.	Biaya yang dikeluarkan lebih murah.

Sumber : (Tim Penerbit Andi : 2009 : 4)

Selain dari beberapa manfaat yang diperoleh, ada juga kelemahan pengembangan sistem pakar, yaitu :

1. Daya kerja dan produktivitas manusia menjadi berkurang karena semuanya dilakukan secara otomatis oleh sistem.
2. Pengembangan perangkat lunak sistem pakar lebih sulit dibandingkan dengan perangkat lunak konvensional. Hal ini dapat dilihat dari tabel II.2 perbandingan berikut ini :

Tabel II.2. Manfaat dan kelemahan pengembangan sistem pakar

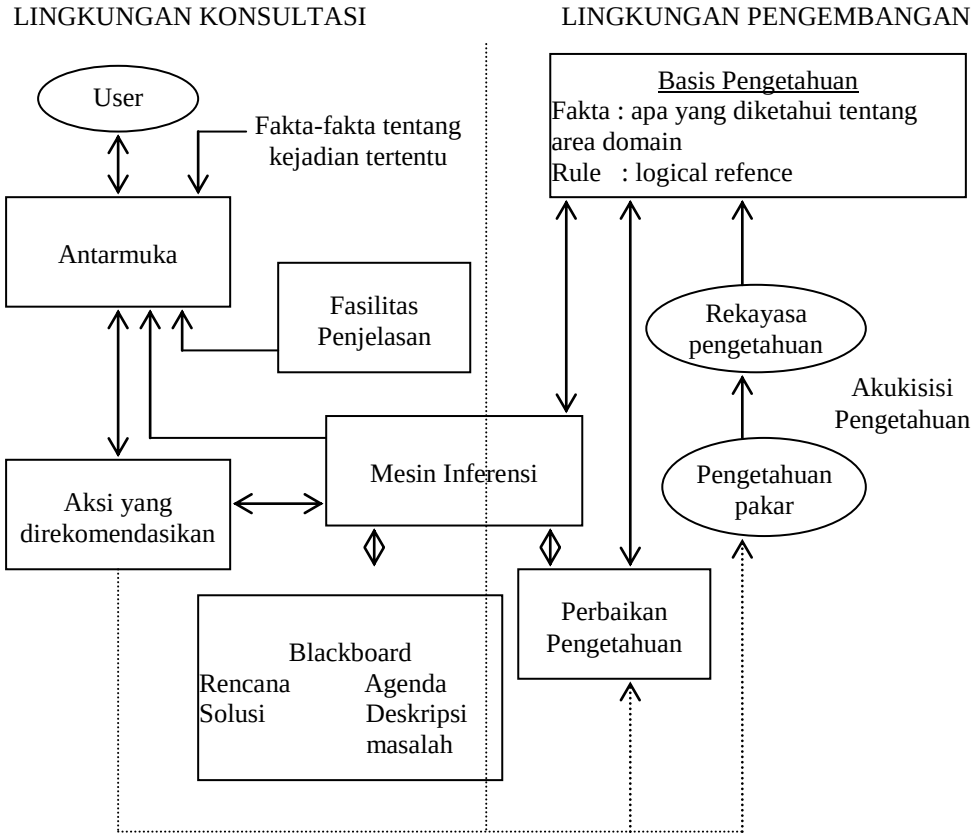
Perangkat Lunak Konvensional	Perangkat Lunak Sistem Pakar
Fokus pada solusi.	Fokus pada permasalahan.
Pengembangan dapat dilakukan secara individu.	Pengembangan dilakukan oleh tim kerja.
Pengembangan secara sekuensial.	Pengembangan secara interaktif.

Sumber : (Tim Penerbit Andi : 2009 : 4)

II.1.2. Struktur Sistem Pakar

Ada dua bagian penting dari Sistem Pakar, yaitu lingkungan pengembangan (*development environment*) dan lingkungan konsultasi (*consultation environment*). Lingkungan pengembangan digunakan oleh pembuat sistem pakar untuk membangun komponen-komponen dan memperkenalkan pengetahuan ke dalam *knowledge base* (basis pengetahuan). lingkungan konsultasi digunakan oleh pengguna untuk berkonsultasi sehingga pengguna mendapatkan pengetahuan dan

nasihat dari sistem pakar layaknya berkonsultasi dengan seorang pakar. Gambar II.1. menunjukkan komponen-komponen yang penting dalam sebuah sistem pakar.



Gambar II.1 Arsitektur sistem pakar

(Sumber: T.Sutejo : 2011 : 166)

II.1.3. Metode *Forward Chaining*

forward chaining adalah teknik pencarian yang dimulai dengan fakta yang diketahui, kemudian mencocokkan fakta-fakta tersebut dengan bagian IF dari *rules* IF-THEN. bila ada fakta yang cocok dengan bagian IF, maka rule tersebut dieksekusi. Bila sebuah rule dieksekusi, maka sebuah fakta baru(bagian THEN) ditambahkan kedalam database. Setiap kali pencocokkan berhenti tidak ada lagi rule yang bisa dieksekusi. Metode pencarian yang digunakan adalah *Depth-First*

Search (DFS), *Breadth-First Search* (BFS) atau *Best First Search*. (T.Sutejo : 2011 : 171)

II.1.4. Metode *Backward Chaining*

Menggunakan pendekatan goal-driven, dimulai dari harapan apa yang akan terjadi (hipotesis) dan kemudian mencari bukti yang mendukung (atau berlawanan) dengan harapan kita. Sering hal ini memerlukan perumusan dan pengujian hipotesis sementara. Jika suatu aplikasi menghasilkan tree yang sempit dan cukup dalam, maka gunakan *backward chaining* (Arhami : 2012 : 111).

II.2 Cerebral Palsy

Cerebral Palsy (CP, Kelumpuhan Otak Besar) adalah suatu keadaan yang ditandai dengan buruknya pengendalian otot, kekakuan, kelumpuhan dan gangguan fungsi saraf lainnya. Cerebral Palsy dan Bell's Palsy tidak sama penyakitnya walaupun sama gejalanya membuat wajah kaku, Cerebral palsy lebih sering ditemukan pada anak kecil sedangkan Bell's Palsy pada orang dewasa. Cerebral Palsy terjadi pada 1-2 dari 1.000 bayi, tetapi 10 kali lebih sering ditemukan pada bayi prematur dan.10-15% kasus terjadi akibat cedera lahir karena aliran darah ke otak sebelum/selama/ segera setelah bayi lahir.

Bayi prematur sangat rentan terhadap CP, kemungkinan karena pembuluh darah ke otak belum berkembang secara sempurna dan mudah mengalami perdarahan atau karena tidak dapat mengalirkan oksigen dalam jumlah yang memadai ke otak. Gejala biasanya timbul sebelum anak berumur 2 tahun dan pada

kasus yang berat, bisa muncul pada saat anak berumur 3 bulan. (Jurnal, Cerebral Palsy Diagnosis Dan Tata Laksana : 2006)

II.3. UML (*Unified Modeling Language*)

UML (*Unified Modeling Language*) adalah suatu alat Bantu yang sangat handal di dunia pengembangan sistem yang berorientasi objek (Munawar ; 2005 : 17). Hal ini disebabkan karena UML menyediakan bahasa pemodelan visual yang memungkinkan bagi pengembangan sistem untuk membuat cetak biru atas visi mereka dalam bentuk yang baku, mudah dimengerti serta dilengkapi dengan mekanisme yang efektif untuk berbagi (*sharing*) dan mengkomunikasikan rancangan mereka dengan yang lain.

Meskipun UML sudah banyak menyediakan diagram yang bisa membantu mendefenisikan suatu aplikasi, tidak berarti bahwa semua diagram tersebut akan bisa menjawab persoalan yang ada. Adapun tipe diagram UML yang ada seperti pada Tabel II.1.

Tabel II.3. Tipe Diagram UML

Diagram	Tujuan	Keterangan
Activity	Prilaku prosedural dan paralel	Sudah ada di UML 1
Class	Class, fitur dan relasinya	Sudah ada di UML 1
Communication	Interaksi diantara objek. Lebih menekankan kepada link	Di UML 1 disebut collaboration
Component	Struktur dan koneksi dari komponen	Sudah ada di UML 1
Composite Structure	Dekomposisi sebuah class saat runtime	Baru untuk UML 2
Deployment	Penyebaran/instalasi ke klien	Sudah ada di UML 1
Interaction Overview	Gabungan dari activity dan sequence diagram	Baru untuk UML 1
Object	Contoh konfigurasi instance	Tidak resmi ada di UML 1
Package	Struktur hierarki saat kompilasi	Tidak resmi ada di

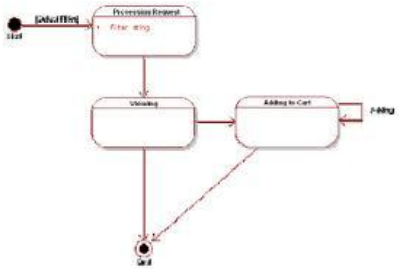
		UML 1
Sequence	Interaksi antara objek. Lebih menekankan pada urutan.	Sudah ada di UML 1
State Machine	Bagaimana event mengubah sebuah objek	Sudah ada di UML 1
Timing	Interaksi antar objek. Lebih menekankan pada waktu	Sudah ada di UML 1
Use Case	Bagaimana user berinteraksi dengan sebuah sistem	Sudah ada di UML 1

Sumber : (Munawar ; 2005 : 23)

II.3.1. Notasi Dasar UML

1. *Statechart Diagram*

Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya) suatu objek pada sistem sebagai akibat dari *stimuli* yang diterima. Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).



Gambar II.2 Contoh Statechart Diagram

Sumber : (Munawar ; 2005 : 3)

2. *Class*

Class, dalam notasi UML digambarkan dengan kotak. Nama class menggunakan huruf besar diawal kalimatnya dan diletakkan diatas kotak. Bila *class* mempunyai nama yang terdiri dari 2 suku kata atau lebih, maka semua suku

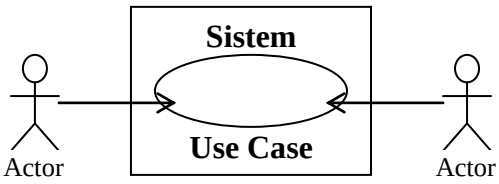
kata digabungkan tanpa spasi dengan huruf awal tiap suku kata menggunakan huruf besar. Berikut notasi *class* dalam UML:

Gambar II.3 Notasi Class di UML

Sumber : (Munawar ; 2005 : 35)

3. *Use Case*

Use Case adalah alat bantu terbaik guna menstimulasi pengguna potensial untuk mengatakan tentang suatu *system* dari sudut pndangnya. Tidak selalu mudah bagi pengguna untuk menyatakan bagaimana mereka bermaksud menggunakan sebuah *system*. Karena system pengembangan tradisional sering ceroboh dalam melakukan analisis, akibatnya pengguna seringkali susah menjawabnya tatkala dimintai masukan tentang sesuatu. Notasi *use cas*e dapat dilihat pada gambar II.4 :



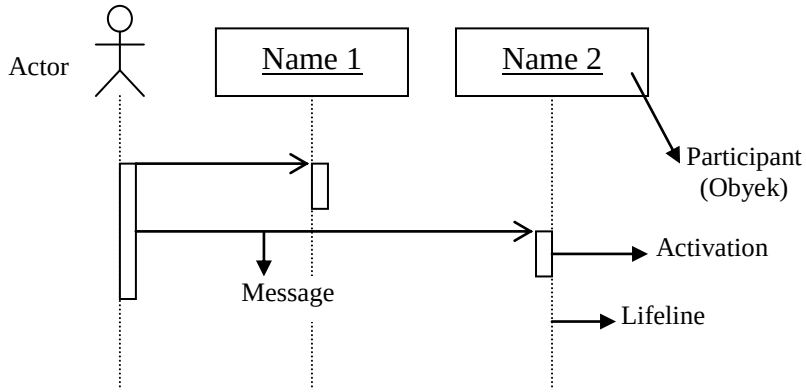
Gambar II.4 Notasi Use Case pada UML

Sumber : (Munawar ; 2005 : 64)

4. *Sequence Diagram*

Sequence diagram digunakan untuk menggambarkan perilaku pada sebuah scenario. Diagram ini menunjukkan sejumlah contoh obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek ini dalam *use case*.

Komponen utama squence diagram terdiri atas obyek yang dituliskan dengan kotak segiempat bernama. Message dengan tanda panah dan waktu yang ditunjukkan dengan progress vertical. Berikut Contoh *sequence diagram* :



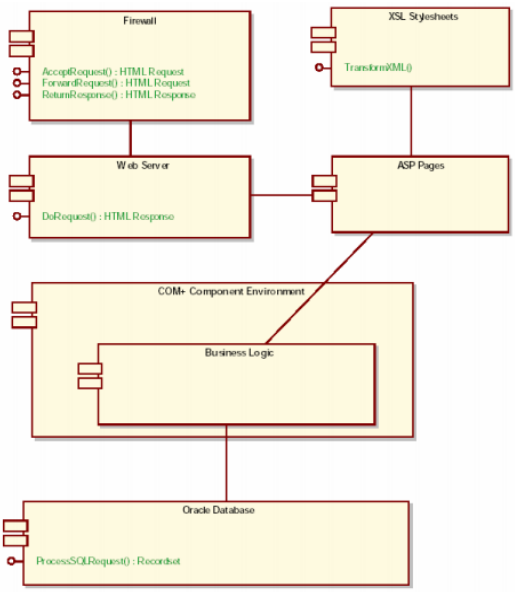
Gambar II.5 Simbol-simbol yang ada pada Sequence Diagram

Sumber : (Munawar ; 2005 : 89)

5. Component Diagram

Component Diagram menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) diantaranya. Komponen piranti lunak adalah modul berisi *code*, baik berisi *source code* maupun *binary code*, baik yang muncul pada *compile time*, *link time*, maupun *run time*. Umumnya komponen terbentuk dari beberapa *class* dan/atau *package*, tapi dapat juga dari komponen-komponen yang lebih kecil.

Komponen dapat juga berupa *interface*, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.

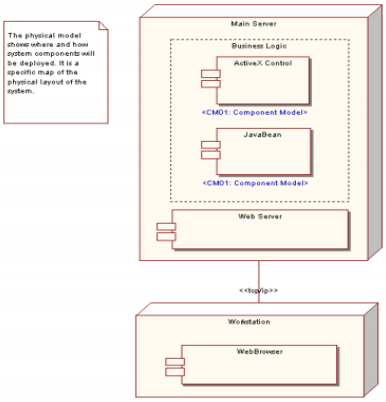


Gambar II.6 Contoh Component Diagram

Sumber : (Munawar ; 2005 : 75)

6. Deployment Diagram

Deployment/physical diagram menggambarkan detail bagaimana komponen di-*deploy* dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal.



Gambar II.7 Contoh Deployment Diagram

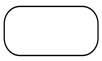
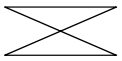
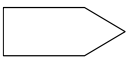
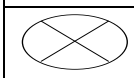
Sumber : (Munawar ; 2005 : 43)

7. *Activity Diagram*

Activity diagram adalah teknik untuk mendiskripsikan 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 prilaku paralel sedangkan *flowchart* tidak bisa.

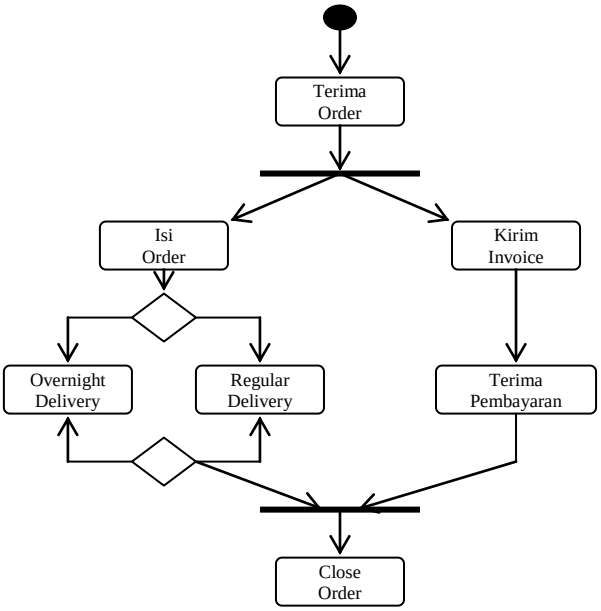
Berikut adalah simbol-simbol yang sering digunakan pada saat pembuatan *activity diagram*.

Tabel II.4 Simbol-simbol yang sering dipakai pada Activity Diagram

Simbol	Keterangan
	Titik awal
	Titik akhir
	Activity
	Pilihan untuk pengambilan keputusan
	Fork; digunakan untuk menunjukkan kegiatan yang dilakukan secara 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)

Adapun contoh dari *Activity Diagram* dapat di lihat pada Gambar II.8.



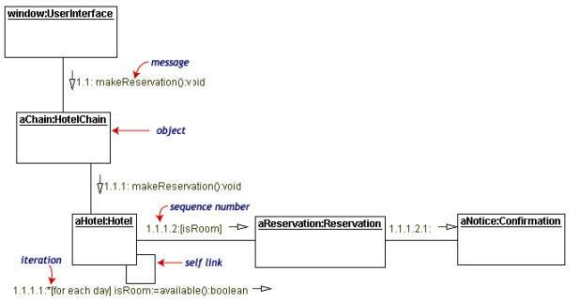
Gambar II.8 Contoh Activity Diagram Sederhana

Sumber : (Munawar ; 2005 : 111)

8. Collaboration Diagram

Collaboration diagram juga menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*.

Setiap *message* memeliki *sequence number*, dimana *message* dari level tertinggi memiliki nomor 1. *Messages* dari level yang sama memiliki prefiks yang sama.



Gambar II.9 Contoh Collaboration Diagram

Sumber : (Munawar ; 2005 : 12)

II.4. Pengertian Database

Database merupakan komponen terpenting dalam pembangunan SI, karena menjadi tempat untuk menampung dan mengorganisasikan seluruh data yang ada dalam sistem, sehingga dapat dieksplorasi untuk menyusun-menyusun informasi-informasi dalam berbagai bentuk. *Database* merupakan himpunan kelompok data yang saling berkaitan. Data tersebut diorganisasikan sedemikian rupa agar tidak terjadi duplikasi yang tidak perlu, sehingga dapat diolah atau di eksplorasi secara cepat dan mudah untuk menghasilkan informasi (Budi Sutedjo Dharma Oetomo ; 2006: 99).

II.4.1 Konsep Database

Database (basis data) adalah sistem penyimpanan beragam jenis data dalam sebuah entitas yang besar untuk diolah sedemikian rupa agar mudah dipergunakan kembali. Dengan menggunakan komputer, konsep pengolahan database tradisional dapat diotomasi sehingga memudahkan pekerjaan. Data yang disimpan bisa sangat variatif (angka, teks, gambar, suara, dan jenis data multi-media lainnya).

Aplikasi manajemen database mengenal dua macam bentuk database:

1. *Flat-file* adalah semua record tersimpan dalam satu tabel.
2. Database relasional adalah memiliki banyak tabel yang saling terkait, dengan sebuah unsur data yg berfungsi sebagai pengait (*primary key*).

Dengan semakin banyaknya data yang dikelola, hampir tidak mungkin bahwa semua rekaman (*record*) disimpan dalau satu tabel. Manfaat database relasional adalah membuat sistem pengolahan data menjadi lebih efisien dan tabel data dapat dipilahkan dengan kategori yang berbeda. Fungsi *primary key* sangat penting dalam menemukan relasi dan logika kaitan antar tabel.

Pengimplementasian database dapat dilakukan secara terdistribusi dan tersentralisasi. Terdistribusi merupakan suatu konsep yang menerapkan lebih dari satu database. Dan tersentralisasi menerapkan satu database secara terpusat. Pada database terdapat tiga tingkatan atau level data, yaitu :

1. Level Fisik (*Phisycal Level*).

   Level fisik merupakan level paling rendah karena menggambarkan bagaimana data disimpan pada kondisi yang sebenarnya pada server.

2. Level Konsepsual (*Conceptual Level*).

   Merupakan level yang menggambarkan data yang disimpan pada database dan menjelaskan secara keseluruhan hubungan antar data tersebut.

3. Level Pandangan (*View Level*).

Merupakan level yang menggambarkan sebagian dari seluruh database sesuai dengan kebutuhan pengguna (Budi Sutedjo Dharma Oetomo ; 2006: 100).

II.4.1. Hierarki Data Dalam Database

Data dalam sebuah *database* disusun berdasarkan sistem hierarki yang unik, yaitu:

1. Database, merupakan kumpulan file yang saling terkait satu sama lain, misalnya file data induk karyawan, file jabatan file penggajian dan lain sebagainya. Kumpulan file yang tidak saling terkait satu sama lain tidak dapat disebut *database*, misalnya file data induk karyawan, file tamu undangan perkawinan, file barang retail pasar swalayan.
2. File, yaitu kumpulan dari *record* yang saling terkait dan memiliki format *field* yang sama dan sejenis.
3. Record, yaitu kumpulan *field* yang menggambarkan suatu unit data individu tertentu.
4. Field, yaitu atribut dari *record* yang menunjukkan suatu item dari data, seperti nama, alamat, dan lain sebagainya.
5. Byte, yaitu atribut dari *field* yang berupa huruf yang membentuk nilai dari sebuah *field*. Huruf tersebut dapat berupa numerik maupun abjad atau karakter khusus
6. Bit, yaitu bagian terkecil dari data secara keseluruhan, yaitu berupa karakter ASCII nol atau satu yang merupakan komponen pembentuk *byte* (Budi Sutedjo Dharma Oetomo ; 2006: 102).

II.4.2. *MySQL*

MySQL adalah salah satu perangkat lunak sistem menejemen basis data (database) SQL atau sering disebut dengan DNMS (Database Management

System). Berbeda dengan basis data konversional. Mysql memiliki kelebihan yaitu bersifat multithead, dan multi-user serta mendukung sistem jaringan. Mysql diitribusikan secara geratis dibawah lisensi GNU General Public Licence (GPL), namun ada juga versi komersial bagi kalangan tertentu yang menginginkannya. (A.M. Hirin & Virgi : 2011 : 27).

II.5. Kamus Data

Kamus data ikut berperan dalam perancangan dan pengembangan SI karena peralatan ini berfungsi untuk :

1. Menjelaskan arti aliran data dan penyimpanan dalam penggambaran dalam data flow diagram.
2. Mendeskripsikan komposisi paket data yang bergerak melalui aliran, misalnya data alamat diurai menjadi nama jalan, nomor, kota, negara dan kode pos.
3. Menjelaskan spesifikasi nilai dan satuan yang relavan terhadap datayang mengalir dalam sistem tersebut.

   Sejumlah symbol yang digunakan dalam penggambaran kamus data ditunjukkan dalam tabel II.3.

Tabel II.5. Notasi Kamus Data

Notasi	Arti
=	Terbentuk dari (is composed) atau terdiri dari (consist of) atau sama dengan (is equivalent of)
+	AND
[]	Salah satu dari (memilih salah satu dari elemen-elemen data di dalam kurung bracket ini)
|	Sama dengan simbol []
M{ }M	Intensi (elemen data didalam kurung brace berinterasi mulai

	minimum N kali dan maksimum M kali)
()	Optional (elemen data di dalam kurung parenthesis sifatnya optional, dapat ada dan dapat tidak ada)
*	Keterangan setelah tanda ini adalah komentar

Sumber : (Budi Sutedjo Dharma Oetomo: 2006: 118)

II.6. Entity Relationship Diagram – ERD

II.6.1. Model-model Data

Struktur yang mendasari suatu basisdata adalah model data yang merupakan kumpulan alat-alat konseptual untuk mendeskripsikan data, relasi data, data semantik, dan batasan konsistensi. Untuk mengilustrasikan konsep model data, berikut disajikan dua model data, yaitu *entity relationship model* dan *relational model.* Kedua model menyediakan cara mendeskripsikan rancangan basisdata pada tingkatan logis (Janner Simarmata & Imam Prayudi: 2006: 59).

II.6.2. Entity Relationship Model

Entity Relationship (ER) data model didasarkan pada persepsi terhadap dunia nyata yang tersusun atas kumpulan objek-objek dasar yang disebut entitas dan hubungan antar objek. Entitas adalah sesuatu atau objek dalam dunia nyata yang dapat dibedakan dari objek lain. Sebagai contoh, masing-masing mahasiswa adalah entitas dan mata kuliah dapat pula dianggap sebagai entitas.

Entitas digambarkan dalam basisdata dengan kumpula atribut. Misalnya atribut nim, nama, alamat dan kota bisa menggambarkan data mahasiswa tertentu dalam suatu universitas. Atribut-atribut membentuk entitas mahasiswa. Demikian pula, atribut kodeMK, namaMK, dan SKS mendeskripsikan entitas mata kuliah.

Atribut NIM digunakan untuk mengidentifikasi mahasiswa secara unik karena dimungkinkan terhadap dua mahasiswa dengan nama, alamat, dan kota yang sama. Pengenal unik harus diberikan pada masing-masing mahasiswa.

Relasi adalah hubungan antara beberapa entitas. Sebagai contoh, relasi menghubungkan mahasiswa dengan mata kuliah yang di ambilnya. Kumpulan semua entitas bertipe sama disebut kumpulan entitas (*entity set)*, sedangkan kumpulan semua relasi bertipe sama disebut kumpulan relasi (*relationship set)*.

Struktur logis (skema *database*) dapat ditunjukkan secara grafis dengan diagram ER yang dibentuk dari komponen-komponen berikut :

Tabel II.6. Notasi ERD (Entity Relationship Diagram)

	Persegi panjang mewakili kumpulan entitas
	Elips mewakili atribut
	Belah ketupat mewakili relasi
	Garis menghubungkan atribut dengan kumpulan entitas dan kumpulan entitas dengan relasi

Sumber : (Janner Simarmata & Imam Prayudi: 2006: 60)

II.6.3. Normalisasi

Normalisasi adalah teknik perancangan yang banyak digunakan sebagai pemandu dalam merancang basis data relasional. Pada dasarnya, normalisasi adalah proses dua langkah yang meletakkan data dalam bentuk tabulasi dengan

menghilangkan kelompok berulang lalu menghilangkan data yang terduplikasi dari tabel relational.

Teori normalisasi didasarkan pada konsep bentuk normal. Sebuah tabel relasional dikatakan berada pada bentuk normal tertentu jika tabel memenuhi himpunan batasan tertentu. Ada lima bentuk normal yang telah ditemukan (Janner Simarmata & Imam Prayudi ; 2009: 77) .

1. **Bentuk Normal Pertama (1NF/*First Normal Form*)**, bentuk normal pertama adalah suatu bentuk relasi dimana atribut bernilai banyak (*multivalues atribute*) telah dihilangkan sehingga kita akan menjumpai nilai tunggal (mungkin saja nilai *null*) pada perpotongan setiap baris dan kolom satu nilai untuk irisan baris dan kolom pada tabel.
2. **Bentuk Normal Kedua (2NF/*Second Normal Form*)**, semua kebergantungan fungsional (*functional dependeny*) yang bersifat sebagian (*partial functional dependeny*) telah dihilangkan.
3. **Bentuk Normal Ketiga (3NF/*Third Normal Form*)**, semua kebergantungan transitif (*transitive dependeny*) telah dihilangkan.
4. **Boyce-Codd Normal Form (BCNF/*Boyce-Codd Normal Form*)**, semua anomali yang tersisa dari hasil penyempurnaan kebergantungan fungsional (*functional dependeny*) diatas telah dihilangkan.
5. **Bentuk Normal Keempat (4NF/*Fifth Normal Form*)**, semua anomali yang berasal dari kebergantungan banyak-nilai (*multivalues dependeny*) telah dihilangkan.

Tujuan normalisasi adalah membuat kumpulan tabel relasional yang bebas dari data berulang yang dapat dimodifikasi secara benar dan konsisten. Ini berarti bahwa semua tabel pada basisdata relasional harus berada pada bentuk normal ketiga (3NF). Sebuah tabel relasional berada pada 3NF jika dan hanya jika semua kolom bukan kunci adalah (a) saling independen dan (b) sepenuhnya tergantung pada kunci utama. Saling independen berarti bahwa tidak ada kolom bukan kunci yang tergantung pada senbarang kombinasi kolom lainnya. Dua bentuk normal pertama adalah langkah antara untuk mencapai tujuan, yaitu mempunyai semua tabel dalam 3NF (Stephens and Plew, 2000) (Janner Simarmata & Imam Prayudi ; 2009 : 78).

II.7 Bahasa Pemrograman Visual Basic 2010

Visual Basic 2010 merupakan salah satu bagian dari produk pemograman terbaru yang dikeluarkan oleh microsoft, yaitu microsoft visual 2010. Visual studio merupakan produk pemograman andalan dari Microsoft corporation, yang didalamnya berisi beberapa jenis IDE pemograman seperti visual basic, visual C++, visual Web Devoloper, Visual C#, dan visual F#.

Semua IDE pemograman tersebut sudah mendukung penuh implementasi Net Framework terbaru, yaitu Net Framework 4.0 yang merupakan pengembangan dari Net Framework 3.5. visual basic 2010 merupakan versi perbaikan dan pengembangan dari versi pendahulunya, yaitu visual basic 2008. (Wahana Komputer: 2010: 2-3).