BAB II

TINJAUAN PUSTAKA

II.1. Pengertian Sistem Pakar

Sistem pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yang dipikirkan oleh pakar. Pakar yang dimaksud disini adalah orang yang mempunyai keahlian khusus yang dapat menyelesaikan masalah yang tidak dapat diselesaikan oleh orang awam. Sebagai contoh, dokter adalah seorang pakar yang mampu mendiagnosis penyakit yang diderita pasien serta dapat memberikan penatalaksanaan terhadap penyakit tersebut. Tidak semua orang dapat mengambil keputusan mengenai diagnosis dan memberikan penatalaksanaan suatu penyakit. Contoh yang lain, montir adalah seorang yang punya keahlian dan pengalaman dalam menyelesaikan kerusakan mesin motor/mobil, psikolog adalah orang yang ahli dalam memahami kepribadian seseorang, dan lain – lain. Sistem pakar, yang mencoba memecahkan masalah yang biasanya hanya bisa dipecahkan oleh seorang pakar, dipandang berhasil ketika mampu mengambil keputusan seperti yang dilakukan oleh pakar aslinya baik dari sisi proses pengambilan keputusannya maupun hasil yang diperoleh. Sebuah sistem pakar memiliki 2 komponen utama yaitu basis pengetahuan dan mesin inferensi. Basis pengetahuan merupakan tempat penyimpanan pengetahuan dalam memori komputer, dimana pengetahuan ini diambil dari pengetahuan pakar. (Kusrini; 2010:3).

II.2. Pengertian Diagnosis

Menurut Prof. Dr. Dr. Daldiyono (2010:49). Diagnosis adalah istilah yang menunjukkan pada nama penyakit yang ada pada pasien yang perlu dirumuskan (ditentukan) oleh dokter. Perumusan diagnosis sebenarnya sangat rumit (ruwet) sekaligus merupakan seni karena menentukan sesuatu kesimpulan dengan bahan yang sangat tidak menentukan (*to make a decision with the uncertain data*), sedangkan keputusan tersebut tidak boleh salah. Dalam realitas pekerjaan sehari – hari, bagi para dokter ada tiga kemungkinan yaitu :

a. Dokter bekerja berdasarkan rumusan masalah, sedangkan hipotesis diagnosis kerja masih sangat umum.
b. Dokter berusaha merumuskan diagnosis, sebagian kasus berhasil sebagian lagi tidak sampai karena data klinik belum cukup atau perbendaharaan pengetahuan belum cukup.
c. Dokter selalu berusaha sampai pada diagnosis.

Apabila dokter mengambil salah satu sikap dari ketiga hal tersebut tidaklah salah, apabila bekerja dengan target (c) maka dokter tersebut berpikir lebih mendalam. Target (b) adalah yang biasa dicapai oleh dokter pada umumnya sedang target (a) kadang – kadang harus dilakukan oleh dokter umum (*general practitioner*) dan dokter keluarga (*family physicians*). Namun, yang perlu direkankan disini adalah untuk setiap target, berpikir haruslah dicapai suatu perumusan hipotesis yang berbentuk diagnosis kerja dan diagnosis banding.

Buku ini berusaha membawa para pembaca, khususnya mahasiswa sampai pada perumusan hipotesis berbentuk diagnosis kerja dan diagnosis banding atau sampai pada diagnosis yang definitif (pasti).

II.3. Pengenalan Penyakit Ateroskleorosis

Ateroskleorosis atau kekakuan pembuluh darah arteri, atau ada yang menyebutnya pengerasan arteri, adalah suatu keadaan di mana terjadi penimbunan lemak bercampur kalsium dan sel darah pada dinding pembuluh arteri. Felix Marchand adalah orang pertama yang memperkenalkan isttilah ateroskleorosis pada tahun 1904. Dia menuduh ateroskleorosis yang bertanggung jawab terhadap semua proses penyumbatan di arteri. Pada tahun 1908, A.I. ignatowski di Rusia melaporkan ada kaitan antara makanan kaya kolesterol dengan kejadian ateroskleorosis. Kemudian Windaus menunjukkan bahwa lesi ateroma mengandung 6 kali lebih banyak kolesterol bebas dan 20 kali lebh banyak kolesterol teresterifikasi dibandingkan ateri normal. Pada tahun 1913, Nikolai Anichkov mendemonstrasikan bahwa ateroskleorosis dapat diinduksi pada kelinci percobaan yang diberi makanan kaya kolesterol. Walaupun demikian, ateroskleorosis merupakan suatu proses penyakit dengan penyebab yang multifaktor selain kolesterol, dalam hal ini ada juga peranan genetis sehingga alur terjadinya proses tersebut, lokasi arteri yang terkena, gradasi ateroskleorosis, dan akselerasinya tidak sama untuk semua orang. (Prof. Dr. Peter Kabo; 2011:21).

II.4. Sejarah Metode Fuzzy

Teori himpunan *fuzzy* diperkenalkan pertama kali oleh *Lotfi A. Zadeh* pada tahun 1965. Ada beberapa alasan mengapa orang menggunakan logika *fuzzy*,

antara lain yaitu konsep logika *fuzzy* mudah dimengerti, logika *fuzzy* sangat *fleksibel*, logika *fuzzy* memiliki toleransi terhadap data – data tidak tepat, logika fuzzy mampu memodelkan fungsi – fungsi *nonlinear* yang sangat kompleks, logika *fuzzy* dapat membangun dan mengaplikasikan pengalaman – pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan, logika *fuzzy* dapat bekerjasama dengan teknik – teknik kendali secara *konvensial*, dan logika *fuzzy* didasarkan pada bahasa alami. Dalam logika *fuzzy* dikenal berhingga keadaan dari nilai "0" sampai ke nilai "1". Logika *fuzzy* tidak hanya mengenal dua keadaan tetapi juga mengenal sejumlah keadaan yang berkisar dari kedaan salah sampai keadaan benar. Ada beberapa hal yang perlu diketahui dalam memahami sistem *fuzzy*, antara lain *variabel fuzzy*, himpunana *fuzzy*, semesta pembicaraan dan domain. *Variabel fuzzy* merupakan *variabel* yang hendak dibahas dalam suatu sistem *fuzzy*. Himpunan *fuzzy* adalah himpunan yang tiap elemennya mempunya derajat keanggotaan tertentu terhadap himpunannya. Himpunan *fuzzy* memiliki dua atribut, yaitu *linguistik* dan *numeris* yaitu suatu nilai (angka) yang menunjukkan ukuran dari suatu *variabel*. Domain himpunan *fuzzy* adalah keseluruhan nilai yang diijinkan dalam semesta pembicaraan dan boleh dioperasikan dalam suatu himpunan *fuzzy*, nilai domain dapat berupa bilangan positif maupun bilangan *negative*. Logika *fuzzy* memiliki beberapa karakteristik yaitu himpunan *fuzzy* dan fungsi keanggotaan. Pada logika *boolean*, sebuah individu dipastikan sebagai anggota dari salah satu himpunan saja, sedangkan pada himpunan *fuzzy* sebuah individu dapat masuk pada dua himpunan yang berbeda. Seberapa besar eksitensinya dalam himpunan tersebut dapat dilihat pada

nilai keanggotaannya. Himpunan *fuzzy A* pada semesta *X* dinyatakan sebagai himpunan pasangan berurutan (*set of ordered pairs*) baik diskrit maupun kontinu.

$A = \{(x, \mu_A (x)) \mid x€ X\}$

Dimana $\mu_A (x)$ fungsi keanggotaan himpunan *fuzzy A.* Fungsi keanggotaan memetakan setiap *x€ X* pada suatu nilai antara [0,1] yang disebut derajat keanggotaan (*membership grade* atau *membership value*). (Jurnal Informatika, Sri Yulianto; 2011 : 160).

Fuzzy Logic atau sistem *fuzzy* merupakan sautu cara yanng tepat untuk memetakan suatu ruang *input* ke dalam suatu ruang *output*. Beberapa alasan digunakannya *fuzzy logic*, antara lain :

a. Konsep *fuzzy logic* mudah dimengerti, karena didalam logika *fuzzy* terhadap konsep matematis sederhana dan mudah dimengerti yang mendasari penalaran *fuzzy*.
b. *Fuzzy logic* sangat fleksibel.
c. *Fuzzy logic* memiliki toleransi terhadap data yang tidak tepat.
d. *Fuzzy logic* mampu memodelkan fungsi – fungsi nonlinier yang sangat kompleks.
e. *Fuzzy logic* dapat bekerjasama dengan teknik – teknik kendali secara konvensional.
f. *Fuzzy logic* didasarkan pada bahasa alami.

Fuzzy logic dapat membangun dan mengaplikasikan pengalaman – pengalaman para pakar secara langsung tanpa harus melalui proses pelatihan. (Jurnal Teknik Informatika, Budi Soesilo; 2010 : 6).

Logika yang hanya berdasarkan atas 2 nilai kebenaran yaitu *TRUE* (1) dan *FALSE* (0) kadang – kadang dirasakan kurang lengkap untuk menyatakan logika berpikir manusia. Sehingga dikembangkan logika yang tidak hanya bernilai 0 atau 1 tapi menggunakan logika yang punya interval nilai antara [0,1] yang disebut dengan logika samar (*Fuzzy Logic*). *Fuzzy Logic* (FL) diperkenalkan pada tahun 1965 oleh Lotfi A. Zadeh, seorang profesor di bidang ilmu komputer, Universitas California, *Barkeley*. *Fuzzy logic* dipakai untuk menyatakan data atau informasi yang bersifat tidak pasti atau samar. Tapi sebernya sejarah *fuzzy logic* dimulai sejauh sebelumnya yaitu ketika jaman yunani kuno. *Aristotle* dan beberapa filsuf lainnya, dalam rangka menemukan teori logika dia mengemukakan hukum – hukum yang disebut *"Laws Of Thought"* salah satu diantaranya adalah *"Law of excluded Middle"* yang menyatakan bahwa setiap pernyataan (propotition) harus bernilai *TRUE* (T) atau *FALSE* (F). Bahkan ketika permindes mengusulkan versi pertama dari hukum tersebut (400 BC) langsung mendapatkan pertentangan dari Heraclitus yang menyatakan bahwa setiap peryataan hanya bernilai *TRUE* dan *NOT TRUE*. Pada saat itu Plato yang meletakkan pondasi bagi *Fuzzy Logic*, menyatakan bahwa ada daerah ketiga (selain *TRUE* dan *FALSE*). Salah satu pernyataan alternatif yang berbeda dengan logika dengan 2 nilai kebenaran (Aristotle) pertama kali ditemukan oleh Lucasiewicz (1920). Dia menemukan logika dengan 3 nilai kebenaran beserta dengan penjelasan matematiknya. Nilai ke-3 dia sebut dengan istilah "mungkin" (*possible*). Dan diberikan nilai numerik yaitu antara *TRUE* (1) dan *FALSE* (0). Selanjutnya Lucasiewicz mengemukakn tentang logika dengan 4 nilai kebenaran, 5 nilai kebenaran, dan kemudian

menyatakan bahwa logika memiliki nilai tak berhingga (*infinite*). Logika dengan 3 nilai dan logika dengan nilai tak berhingga yang paling menarik. Tapi selanjutnya dia lebih memilih logika dengan 4 nilai kebenaran karena paling mudah disesuaikan dengan logika Aristotle (2 nilai kebenaran). Juga perlu dicatat Knuth, juga menyatakan logika dengan 3 nilai kebenaran hampir sama seperti Lucasiewicz. Knuth berspekulasi bahwa matematik akan menjadi lebih nyaman jika dibandingkan secara tradisional dengan hanya 2 nilai kebenaran. Ide dari logika dengan nilai tak berhingga sudah diperkenalkan oleh Lotfi A. Zadeh dalam tulisannya yang berjudul tentang "*Fuzzy Sets*" (himpunan *fuzzy*) disertai dengan penjelasan matematik teori himpunan *fuzzy* dan juga tentang logika *fuzzy*. Dalam teori ini juga dijelaskan tentang pembentukan fungsi keanggotaan (*membership fuction*) yang beroperasi pada *range* nilai antara [0,1]. Disamping itu juga diusulkan tentang operasi – operasi matematika logika yang pada prinsipnya merupakan pengembangan dari logika klasik. *Fuzzy Logic* sudah memberikan perubahan dalam pengembilan keputusan dimana kemampuan berpikir manusia yang tidak pasti dapat dipakai dalam sistem berbasis pengetahuan. Teori *Fuzzy Logic* sudah menyediakan teori matematika untuk menampung ketidakpastian proses berpikir manusia. Beberapa ciri dari *fuzzy logic* (Zadeh, 1992) adalah :

a. Dalam FL, logika pasti (*exact*) dianggap sebagai kasus terbatas dari logika tidak pasti (approximate).
b. Dalam FL, segala sesuatu (pernyataan) ditentukan berdasarkan tingkatan (degree).

c. Dalam FL, pengetahuan merupakan kumpulan dari batasan – batasan yang elastis atau tidak pasti (*fuzzy*).
d. Pengambilan keputusan adalah proses peralihan dari batasan – batasan elastis atau tidak pasti.
e. Semua sistem logika dapat dibuat menjadi samar (*fuzzy*).

Ada 2 ciri utama dari sistem *fuzzy* sehingga sistem ini dapat diterapkan dengan baik pada beberapa aplikasi tertentu :

a. Sistem *fuzzy* sangat cocok untuk logika berpikir yang tidak pasti, khususnya untuk sistem yang sulit dimodelkan secara matematika.
b. FL, membolehkan pengambilan keputusan dengan nilai perkiraan atau berdasarkan informasi yang tidak lengkap atau tidak pasti. (Soenjono Dardjowidjojo ; 2011 : 28-29).

II.5. Normalisasi

Menurut Abdul Kadir (2010:66). Istilah normalisasi berasal dari E. F. Codd, salah seorang perintis teknologi basis data. Selain dipakai sebagai metodologi tersendiri untuk menciptakan struktur tabel (relasi) dalam basis data (dengan tujuan untuk mengurangi kemubaziran data), normalisasi terkadang hanya dipakai sebagai perangkat verifikasi terhadap tabel – tabel yang dihasilkan oleh metodologi lain. Normalisasi memberikan panduan yang sangat mebantu bagi pengembang untuk mencegah penciptaan struktur tabel yang kurang fleksibel atau mengurangi ketidakefisienan. Kronke mendefinisikan normalisasi sebagai proses untuk mengubah suatu relasi yang memiliki masalah tertentu ke dalam dua

buah relasi atau lebih yang tak memiliki masalah tersebut. Masalah yang dimaksud oleh Kronke ini sering disebut dengan istilah anomali.

II.5.1. Bentuk-bentuk Normalisasi

a. Bentuk tidak normal

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

b. Bentuk normal tahap pertama (1” Normal Form)

Definisi :

Sebuah table disebut 1NF jika :

- Tidak ada baris yang duplikat dalam tabel tersebut.
- Masing-masing cell bernilai tunggal

Catatan: Permintaan yang menyatakan tidak ada baris yang duplikat dalam sebuah tabel berarti tabel tersebut memiliki sebuah kunci, meskipun kunci tersebut dibuat dari kombinasi lebih dari satu kolom atau bahkan kunci tersebut merupakan kombinasi dari semua kolom.

Berikut ini akan dicontohkan normalisasi dari tabel kuliah yang memiliki atribut : kode_kul, nama_kul, sks, semester, waktu, tempat, dan nama_dos.

Tabel kuliah tersebut tidak memenuhi normalisasi pertama, karena terdapat atribut waktu yang tergolong ke dalam atribut bernilai banyak. Agar tabel tersebut dapat memenuhi 1NF, maka solusinya adalah dengan mendekomposisi tabel kuliah menjadi :

- Tabel kuliah (kode_kul, nama_kul, sks, semester, nama_dos).

- Tabel jadwal (kode_kul, waktu, ruang).

c. Bentuk normal tahap kedua (2nd normal form)

Bentuk normal kedua (2NF) terpenuhi jika pada sebuah tabel semua atribut yang tidak termasuk dalam primary key memiliki ketergantungan fungsional pada primary key secara utuh.

Sebuah tabel dikatakan tidak memenuhi 2NF, jika ketergantungannya hanya bersifat parsial (hanya tergantung pada sebagian dari primary key). Bentuk normal kedua akan dicontohkan berikut.

Misal tabel nilai terdiri dari atribut kode_kul, nim dan nilai. Jika pada tabel nilai. Misalnya kita tambahkan sebuah atribut yang bersifat redundan, yaitu nama_mhs, maka tabel nilai ini dianggap melanggar 2NF.

Primary key pada tabel nilai adalah (kode_kul, nim).

Penambahan atribut baru (nama_mhs) akan menyebabkan adanya ketergantungan fungsional yang baru yaitu nim > nama_mhs. Karena atribut nama_mhs ini hanya memiliki ketergantungan persial pada primary key secara utuh (hanya tergantungn pada nim, padahal nim hanya bagian dari primary key). Bentuk normal kedua ini dianggap belum memadai karena meninjau sifat ketergantungan atribut terhadap atribut terhadap primary key saja.

d. Bentuk normal tahap ketiga (3rd normal form)

Sebuah tabel dikatakan memenuhi bentuk normal ketiga (3NF), jika untuk setiap ketergantungan fungsional dengan notasi X -> A, dimana A mewakili semua atribut tunggal di dalam tabel yang tidak ada di dalam X, maka :

- X haruslah superkey pada tabel tersebut.
- Atau A merupakan bagian dari primary key pada tabel tersebut.

Misalkan pada tabel mahasiswa, atribut alamat_mhs dipecah kedalam alamat_jalan, alamat_kota dan kode_pos. Bentuk ini tidak memenuhi 3NF, karena terdapat ketergantungan fungsional baru yang muncul pada tabel tersebut, yaitu :

alamat_jalan, nama_kota – kode_pos

Dalam hal ini (alamat_jalan, nama_kota) bukan superkey sementara kode_pos juga bukan bagian dari primary key pada tabel mahasiswa. Jika tabel mahasiswa didekomposisi menjadi tabel mahasiswa dan tabel alamat, maka telah memenuhi 3NF. Hal itu dapat dibuktikan dengan memeriksa dua ketergantungan fungsional pada tabel alamat tersebut, yaitu :

alamat_jalan, nama_kota – kode_pos

kode_pos – nama_kota

Ketergantungan fungsional yang pertama tidak melanggar 3NF, karena (alamat_jalan, nama_kota) merupakan superkey (sekaligus sebagai primary key) dari tabel alamat tersebut. Demikian juga dengan ketergantungan fungsional yang kedua meskipun (kode_pos) bukan merupakan supeerkey, tetapi nama_kota merupakan bagian dari primary key dari tabel alamat. Karena telah memenuhi 3NF, maka tabel tersebut tidak perlu di-dekomposisi lagi.

e. Bentuk Normal Tahap Keempat dan Kelima

Penerapan aturan normalisasi sampai bentuk normal ketiga sudah memadai untuk menghasilkan tabel berkualitas baik. Namun demikian, terdapat pula bentuk normal keempat (4NF) dan kelima (5NF). Bentuk Normal keempat berkaitan dengan sifat ketergantungan banyak nilai (*multivalued dependency*) pada suatu tabel yang merupakan pengembangan dari ketergantungan fungsional. Adapun bentuk normal tahap kelima merupakan nama lain dari *Project Join Normal Form* (PJNF).

f. Boyce Code Normal Form (BCNF)

- Memenuhi 1st NF
- Relasi harus bergantung fungsi pada atribut superkey (Kusrini, M.Kom ; 2010 : 41-43).

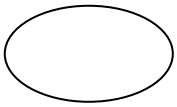
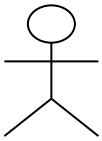
II.6. *Unified Modeling Language* (UML)

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 industry perangkat lunak dan pengembangan sistem. Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

- *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. (Windu Gata ; 2013 : 4).

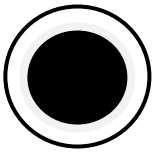
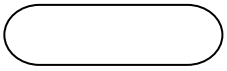
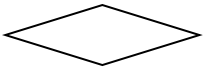
	*Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit – unit yang bertukar pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case.*
	*Actor* atau aktor adalah *abstraction* dari orang atau sistem yang lain yang mengaktifkan fungsi 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 kontrol terhadap *use case*.

Gambar II.1. Diagram *Use Case*.

(Sumber : Windu Gata; 2013 : 4)

- Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. (Windu Gata ; 2013 : 6).

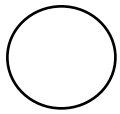
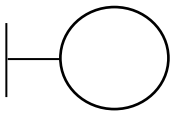
	*Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point*, akhir aktifitas.
	*Activities*, menggambarkan suatu proses / kegiatan bisnis.
	*Decision point*, menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false*.

Gambar II.2. Diagram Aktivitas.

(Sumber : Windu Gata; 2013 : 6)

- Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada usecase dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. (Windu Gata ; 2013 : 7).

	*Entitas 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.

Gambar II.3. Sequence Diagram.

(Sumber : Windu Gata; 2013 : 7)

- *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 tanggungjawab 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. (Windu Gata ; 2013 : 8).

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 maksimal 4

Gambar II.4. *Class Diagram*.

(Sumber : Windu Gata; 2013 : 9)

II.7. Mengenal Visual Basic

Visual Basic dibuat oleh microsoft, merupakan salah satu bahasa pemrograman berorientasi objek yang mudah dipelajari. Selain menawarkan kemudahan, Visual Basic juga cukup andal untuk digunakan dalam pembuatan berbagai aplikasi, terutama aplikasi database. Visual basic merupakan bahasa pemrograman event drive, dimana program akan menunggu sampai ada respons dari user/pemakai program aplikasi yang dapat berupa kejadian atau event, misalnya ketika user mengklik tombol atau menekan enter. Jika kita membuat aplikasi dengan visual basic maka kita akan mendapatkan file yang menyusun aplikasi tersebut, yaitu :

1. File Project (*.vbp)

   File ini merupakan kumpulan dari aplikasi yang kita buat. File project bisa berupa file *.frm, *.dsr atau file lainnya.

2. File Form (*.frm)

File ini merupakan file yang berfungsi untuk menyimpan informasi tentang bentuk form maupun interface yang kita buat, (Edy Winarto ; 2010 : 1).

II.8. SQL Server

SQL *Server* 2008 adalah sebuah RDBMS (*Relational Database Management System*) yang di *develop* oleh *Microsoft*, yang digunakan untuk menyimpan dan mengolah data. Pada SQL *Server* 2008, kita bisa melakukan pengambilan dan modifikasi data yang ada dengan cepat dan efisien. Pada SQL *Server* 2008, kita bisa membuat *object – object* yang sering digunakan pada aplikasi bisnis, seperti membuat *database*, *table*, *fuction*, *stored procedure*, *trigger* dan *view*. Selain *object*, kita juga menjalankan perintah SQL (*Structured Query Language*) untuk mengambil data. (Cybertron Solution ; 2010 : 101).

II.9. Pengertian Database

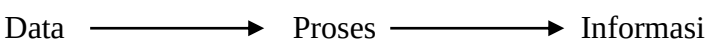
Banyak sekali definisi tentang database yang diberikan oleh para pakar dibidang ini. Database terdiri dari dua penggalan kata yaitu data dan base, yang artinya berbasiskan pada data. Tetapi secara konseptual, database diartikan sebuah koleksi atau kumpulan data yang saling berhubungan (relation), disusun menurut aturan tertentu secara logis, sehingga menghasilkan informasi. Sebuah informasi yang berdiri sendiri tidaklah dikatakan database.

Contoh : Nomor telepon seorang pelanggan, disimpan dalam banyak tempat apakah itu di file pelanggan, di file alamat dan di lokasi yang lain. Antara file yang satu dengan file yang lainnya tidak saling berhubungan, sehingga apabila salah seorang pelanggan berganti nomor telepon dan anda hanya mengganti di file pelanggan saja, akibatnya akan terjadi ketidakcocokan data, karena di lokasi yang lain masih tersimpan data telepon yang lama.

Dalam sistem database hal ini tidak boleh dan tidak bisa terjadi, karena antara file yang satu dengan file yang lain saling berhubungan. Jika suatu data yang sama anda ubah, data tersebut di file yang lain akan otomatis berubah juga. Sehingga mampu menjadi informasi yang diinginkan dan dapat dilakukan proses pengambilan, penghapusan, pengeditan, terhadap data secara mudah dan cepat (Efektif, Efisien dan Akurat).

Data adalah fakta, baik berupa sebuah objek, orang dan lain – lain yang dapat dinyatakan dengan suatu nilai tertentu (angka, simbol, karakter tertentu, dan lain – lain). Sedangkan informasi adalah data yang telah diolah sehingga bernilai guna dan dapat dijadikan bahan dalam pengambilan keputusan, (Yuhefizard ; 2010 : 2).

Hubungan data dan informasi dapat digambarkan sebagai berikut :



Gambar II.5. Data dan Informasi.

(Sumber : Yuhefizard ; 2010 : 2)