# BAB II

# TINJAUAN PUSTAKA

## II.1. Pengertian Sistem

Sistem merupakan sekumpulan elemen-elemen yang saling terintegrasi serta melaksanakan fungsinya masing-masing untuk mencapai tujuan yang telah ditetapkan. Karakteristik sistem terdiri dari :

1. Komponen Sistem

   Suatu sistem terdiri dari sejumlah komponen yang saling berinteraksi, yang artinya saling bekerja sama membentuk suatu kesatuan. Komponen-komponen sistem atau elemen-elemen sistem dapat berupa suatu subsistem atau bagian-bagian dari sistem.

2. Batasan Sistem

   Batasan merupakan daerah yang membatasi antara suatu sistem dengan sistem yang lainnya atau dengan lingkungan luarnya. Batasan sistem ini memungkinkan suatu sistem dipandang suatu kesatuan. Batasan suatu sistem menunjukkan ruang lingkup (*scope*) dari sistem tersebut.

3. Lingkungan Luar Sistem

   Lingkungan luar dari suatu sistem adalah apapun diluar batas dari sistem yang mempengaruhi operasi sistem. Lingkung luar sistem dapat bersifat menguntungkan dan dapat juga bersifat merugikan sistem tersebut.

4. Penghubung Sistem

   Penghubung merupakan media penghubung antara satu subsistem dengan subsistem lainnya. Melalui penghubung ini memungkinkan sumber-sumber daya mengalir dari satu subsistem ke subsistem lainnya.

5. Masukan Sistem

   Masukan sistem adalah energi yang dimasukkan ke dalam sistem. Masukan dapat berupa masukan perawatan (*maintance input*) dan masukan sinyal (*signal input*).

6. Keluaran Sistem

   Keluaran sistem adalah hasil energi yang diolah dan diklasifikasikan menjadi keluaran yang berguna dan sisa pembuangan.

7. Pengolah Sistem

   Suatu sistem dapat mempunyai suatu bagian pengolah atau sistem itu sendiri sebagai pengolahnya. Pengolah akan mengubah masukan menjadi keluaran.

8. Sasaran Sistem

Suatu sistem mempunyai tujuan (*goal*) atau sasaran (*objective*). Kalau suatu sistem tidak mempunyai sasaran, maka operasi sistem tidak ada gunanya (Sulindawati, 2010 : 135).

## II.2. 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.3. 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 adecisionwiththeuncertaindata*),

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 (*generalpractitioner*) dan dokter keluarga (*familyphysicians*). 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.4. Pengenalan Hipoglikemia Dan Hiperglikemia

Hipoglikemia merupakan gambaran umum dari diabetes melitus tipe I dan dapat pula timbul pada klien dengan tipe II yang diobati dengan insulin atau preparat oral. Kadar glukosa darah yang tepat dimana klien mengalami hipoglikemia bervariasi, tetapi manifestasi ini umumnya tidak terjadi sampai

kadar glukosa darah kurang dari 50 sampai 60 mg/dl. Hipoglikemia umumnya ditandai oleh pucat, piloereksi, gelisah, lemah, lapar, berkeringat dingin, mata berkunang – kunang, kusut pikir, semutan sekitar mulut, sakit kepala dan akhirnya koma. Reaksi hipoglikemia terjadi akibat :

1. Tidak makan atau makan lebih sedikit dari biasanya.
2. Aktivitas fisik berlebihan tanpa adanya tambahan karbohidrat.
3. Ketidakseimbangan nutrisional dan cairan akibat muntah dan mual.
4. Konsumsi alkohol.

Kesalahan atau ketidaktepatan dosis insulin sering menyebabkan hipoglikemia. Perubahan jadwal pemberian insulin atau pemberian makan, aktivitas fisik berat yang tidak diperkirakan, atau tidur lebih larut dari biasanya pada pagi hari juga dapat menyebabkan hipoglikemia juga dapat terjadi sekunder terhadap pemberian preparat hipoglikemik oral. Penanggulangan segera bila terjadi hipoglikemia adalah makanan yang mengandung gula atau permen, minuman manis yang dibuat dengan cara melarutkan satu sendok makan gula pasir ke dalam satu gelas air. Ingatkan penderita diabetik untuk selalu membawa makanan atau gula bila sedang berpergian.

Hiperglikemia terjadi ketika glukosa tidak dapat dikirimkan ke sel – sel karena kurangnya insulin. Tanpa adanya karbohidrat untuk bahan bakar selular, hepar akan mengubah simpanan glikogennya menjadi glukosa (glikogenolisis) dan meningkatkan biosintetis glukosa (glukoneogenesis). Namun begitu, respons ini makin memperburuk situasi dengan meningkatkan kadar glukosa darah semakin tinggi. Hiperglikemia merupakan komplikasi dari diabetes tipe I,

meskipun klien dengan juga dapat mengalaminya selama periode stress yang ekstrim. Penyebab hiperglikemia imimnya mencakup :

1. Menggunakan terlalu sedikit insulin.
2. Tidak menggunakan insulin sama sekali.
3. Kegagalan untuk memenuhi kebutuhan insulin yang meningkat akibat operasi, trauma, kehamilan, stress, pubertas, atau infeksi.
4. Kurang aktivitas fisik.
5. Membentuk resisten insulin sebagai akibat adanya antibodi insulin.

Bila klien mengalami hiperglikemia, bawa segera ke tempat pelayanan kesehatan terdekat sambil tetap meningkatkan asupan cairan bila kesadaran klien masih baik. Karena komplikasi ini dapat terjadi dalam waktu singkat serta mengancam hidup klien, hendaknya klien dilengkapi dengan kartu identitas yang menyebutkan sebagai penderita diabetes sehingga bila terjadi suatu keadaan darurat pemberi pertolongan dapat melakukan tindakan yang tepat dan cepat. (Hotma Rumahorbo, Skp; 2010:112).

## II.5. Diabetes

Diabetes melitus (DM) adalah kelainan metabolisme karbohidrat, dimana glukosa darah tidak dapat digunakan dengan baik, sehingga menyebabkan keadaan hiperglikemia. Dengan kata lain, Diabetes Melitus adalah penyakit yang ditandai oleh kadar gula darah yang tinggi melebihi batas-batas normal. (Anik Maryunani; 2013:9).

## II.6. Metode *Certainty Factor*

*CertaintyTheory* ini diusulkan oleh Shortliffe dan Buchanan pada tahun 1975 untuk mengakomodasi ketidakpastian pemikiran (*inexactreasoning*) seorang pakar. Teori ini berkembang bersamaan dengan pembuatan sistem pakar MYCIN. *Team* pengembang MYCIN mencatat bahwa tim ahli sering kali menganalisa informasi yang ada dengan ungkapan seperti misalnya : mungkin, kemungkinan besar, hampir pasti. Untuk mengakomodasi hal ini tim MYCIN menggunakan *certaintyfactor* (CF) guna menggambarkan tingkat keyakinan pakar terhadap masalah yang sedang dihadapi. Secara umum, *rule* direpresentasikan dalam bentuk sebagai berikut :

IF E1 [AND/OR] E2 [AND/OR] . . . $E_n$

THEN H (CF = CF) . . . . . (1). (Jurnal Teknik Informatika, Bain Khusnul Khotimah; 2010:13).

*CeratintyFactor*merupakan suatu metode yang digunakan untuk menyatakan kepercayaan dalam sebuah kejadian (fakta atau hipotesis) berdasarkan bukti atau penilaian pakar. Secara konsep, *certaintyfactor* (CF) merupkan salah satu teknik yang digunakan untuk mengatasi ketidakpastian dalam pengambilan keputusan. *CertaintyFactor* (CF) dapat terjadi dengan berbagai kondisi. Diantara kondisi yang terjadi adalah terdapat beberapa *antesenden* (dalam *rule* yang berbeda) dengan satu konsekuen yang sama. Dalam kasus ini, kita harus mengagregasikan nilai CF keseluruhan dari setiap kondisi yang ada *belive* dan *disbelive*. *Belive* merupakan keyakinan, sedangkan *disbelive* merupakan

ketidakyakinan. Adapun notasi atau rumusan dasar dari *certaintyfactor*, sebagai berikut.

CF [h,e] = MB [h,e] – MD [h,e]

Keterangan :

CF [h,e] = *CertaintyFactor*dalam hipotesis h yang dipengaruhi oleh fakta e.

MB [h,e] = *MeansureofBelive*, merupakan nilai kenaikan dari kepercayaan hipotesis h dipengaruhi oleh fakta e.

MD [h,e] = *MeasureofDisbelive*, merupakan nilai kenaikan dari ketidakpercayaan hipotesis h dipengaruhi oleh fakta e.

h = Hipotesis

e = *Evidence*. (Jurnal Ilmiah, Putu Ary Darma Yasa; 2012:2).

## II.7. 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.7.1. 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.8. *Unified Modeling Language* (UML)

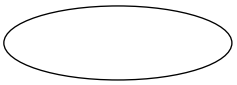
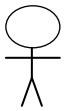
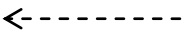
UML adalah bahasa spesifikasi standar yang dipergunakan untuk mendokumentasikan, menspesifikasikan dan membangun perangkat lunak.UML merupakan metodologi dalam mengembangkan sistem berorientasi objek dan juga merupakan alat untuk mendukung pengembangan sistem.UML saat ini sangat banyak dipergunakan dalam 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 :

- *UseCase*Diagram

*Usecase*diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Usecase*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).

**Tabel II.1. Diagram*UseCase*.**

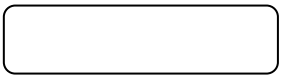
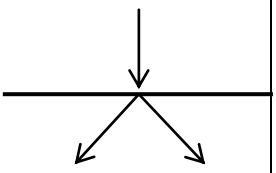
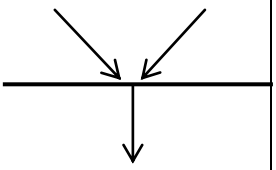
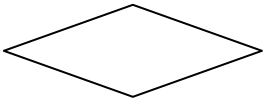
| Gambar | Keterangan |
|---|---|
|  | *Use case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukan pesan antar unit dengan aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal nama *use case*. |
|  | Aktor adalah *abstraction* dari orang atau sistem yang 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 : 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).

**Tabel II.2. Diagram Aktivitas.**

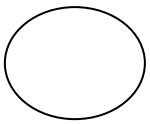
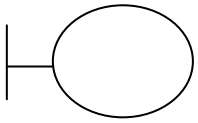
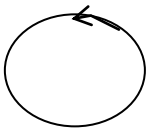
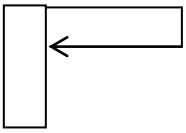
| Gambar | Keterangan |
|---|---|
|  | *Start point*, diletakkan pada pojok kiri atas dan merupakan awal aktifitas. |
|  | *End point*, akhir aktifitas. |
|  | *Activites*, menggambarkan suatu proses/kegiatan bisnis. |
|  | *Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara 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 : 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).

**Table II.3. Diagram Urutan**

| Gambar | Keterangan |
|---|---|
|  | *EntityClass,* 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 : 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).

**Table II.4. 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 maksimal 4 |

**(Sumber : Windu Gata; 2013 : 9)**

## II.9. 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.10. 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*, *storedprocedure*, *trigger* dan *view*. Selain *object*, kita juga menjalankan perintah SQL (*Structured Query Language*) untuk mengambil data. (Cybertron Solution ; 2010 : 101).

## II.11. 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

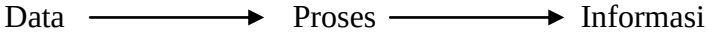
aturan tertentu secara logis, sehingga menghasilkan informasi. Sebuah informasi yang berdiri sendiri tidaklah dikatakan database.

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.1. Data dan Informasi.**

**(Sumber : Yuhefizard ; 2010 : 2)**