BAB II

TINJAUAN PUSTAKA

II.1. Sistem Pakar

Pengetahuan yang di muat ke dalam sistem pakar dapat berasal dari seorang pakar ataupun pengetahuan yang berasal dari buku, jurnal, majalah dan dokumentasi yang dipublikasikan lainnya serta orang yang memiliki pengetahuan meskipun bukan ahli. Istilah sistem pakar sering disinonimkan dengan sistem berbasis pengetahuan (*knowledge-based* system) atau sistem pakar berbasis pengetahuan (*knowledge based expert* system). Sebuah sistem pakar umumnya tidak memiliki pengetahuan lain diluar area pengetahuannya kecuali jika diprogram dan dimuat ke dalam sistem (Rika Rosnelly,2012;3-4).

Sistem Pakar pada umumnya dirancang untuk memenuhi beberapa karakteristik umum berikut ini :

1. Kinerja sangat baik (*high performance*). Sistem harus mampu memberikan respon berupa saran (*advice*) dengan tingkat kualitas yang sama dengan seorang pakar atau melebihinya.
2. Waktu respon yang baik (*adequate respon time*). Sistem juga harus mampu bekerja dalam waktu yang sama baiknya (*reasonable*) atau lebih cepat dibandingkan dengan seorang pakar dalam menghasilkan keputusan.
3. Dapat diandalkan (*good reliability*). Sistem harus dapat diandalkan dan tidak mudah rusak/ crash.

4. Dapat dipahami (*understandable*). Sistem harus mampu menjelaskan lngkah-langkah penalaran yang dilakukannya seperti seorang pakar.
5. Fleksibel (*fleksibility*). Sistem harus menyediakan mekanisme untuk menambah, mengubah, dan menghapus pengetahuan (Rika Rosnelly, 2012;20-21).

II.1.1. Ciri-ciri Sistem Pakar

Sistem Pakar memiliki beberapa ciri-ciri, diantanya sebagai berikut :

1. Terbatas pada domain keahlian tertentu.
2. Dapat memberikan penalaran untuk data-data yang tidak lengkap atau tidak pasti.
3. Dapat menjelaskan alasan-alasan dengan cara yang dapat dipahami.
4. Bekerja berdasarkan kaidah/*rule* tertentu.
5. Mudah dimodifikasi
6. Basis pengetahuan dan mekanisme inferensi terpisah.
7. Keluarannya bersifat anjuran.
8. Sistem dapat mengaktifkan kaidah secara searah yang sesuai, dituntun oleh dialog dengan pengguna (T.Sutojo,2011;162).

II.1.2. Manfaat Sistem Pakar

Sistem Pakar menjadi sangat populer karena sangat banyak kemampuan dan manfaat yang diberikan, diantaranya :

1. Meningkatkan produktivitas, karena Sistem Pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awan bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberikan nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Memudahkan akses pengetahuan seorang pakar.
6. Meningkatkan kapabilitas sistem komputer. Integritas Sistem Pakar dengan sistem komputer lain membuat sistem lebih efektif dan mencangkup lebih banyak sistem.
7. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti. Berbeda dengan sistem komputer konvensional, Sistem Pakar dapat bekerja dengan informasi yang tidak lengkap.
8. Bisa digunakan sebagai media pelengkap dalam pelatihan. Pengguna pemula bekerja dengan Sistem Pakar akan menjadi lebih berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.
9. Meningkatkan kemampuan untuk menyelesaikan masalah karena Sistem Pakar mengambil sumber pengetahuan dari banyak pakar (T.Sutojo, 2011;160-161).

II.1.3. Kekurangan Sistem Pakar

Selain manfaat sistem pakar juga memiliki beberapa kelemahan, diantaranya :

1. Memerlukan biaya yang sangat mahal untuk membuat dan memelihatanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
3. Sistem Pakar tidak selamanya 100% bernilai benar (T.Sutojo,2011;161).

II.1.4. Konsep Dasar Sistem Pakar

Konsep dasar sistem pakar meliputi enam hal berikut ini :

1. Kepakaran (*Expertise*)

Kepakaran merupakan suatu pengetahuan yang diperoleh dari pelatihan, membaca dan pengalaman. Kepakaran inilah yang memungkinkan para ahli dapat mengambil keputusan lebih cepat dan lebih baik daripada seseorang yang bukan pakar.

2. Pakar (*Expert*)

Pakar adalah seseorang yang mempunyai pengetahuan, pengalaman dan metode khusus serta mampu menerapkannya untuk memecahkan masalah atau memberi nasehat. Seorang pakar harus mampu menjelaskan dan mempelajari hal-hal baru yang berkaitan dengan topik permasalahan, jika perlu harus mampu menyusun kembali pengetahuan-pengetahuan yang didapatkan dan dapat memecahkan aturan-aturan serta menentukan relevansi kepakarannya.

3. Pemindahan Kepakaran (*Transfering Expertisi*)

Tujuan dari Sistem Pakar adalah memindahkan kepakaran dari seorang pakar ke dalam komputer, kemudian kepada orang lain yang bukan pakar. Proses ini melibatkan empat kegiatan, yaitu:

a. Akuisisi pengetahuan (dari pakar atau sumber lain).

b. Representase pengetahuan (pada komputer).

c. Inferensi pengetahuan.

d. Pemindahan pengetahuan ke pengguna.

4. Inferensi (*Inferencing*)

Inferensi adalah sebuah prosedur (program) yang mempunyai kemampuan dalam melakukan penalaran. Inferensi ditampilkan pada suatu komponen yang disebut mesin inferensi yang mencangkup prosedur-prosedur mengenai pemecahan masalah. Semua pengetahuan yang dimiliki oleh seorang pakar disimpan pada basis pengetahuan oleh sistem pakar. Tugas mesin inferensi adalah mengambil kesimpulan berdasarkan basis pengetahuan yang dimilikinya.

5. Aturan-aturan (*Rules*)

Kebanyakan software pakar komersil adalah sistem yang berbasis *rule* (*rule-based system*), yaitu pengetahuan disimpan terutama dalam bentuk *rule,* sebagai prosedure pemecahan masalah.

6. Kemampuan Menjelaskan (*Explanation Capability*)

Fasilitas lain dari sistem pakar adalah kemampuannya untuk menjelaskan saran atau rekomendasi yang diberikannya. Penjelasan dilakukan dalam subsistem yang disebut subsistem penjelasan (*explanation*). Bagian dari sistem ini

memungkinkan sistem untuk memeriksa penalaran yang dibuatnya sendiri dan menjelaskan operasi-operasinya (T.Sutojo,2011;163-165).

II.2. Penyakit Polip

II.2.1. Defenisi Penyakit Polip

Polip hidung adalah kelainan selaput permukaan hidung berupa massa lunak yang bertangkai, berbentuk bulat atau lonjong, berwarna putih keabu-abuan dengan permukaan licin dan agak bening karena mengandung banyak cairan. Kelainan pada hidung biasanya timbul karena manifestasi dari penyakit yang lain dan tidak berdiri sendiri. Penyakit ini sering dihubungkan dengan asthma, rhinitis alergika dan sinusitis.

Etiologi secara pasti hingga sekarang belum diketahui, tetapi terdapat 3 faktor penting yang berperan didalam terjadinya polip, yaitu :

1. Peradangan lama dan berulang pada selaput permukaan hidungdan sinus.
2. Gangguan keseimbangan Vasomotor.
3. Peningkatan tekanan cairan antar ruang sel dan bengkak selaput permukaan hidung.

Fenomena Bernouli menyatakan bahwa udara yang mengalir melalui celah yang sempit akan mengakibatkan tekanan negatif pada daerah sekitarnya, sehingga jaringan yang lemah akan terhisap oleh tekanan negatif ini sehingga menyebabkan polip. Fenomena ini dapat menjelaskan mengapa polip banyak terjadi pada area yang sempit di kompleks osteomaral.

Patofisiologi polip pada awalnya ditemukan banyak selaput permukaan yang kebanyakan terdapat pada meatus medius, kemudian stroma akan terisi oleh cairan interseluler sehingga selaput permukaan yang sembab menjadi berbenjol-benjol. Bila proses terus membesar dan kemudian turun ke dalam rongga hidung sambil membentuk tangkai sehingga terjadi polip (Elisabeth Ari,2010;149).

II.2.2. Gejala Penyakit Polip

Gejala-gejala dari penyakit polip adalah sebagai berikut :

1. Hidung tersumbat, sumbatan ini menetap dan tidak hilang timbul. Semakin lama keluhan dirasakan semakin berat.
2. Pasien sering mengeluhkan terasa ada benjolan di dalam hidung dan sukar membuang ingus.
3. Gangguan penciuman.
4. Dapat timbul jika terdapat kelainan di organ sekitarnya seperti post nasal drip (cairan yang mengalir di bagian belakang mulut) (Elisabeth Ari,2010;150).

II.3. Metode *Dempster Shafer*

Ada berbagai macam penalaran dengan model yang lengkap dan sangat konsisten, tetapi pada kenyataannya banyak permasalahan yang tidak dapat terselesaikan secara lengkap dan konsisten. Ketidak konsistenan yang tersebut adalah akibat adanya penambahan fakta baru. Penalaran yang seperti itu disebut dengan penalaran *non monotonis*. Untuk mengatasi ketidak konsistenan tersebut maka dapat menggunakan penalaran dengan teori *Dempster-Shafer*. *Dempster-*

Shafer adalah suatu teori matematika untuk pembuktian berdasarkan *belief functions and plausible reasoning*(fungsi kepercayaan dan pemikiran yang masuk akal), yang digunakan untuk mengkombinasikan potongan informasi yang terpisah (bukti) untuk mengkalkulasi kemungkinan dari suatu peristiwa. Teori ini dikembangkan oleh Arthur P. Dempster dan Glenn Shafer. Secara umum teori *Dempster-Shafer* ditulis dalam suatu interval :

1. [*Belief,Plausibility*]

Belief (*Bel*) adalah ukuran kekuatan evidence dalam mendukung suatu himpunan proposisi. Jika bernilai 0 maka mengindikasikan bahwa tidak ada evidence, dan jika bernilai 1 menunjukkan adanya kepastian. Dimana nilai bel yaitu (0-0.9).

2. *Plausibility* (Pl) dinotasikan sebagai :

Pl(s) = 1 – Bel (-s)

Plausibility juga bernilai 0 sampai 1. Jika yakin akan -s, maka dapat dikatakan bahwa Bel(-s)=1, dan Pl(-s)=0. Pada teori *Dempster-Shafer* dikenal adanya *frame of discrement* yang dinotasikan dengan θ. Frame ini merupakan semesta pembicaraan dari sekumpulan hipotesis. Tujuannya adalah mengaitkan ukuran kepercayaan elemen-elemen θ. Tidak semua evidence secara langsung mendukung tiap-tiap elemen. Untuk itu perlu adanya probabilitas fungsi densitas (m). Nilai m tidak hanya mendefinisikan elemen-elemen θ saja, namun juga semua subsetnya. Sehingga jika θ berisi n elemen, maka subset θ adalah 2^n . Jumlah semua m dalam subset θ sama dengan 1. Apabila tidak ada informasi apapun untuk memilih hipotesis, maka nilai : m{θ} = 1,0.

Apabila diketahui X adalah subset dari θ, dengan m1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari θ dengan m2 sebagai fungsi densitasnya, maka dapat dibentuk fungsi kombinasi m1 dan m2 sebagai m3, yaitu:

$$m3(Z) = \left(\frac{\sum x \cap y = z \; m1(X).m2(Y)}{1 - \sum x \cap y = \theta m1(X).m2(Y)} \right)$$

Dimana :

m3(Z) = *mass function* dari *evidence* (Z)

m1(X) = *mass function* dari *evidence* (X), yang diperoleh dari nilai keyakinan suatu *evidence* dikalikan dengan nilai *disbelief* dari *evidence* tersebut.

m2(Y) = *mass function* dari *evidence* (Y), yang diperoleh dari nilai keyakinan suatu *evidence* dikalikan dengan nilai *disbelief* dari *evidence* tersebut.

$\sum x \cap y = \theta m1(X).m2(Y)$ = merupakan nilai kekuatan dari *evidence* Z yang diperoleh dari kombinasi nilai keyakinan sekumpulan *evidence* (Jurnal Aprilia Sulistyohati,2008;2).

II.4. Unified Modeling Language (UML)

II.4.1. Defenisi Unified Modeling Language (UML)

Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak. UML menawarkan sebuah standar untuk merancang model sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk semua jenis aplikasi piranti lunak, dimana aplikasi tersebut

dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun. Tetapi karena UML juga menggunakan bahasa *class* dan *operation* dalam konsep dasarnya, maka ia lebih cocok untuk penulisan piranti lunak dalam bahasa-bahasa berorientasi objek seperti C++, Java, C# atau VB.NET. Walaupun demikian UML tetap dapat digunakan untuk modeling aplikasi prosedural dalam VB atau C (Yuni Sugiarti,2013;34).

UML biasa digunakan untuk :

1. Menggambarkan batasan sistem dan fungsi-fungsi sistem secara umum, dibuat dengan *use case* dan *actor.*
2. Menggambarkan kegiatan atau proses bisnis yang dilakukan secara umum, dibuat dengan *interactions diagrams.*
3. Menggambarkan representasi struktur statik sebuah sistem dalam bentuk *class diagrams*.
4. Membuat model *behavior* "yang menggambarkan kebiasaan atau sifat sebuah sistem" dengan *state transition diagrams*.
5. Menyatakan arsitektur implementasi fisik menggunakan *component* dan *development diagrams*.
6. Menyampaikan atau memperluas *fungsionality* dan *stereotypes* (Yuni Sugiarti,2013;36).

II.4.2. Diagram – Diagram Pada UML

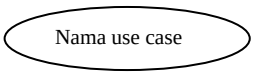
Beberapa diagram yang terdapat pada *UML* diantaranya adalah :

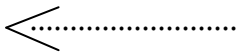
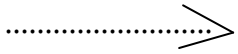
1. *Use Case Diagram*

Use Case Diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. *Diagram Use Case* mendeskripsikan sebuah interaksi antara satu atau lebih *actor* dengan sistem yang akan dibuat. *Diagram Use Case* digunakan untk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Hal yang perlu diingat mengenai *diagram use case* adalah *diagram use case* bukan menggambarkan tampilan antarmuka *user*, arsitektur dari sistem, kebutuhan nonfungsional dan tujuan performansi (Yuni Sugiarti,2013;41).

Berikut ini adalah simbol-simbol yang ada pada *use case diagram* :

Tabel II.1. Simbol – Simbol Pada Use Case Diagram

Simbol	Deskripsi
Use case 	Fungsionalitas yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor, biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama *use case*
Actor  nama actor	Orang, proses atau sistem yang lain berinteraksi dengan sistem informasi yang akan dibuat diluar sistem informasi yang akan di buat itu sendiri
Asosiasi / *association*	Komunitas antara actor dan *use case* yang berpartisipasi pada *use case*, atau *use case* memiliki interasi dengan aktor

Extend  <<extend>>	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* yang ditambahkan dapat berdiri sendiri walau tanpa *use case* tambahan itu, mirip dengan prinsip *inheritance* pada pemrograman berorientasi objek, biasanya *use case* tambahan memiliki nama depan yang sama dengan *use case* yang ditambahkan, arah panah mengarah pada *use case* yang dituju.
Include <<include>> 	Relasi *use case* tambahan ke sebuah *use case* dimana *use case* ditambahkan memerlukan *use case* ini untuk menjalankan fungsinya atau sebagai syarat dijalankan *use case* ini. Ada sudut pandang yang cukup besar mengenai *include* di *use case*, *include* berarti *use case* yang ditambahkan akan selalu dipanggil saat *use case* tambahan dijalankan.

Sumber : (Yuni Sugiarti,2013;42)

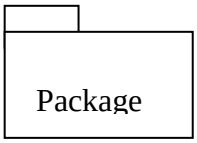
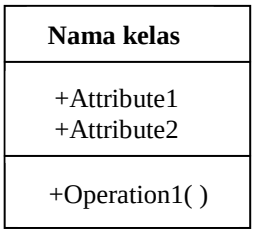
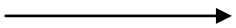
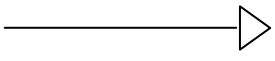
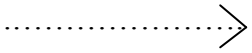
2. *Class Diagram* (Diagram Kelas)

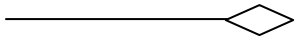
Class Diagram menggambarkan struktur sistem dari segi pendefenisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Atribut mendeskripsikan properti dengan sebaris teks didalam kotak kelas tersebut. Operasi atau metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Diagram kelas mendeskripsikan jenis-jenis objek dalam

sistem dan berbagai hubungan statis yang terdapat diantara mereka (Yuni Sugiarti,2013;57).

Berikut ini adalah simbol-simbol yang ada pada diagram kelas :

Tabel II.2. Simbol - Simbol Pada Diagram Kelas

Simbol	Deskripsi
Package 	Package merupakan sebuah bungkusan dari sau atau lebih kelas
Aktivitas 	Kelas pada struktur sistem
Antarmuka / *Interface* 	Sama dengan konsep *interface* dalam pemrograman berorientasi objek
Asosiasi	Relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan *multiplicity*
Asosiasi berarah / *Directed* asosiasi 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antar kelas dengan makna generalisasi-spesialisasi (umumkhusus)
Kebergantungan / *defedency* 	Relasi antar kelas dengan makna kebergantungan antar kelas

Agresasi 	Relasi antar kelas dengan makna semua bagian (whole-part)

Sumber : (Yuni Sugiarti,2013;59)

3. Sequence Diagram

Diagram sekuence menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Menggambar diagram sekuences maka harus diketahui objek-objek yang terlibat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuence yang harus digambar adalah sebanyak pendefenisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefenisikan interaksi jalannya pesan sudah cukup pada diagram sekuence sehingga semakin banyak *use case* yang didefenisikan maka diagram sekuence yang dibuat juga semakin banyak (Yuni Sugiarti,2013;69).

4. *Activity Diagram* (Aktivitas)

Diagram aktivitas atau *activity diagram* menggambarkan aliran kerja atau aktivitas dari sebuah sistem atau proses bisnis. Diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem.

Diagram aktivitas juga banyak digunakan untuk mendefenisikan hal-hal berikut :

a. Rancangan proses bisnis dimana setiap urusan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefenisikan.
b. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas dianggap memiliki sebuah rancangan antarmuka tampilan.
c. Rancangan pengujian dimana setiap aktivitas dianggap memerlukan sebuah pengujian yang perlu didefenisikan kasus ujiannya.

Activity diagram merupakan state diagram khusus, dimana sebagian besar state adalah *action* dan sebagian besar transisi ditrigger oleh selesainya state sebelumnya. Oleh karena itu, *activity diagram* tidak menggambarkan *behavior* internal sebuah sistem secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum (Yuni Sugiarti,2013;75).

II.5. ERD (*Entity Relationship Diagram*)

Pemodelan basis data dengan menggunakan diagram relasi antar entitas dapat dilakukan dengan menggunakan suatu pemodelan basis data yang bernama ERD (*Entity Relationship Diagram*) (Yudi Priadi,2014;20).

Adapaun simbol-simbol ERD (*Entity Relationship Diagram*) adalah sebagai berikut :

Tabel II.3. Simbol – Simbol Pada ERD

Simbol	Deskripsi
Entitas/ *entity* Nama_entitas	Entitas merupakan notasi untuk mewakili suatu objek dengan karakteristik sama, yang dilengkapi oleh atribut, sehingga pada suatu lingkungan yang nyata setiap objek akan berbeda dengan objek lainnya.
Relasi Nama_relasi	Relasi merupakan notasi yang digunakan untuk menghubungkan beberapa entitas berdasarkan fakta pada suatu lingkungan
Atribut Nama_atribut	Atribut merupakan notasi yang menjelaskan karakteristik suatu entitas dan juga relasinya
Garis Penghubung _____	Garis penghubung merupakan notasi untuk merangkaian keterkaitan antara notasi-notasi yang digunakan dalam Diagram E-R, yaitu entitas, relai dan atribut.

Sumber : (Yudi Priadi,2014;21)

II.6. Normalisasi Data

Ketergantungan secara fungsional adalah hubungan yang menjelaskan ketergantungan antar atribut dalam suatu tabel. Konsep inilah yang menjadi dasar dalam proses normalisasi. Kegiatan ini adalah proses sistematis yang dilakukan mulai dari bentuk tidak normal menjadi bentuk normal, melalui suatu syarat yang harus dipenuhi pada saat menuju suatu bentuk yang lebih baik (Yudi Priadi,2014;59).

Normalisasi merupakan tahapan yang harus dilakukan pada struktur tabel basis data menjadi struktur tabel yang memiliki integritas data, sehingga tidak memiliki data anomali pada saat melakukan *insert, delete,* dan *update* (Yudi Priadi,2014;67).

II.7. Visual Basic 2010

Visual Basic.NET merupakan bahasa pemograman yang memberikan berbagai fasilitas pembuatan aplikasi visual. Keunggulan bahasa pemrograman ini terletak pada produktivitas, kualitas, pengembangan perangkat lunak, kecepatan kompilasi, pola desain yang menarik dan pemrograman terstruktur. Keunggulan lein dari *Visual Basic.NET* adalah dapat digunakan untuk merancang program aplikasi yang memiliki tampilan seperti program aplikasi yang lain yang berbasis *Windows* (Aghus Sofwan,2013;2).

Visual Basic 2010 merupakan salah satu bagian dari produk pemrograman terbaru yang dikeluarkan oleh *Microsoft,* yaitu *Microsoft Visual Studio* 2010. Sebagai produk lingkungan pengembangan integrasi atau IDE andalan yang

dikeluarkan *Microsoft, Visual Studio* 2010 menambahkan perbaikan-perbaikan vitur dan fitur yang lebih lengkap dibandingkan versi *Visual Studio* pendahulunya, yaitu *Microsoft Visual Studio* 2008 (Wahana Komputer,2010;2).

II.8. Basis Data

Suatu basis data terdiri dari kumpulan tabel yang saling berelasi ataupun tidek berelasi. Semua tabel tersebut merupakan representasi tempat untuk penyimpanan data yang mendukung fungsi dari basis data tersebut pada suatu sistem.Basis data difokuskan menjadi beberapa elemen, yaitu tabel, *field key (primary dan foreign), field non key, record* dan kardinalitas.

Basis data adalah sekumpulan fakta berupa representasi tabel yang saling berhubungan dan disimpan dalam media penyimpanan secara digital. Basis data terdiri dari sekumpulan tabel yang saling berelasi ataupun tidak berelasi (Yudi Priadi,2014;1-2).

II.9. SQL Server 2008

SQL Server 2008 adalah sebuah terobosan baru dari *Microsoft* dalam bidang database. *SQL Server* adalah DBMS (*Database Management System*) yang dibuat oleh *Microsoft* untuk ikut berkecimpung dalam persaingan dunia pengolahan data menyusul pendahulunya seperti IBM dan *Oracle. SQL Server* 2008 dibuat pada saat kemajuan dalam bidang *hardware* sedemikian pesat. Oleh karena itu sudah dapat dipastikan bahwa *SQL Server* 2008 membawa beberapa

terobosan dalam bidang pengolahan dan penyimpanan data (Wenny Widya,2013;3).