BAB II

TINJAUAN PUSTAKA

II.1. Keaslian Penelitian

Keaslian penelitian ini menjadi salah satu acuan peneliti dalam melakukan penelitian sehingga peneliti dapat memperkaya teori yang digunakan dalam mengkaji penelitian yang dilakukan. Dari keaslian penelitian, peneliti tidak menemukan penelitian dengan judul yang sama seperti judul penelitian peneliti, Namun peneliti mengangkat beberapa penelitian sebagai referensi dalam memperkaya bahan kajian pada penelitian peneliti.

Pada peneliti pertama yang berjudul ***"Implementasi Case Base Reasoning Pada Sistem Pakar Dalam Menentukan Jenis Gangguan Kejiwaan"***, Penelitian tersebut membahas tentang jenis gangguan kejiwaan yang difokuskan pada gangguan kejiwaan *phobia*.

Pada peneliti kedua yang berjudul ***"Sistem Pendukung Keputusan Klinis Untuk Mengefisienkan Diagnosa Penyakit Kejiwaan Menggunakan Case Based Reasoning"***, Penelitian ini menggunakan metode yang sama dengan peneliti pertama dan membahas tentang sistem pendukung keputusan klinis yang dapat digunakan untuk melakukan diagnosa awal terhadap penyakit kejiwaan *skizofrenia*.

Pada peneliti ketiga yang berjudul ***"Sistem Pakar Diagnosis Penyakit Kejiwaan Skizofrenia Menggunakan Metode Tsukamoto"***, Penelitian tersebut mendiagnosa tipe penyakit kejiwaan *skizofrenia* yang tingkat keakuratan diperoleh dari kesesuaian antara hasil diagnosis sistem pakar dengan basis pengetahuan satu pakar.

Sedangkan pada peneliti keempat yang berjudul ***"Sistem Pakar Diagnosa Awal Gangguan Jiwa Dengan Metode Certainty Factor Berbasis Mobile Cellular"***, Penelitian ini membatasi permasalahan penyakit yang dimasukkan ke dalam sistem pakar ini hanya berupa penyakit gangguan jiwa neurosis yang terbagi dalam tujuh penyakit, yakni : Cemas, Neurasthenia, Histeria, Depresi, Fobik, Obsesif-kompulsif, dan Somatisasi.

II.2. Sistem

Menurut Asbon Hendra (2012 : 157), Sistem merupakan kumpulan dari unsur atau elemen-elemen yang saling berkaitan/berinteraksi dan saling memengaruhi dalam melakukan kegiatan bersama untuk mencapai suatu tujuan tertentu.

II.3. Pakar

Menurut T.Sutojo, dkk (2011 : 163-164), 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. Jadi seorang pakar harus mampu melakukan kegiatan-kegiatan berikut :

1. Mengenali dan memformulasikan permasalahan.
2. Memecahkan permasalahan secara cepat dan tepat.
3. Menerangkan pemecahannya.
4. Belajar dari pengalaman.
5. Merestrukturisasi pengetahuan.
6. Memecahkan aturan-aturan.
7. Menentukan relevansi.

II.4. Sistem Pakar

Menurut T.Sutojo, dkk (2011 : 13), Sistem pakar adalah suatu sistem yang dirancang untuk dapat menirukan keahlian seorang pakar dalam menjawab pertanyaan dan memecahkan suatu masalah. Sistem pakar akan memberikan pemecahan suatu masalah yang didapat dari dialog dengan pengguna. Dengan bantuan sistem pakar seseorang yang bukan pakar/ahli dapat menjawab pertanyaan, menyelesaikan masalah serta mengambil keputusan yang biasanya dilakukan oleh seorang pakar.

II.4.1. Manfaat Sistem Pakar

Menurut T.Sutojo, dkk (2011 : 160-161), Sistem pakar menjadi sangat populer karena sangat banyak kemampuan dan manfaat yang diberikannya, di antaranya :

1. Meningkatkan produktivitas, karena sistem pakar dapat bekerja lebih cepat daripada manusia.
2. Membuat seorang yang awam bekerja seperti layaknya seorang pakar.
3. Meningkatkan kualitas, dengan memberi nasehat yang konsisten dan mengurangi kesalahan.
4. Mampu menangkap pengetahuan dan kepakaran seseorang.
5. Dapat beroperasi di lingkungan yang berbahaya.
6. Memudahkan akses pengetahuan seorang pakar.
7. Andal, sistem pakar tidak pernah menjadi bosan dan kelelahan atau sakit.
8. Meningkatkan kapabilitas sistem komputer, integrasi sistem pakar dengan sistem komputer lain membuat sistem lebih efektif dan mencakup lebih banyak aplikasi.
9. Mampu bekerja dengan informasi yang tidak lengkap atau tidak pasti, berbeda dengan sistem komputer konvensial, sistem pakar dapat bekerja dengan informasi yang tidak lengkap. Pengguna dapat merespons dengan : “tidak tahu” atau “tidak yakin” pada satu atau lebih pertanyaan selama konsultasi dan sistem pakar tetap akan memberikan jawabannya.
10. Bisa digunakan sebagai media pelengkap dalam pelatihan. Pengguna pemula yang bekerja dengan sistem pakar akan menjadi lebih

berpengalaman karena adanya fasilitas penjelas yang berfungsi sebagai guru.

11. Meningkatkan kemampuan untuk menyelesaikan masalah karena sistem pakar mengambil sumber pengetahuan dari banyak pakar.

II.4.2. Kekurangan Sistem Pakar

Menurut T.Sutojo, dkk (2011 : 161), Selain manfaat, ada juga beberapa kekurangan yang ada pada sistem pakar, diantaranya :

1. Biaya yang sangat mahal untuk membuat dan memeliharanya.
2. Sulit dikembangkan karena keterbatasan keahlian dan ketersediaan pakar.
3. Sistem pakar tidak 100% bernilai benar.

II.4.3. Pemindahan Kepakaran

Menurut T.Sutojo, dkk (2011 : 164), Tujuan dari sistem pakar adalah memindahkan kepakaran dari seorang pakar ke dalam komputer, kemudian ditransfer kepada orang lain yang bukan pakar. Proses ini melibatkan empat kegiatan, yaitu :

1. Akuisisi pengetahuan (dari pakar atau sumber lain).
2. Representasi pengetahuan (pada komputer).
3. Inferensi pengetahuan.
4. Pemindahan pengetahuan ke pengguna.

II.5. Gangguan Kejiwaan

Gangguan jiwa adalah perubahan suasana perasaan dan perilaku yang terjadi tanpa alasan yang jelas, dan menyebabkan kendala terhadap diri sendiri atau orang lain. Pendapat yang berkembang di masyarakat penyakit jiwa identik dengan gila, ini adalah pandangan yang keliru turun menurun. Akan tetapi gangguan jiwa tidak sama dengan sakit jiwa. Menurut laporan dari organisasi kesehatan dunia WHO tahun 2001, sekitar 450 juta jiwa penduduk dunia menderita gangguan kesehatan jiwa. (Sri Wahyuni Wita, dkk ; 2012 : 3)

II.6. Metode *Dempster Shafer*

Menurut Esthi Dyah Rikhiana, dkk (2013 : 4), Ada berbagai macam penalaran dengan model yang lengkap dan sangat konsisten, tetapi pada kenyataannya banyak permasalahan yang tidak dapat terselesaikan secara lengkap dan konsisten. Ketidakkonsistenan tersebut adalah akibat adanya penambahan fakta baru. Penalaran yang seperti itu disebut dengan penalaran non monotonis. Untuk mengatasi ketidakkonsistenan tersebut maka dapat menggunakan penalaran dengan teori *Dempster Shafer*. Secara umum teori *Dempster Shafer* ditulis dalam suatu *interval* :

[*Belief,Plausibility*]

Belief (Bel) adalah ukuran kekuatan *evidence* dalam mendukung suatu hipotesa, jika bernilai 0 maka mengindikasikan bahwa tidak ada *evidence*, dan jika bernilai 1 menunjukkan adanya kepastian atau *Plausibility* (Pl), yang dinotasikan sebagai :

$$Pl(H) = 1 - Bel(\neg H) \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1)$$

Plausibility juga bernilai 0 sampai 1. Jika yakin akan ¬H, maka dapat dikatakan bahwa Bel(¬H)=1, dan Pl(¬H)=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 m_1 sebagai fungsi densitasnya, dan Y juga merupakan subset dari θ dengan m_2 sebagai fungsi densitasnya, maka dapat dibentuk fungsi kombinasi m1 dan m_2 sebagai m_3, dengan rumus seperti pada persamaan 2 berikut :

$$m_3(Z) = \frac{\sum_{X \cap Y = Z} m1(X).\, m2(Y)}{1 - \sum_{X \cap Y = \phi} m1(X).\, m2(Y)} \qquad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

Dimana :

$m_3(Z)$: *mass function* dari *evidence* (Z)

$m_1(X)$: *mass function* dari *evidence* (X)

$m_2(Y)$: *mass function* dari *evidence* (Y)

$Zm_1(X).m_2(Y)$: ada hasil irisan dari m_1 dan m_2

Ø $Zm_1(X).m_2(Y)$: tidak ada hasil irisan (irisan kosong (Ø))

Contoh Kasus :

Pada contoh dibawah ini, akan di cari persentase kemungkinan dari gangguan kepribadian *skizotipal* dengan menggunakan perhitungan pada tabel di bawah ini :

Tabel II.1. Contoh Gejala Pilihan Pasien

No.	Id Gejala	Gejala	Nilai Densitas
1.	GJ001	Memiliki pola bicara yang aneh	0,80
2.	GJ002	Bersifat kurang memiliki teman akrab	0,90

Maka untuk menghitung nilai *Dempster Shafer* (DS) gejala yang dipilih dengan menggunakan nilai *believe* yang telah ditentukan.

$$Pl(H) = 1 – Bel (\neg H)$$

Dimana nilai bel (*believe*) merupakan nilai densitas yang diinput oleh pakar, maka untuk mencari nilai dari kedua gejala diatas, terlebih dahulu dicari nilai dari Ø, seperti yang dibawah ini :

Gejala 1 : Pola bicara yang aneh (GJ001)

Maka : $m_1\{GK001\} = 0.80$

$m_1(Ø) = 1 – m_1\{GK001\}$

$= 1 – 0.80 = 0.20$

Gejala 2 : Kurang memiliki teman akrab (GJ002)

Maka : m_2 {GK001} = 0.90

m_2 (Ø) = 1 – m_2 {GK001}

= 1 – 0.90 = 0.10

Sehingga diperoleh nilai m_3 sebagai hasil kombinasi m_1 dan m_2 sebagai berikut :

	{GK001} 0.90	{Ø} 0.10
{GK001} 0.80	{GK001} 0.720	{GK001} 0.080
{Ø} 0.20	{GK001} 0.180	{Ø} 0.020

Selanjutnya menghitung aturan kombinasi untuk m_3 yaitu :

$$m_3\{GK001\} = \frac{0.720 + 0.080 + 0.180}{1 - 0.02}$$

$$= 0.98$$

$$m_3\{Ø\} = \frac{0.02}{1 - 0.02}$$

$$= 0.02$$

Maka nilai densitas dari kedua gejala tersebut adalah 0.98, Dengan nilai densitas 0.98 maka pasien memiliki *evidence* yang kuat mengalami gangguan kepribadian *skizotipal*. (Mikha Dayan Sinaga, dkk ; 2016 : 104-105)

II.7. PHP

PHP (*Hypertext Preprocessor*) merupakan bahasa pemrograman untuk membuat *web* yang bersifat *server-side scripting*. PHP dapat dijalankan pada berbagai macam sistem operasi, misalnya *Windows*, Linux, dan Mac OS. Selain Apache, PHP juga mendukung beberapa *server* lain, misalnya Microsoft IIS, Caudium, PWS. PHP dapat memanfaatkan *database* untuk menghasilkan halaman *web* yang dinamis. Sistem manajemen *database* yang sering digunakan PHP adalah MySQL. Namun PHP juga mendukung sistem manajemen *database* Oracle, Microsoft Access, Interbase, dBase, PostgreSQL. Hingga kini PHP sudah berkembang hingga versi 5. PHP 5 mendukung penuh *Objek-Oriented Programming* (OOP), integrasi XML mendukung semua ekstensi terbaru MySQL, pengembangan *web services* dengan SOAP dan REST, serta ratusan peningkatan lainnya dibandingkan versi sebelumnya. PHP juga bersifat *open source* sehingga setiap orang dapat menggunakannya secara gratis. (Dwi Sugiarti, dkk ; 2013 : 109)

II.8. MySQL

MySQL adalah sebuah sistem manajemen *database* yang bersifat *open source* dan merupakan pasangan serasi dari PHP, MySQL dibuat dan dikembangkan oleh MySQL AB yang berada di Swedia. MySQL dapat digunakan untuk membuat dan mengelola *database* beserta isinya. Dan dapat dimanfaatkan untuk menambahkan, mengubah, dan menghapus data yang berada di dalam *database*. MySQL merupakan sistem manajemen *database* yang bersifat

relasional artinya data-data yang dikelola dalam *database* akan diletakkan pada beberapa tabel yang terpisah sehingga manipulasi data akan menjadi lebih cepat. MySQL dapat digunakan untuk mengelola *database* mulai dari yang kecil sampai dengan yang sangat besar. MySQL juga dapat menjalankan perintah-perintah SQL untuk mengelola *database-database* relasional yang ada di dalamnya. Hingga kini MySQL sudah berkembang hingga versi 5. MySQL 5 sudah mendukung trigger untuk memudahkan pengelolaan tabel dalam *database*. (Dwi Sugiarti, dkk ; 2013 : 109-110)

II.9. UML (*Unified Modelling Language*)

Unified Modeling Language (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. (Ade Hendini ; 2016 : 108)

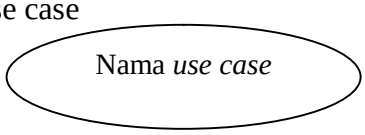
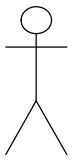
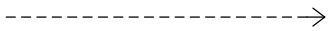
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

II.9.1. *Use Case Diagram*

Use Case Diagram merupakan pemodelan untuk kelakuan (*behavior*) sistem informasi yang akan dibuat. *Use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak

menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *Use Case Diagram* yaitu :

Tabel II.2. Simbol-simbol Pada *Use Case Diagram*

Gambar	Keterangan
Use case  	*Use Case* menggambarkan fungsionalitas yang disediakan sistem sebagai unit-unit yang bertukar pesan antar unit dengan aktor, yang dinyatakan dengan menggunakan kata kerja.
Aktor  Nama Aktor	*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*.
Asosiasi/ *association* 	*Asosiasi* antara aktor dan *use case*, digambarkan dengan garis tanpa panah yang mengindikasikan siapa atau apa yang meminta interaksi secara langsung dan bukannya mengindikasikan data.
Asosiasi/ *association* 	*Asosiasi* antara aktor dan use case yang menggunakan panah terbuka untuk mengindikasikan bila aktor berinteraksi secara pasif dengan sistem.
Include <<include>> 	*Include*, merupakan di dalam *use case* lain (*required*) atau pemanggilan *use case* oleh *use case* lain, contohnya adalah pemanggilan sebuah fungsi program.

Extend <<extend>> <----------	*Extend*, merupakan perluasan dari *use case* lain jika kondisi atau syarat terpenuhi.

Sumber : (Ade Hendini ; 2016 : 108-109)

II.9.2. Diagram Kelas (*Class Diagram*)

Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab 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 (*Assosiations*), *Generalitation* dan *Aggregation*, atribut (*Attributes*), operasi (*Operation/method*) dan *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau attribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *Multiplicity* atau *Cardinality*.

Tabel II.3. *Multiplicity Class Diagram*

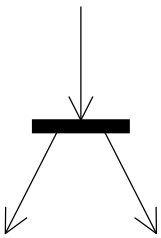
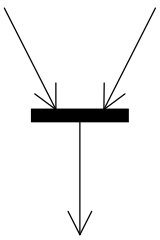
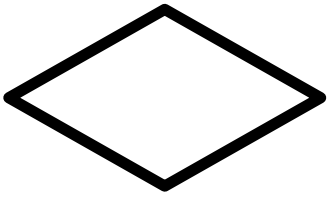
Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara, Contoh 2..4 mempunyai arti minimal 2 maksimal 4

Sumber : (Ade Hendini ; 2016 : 111)

II.9.3. Aktivitas Diagram (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *Activity Diagram* yaitu :

Tabel II.4. Simbol-simbol Pada *Activity Diagram*

Gambar	Keterangan
	*Start point,* diletakkan pada pojok kiri atas dan merupakan awal aktifitas.
	*End point,* akhir aktivitas.
	*Activities,* menggambarkan suatu proses/kegiatan bisnis.
	*Fork* (Percabangan), digunakan untuk menunjukkan kegiatan yang dilakukan secara paralel atau untuk menggabungkan dua kegiatan paralel menjadi satu.
	*Join* (Penggabungan) atau *rake*, digunakan untuk menunjukkan adanya dekomposisi.
	*Decision Points*, menggambarkan pilihan untuk pengambilan keputusan, *true* atau *false.*

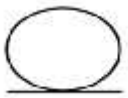
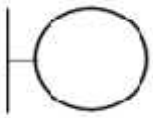
NEW SWIMLANE	*Swimlane*, pembagian *activity diagram* untuk menunjukkan siapa melakukan apa.

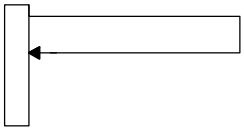
Sumber : (Ade Hendini ; 2016 : 109-110)

II.9.4. Diagram Urutan (*Sequence Diagram*)

Sequence Diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *Sequence Diagram* yaitu :

Tabel II.5. Simbol-simbol Pada *Sequence Diagram*

Gambar	Keterangan
	*Entity 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 *form entry* 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*, 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*.

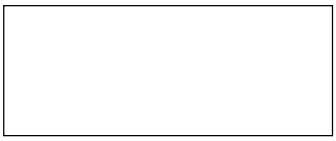
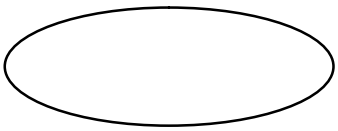
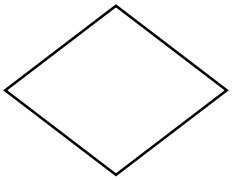
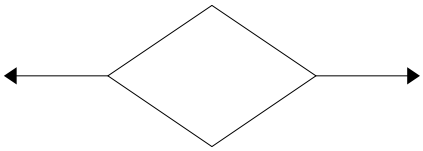
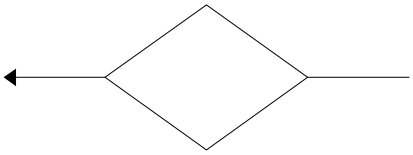
Sumber : (Ade Hendini ; 2016 : 110)

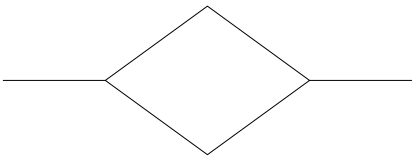
II.10. ERD (*Entity Relationship Diagram*)

Entity Relationship Diagram (ERD) adalah sekumpulan cara atau peralatan untuk mendeskripsikan data-data atau objek-objek yang dibuat berdasarkan dan berasal dari dunia nyata yang disebut entitas (*entity*) serta hubungan (*relationship*) antar entitas-entitas tersebut dengan menggunakan

beberapa notasi. Komponen-komponen pembentuk ERD dapat di lihat sebagai berikut :

Tabel II.6. Komponen-komponen *Entity Relationship Diagram* (ERD)

Notasi	Komponen	Keterangan
	Entitas	Individu yang mewakili suatu objek dan dapat dibedakan dengan objek yang lain.
	Atribut	Properti yang dimiliki oleh suatu entitas, dimana dapat mendeskripsikan karakteristik dari entitas tersebut.
	Relasi	Menunjukkan hubungan diantara sejumlah entitas yang berbeda.
	Relasi 1 : 1	Relasi yang menunjukkan bahwa setiap entitas pada himpunan entitas pertama berhubungan dengan paling banyak satu entitas pada himpunan entitas kedua.
	Relasi 1 : N	Relasi yang menunjukkan bahwa hubungan antara entitas pertama dengan entitas kedua adalah satu banding banyak atau sebaliknya. Setiap entitas dapat berelasi dengan banyak entitas pada himpunan entias yang lain.

	Relasi N : N	Hubungan ini menunjukkan bahwa setiap entitas pada himpunan entitas yang pertama dapat berhubungan dengan banyak entitas pada himpunan entitas yang kedua, demikian juga sebaliknya.

Sumber : (Doro Edi, dkk ; 2009 : 75-76)

II.10.1. Normalisasi

Normalisasi merupakan peralatan yang digunakan untuk melakukan proses pengelompokkan data menjadikan tabel-tabel yang menunjukkan entitas dan relasinya. Secara umum proses normalisasi dibagi menjadi tiga tahap, yaitu tahap tidak normal, normalisasi tahap 1, normalisasi tahap 2, normalisasi tahap 3. Pada tahap ketiga biasanya sudah akan diperoleh tabel yang optimal.

1. Bentuk Tidak Normal (*Unnormalized Form*) Pada tahap ini, semua data yang ada direkam tanpa format tertentu. Data bisa jadi mengalami duplikasi.
2. Bentuk Normal Tahap 1 (*1st Unnormalized Form*) Pada tahap ini, dibentuk tabel-tabel yang menampung data yang ada dikelompokkan berdasarkan suatu karakteristik tertentu.
3. Bentuk Normal Tahap 2 (*2st Unnormalized Form*) Pada tahap ini, dilakukan penentuan *field* kunci dari masing-masing tabel. Kunci tersebut harus unik dan mewakili tabel. Bentuk normal tahap 2 (2NF) terpenuhi jika pada sebuah tabel, semua atribut selain *primary key* memiliki

ketergantungan fungsional pada *primary key* yang utuh. (Lukman ; 2012 : 219-220)

II.11. Kamus Data

Kamus data adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan kamus data sistem analis dapat mendefinisikan data yang mengalir pada sistem dengan lengkap. Kamus data dibuat berdasarkan arus data yang dibuat pada *data flow diagram*. Arus yang ada di DFD (*Data Flow Diagram*) bersifat global dan hanya menunjukkan nama arus datanya saja. Kamus data atau *data dictionary* harus dapat mencerminkan keterangan yang jelas tentang data yang dicatatnya. (Drs. Hermansyah Sembiring, M.Kom, dkk ; 2013 : 11)