BAB II

LANDASAN TEORI

2.1. Tinjauan Pustaka

A. Sistem Pakar

Menurut Suyoto dalam Samsul Arifin (2009:60) menyatakan, "sistem pakar adalah sistem yang didesain dan diimplementasikan dengan bantuan bahasa pemprograman tertentu untuk dapat menyelesaikan masalah seperti yang dilakukan oleh para ahli".

Menurut Kusrini dalam Muhammad Rino Prayogi Siahaan (2015:80), "Sistem pakar adalah aplikasi berbasis komputer yang digunakan untuk menyelesaikan masalah sebagaimana yang dipikirkan oleh pakar".

Dari pengertian diatas maka penulis menyimpulkan bahwa sistem pakar adalah sebuah program komputer yang di desain untuk menggantikan seorang pakar dibidang tertentu.

B. Mesin Inferensi (*Inference Engine*)

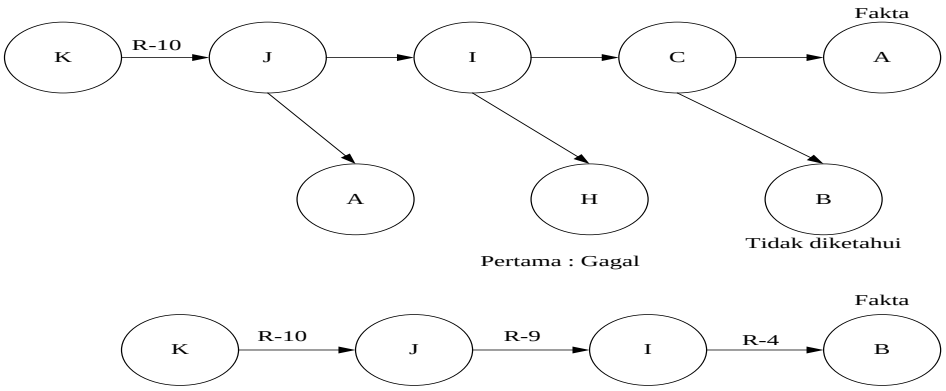
Menurut Rohman, Ami (2008:6), "Metode inferensi adalah program komputer yang memberikan metodologi untuk penalaran tentang informasi yang ada dalam basis pengetahuan dan dalam *workplace*, dan untuk menformulasikan kesimpulan".

Metode inferensi dalam sistem pakar adalah bagian yang menyediakan mekanisme fungsi berpikir dan pola-pola penalaran sistem yang digunakan oleh seorang pakar. Metode ini akan menganalisis masalah tertentu dan selanjutnya akan mecari jawaban atau kesimpulan yang terbaik serta akan memulai pelacakannya dengan mencocokan kaidah-kaidah dalam basis pengetahuan dengan fakta-fakta yang ada dalam basis data.

Pendekatan metode inferensi dalam buku Perancangan Sistem Pakar karya Nita Merlina (2012:21) ada dua yaitu :

1. *Backward Chaining*

   Backward chaining adalah pendekatan *goal-driven* yang dimulai dari harapan apa yang akan terjadi (hipotesis) dan kemudian mencari bukti yang mendukung (atau berlawanan) dengan harapan. Sering, hal ini memerlukan perumusan dan pengujian hipotesis sementara (subhipotesis). Berikut adalah gambar dari cara kerja mesin inferensi *backward chaining*.



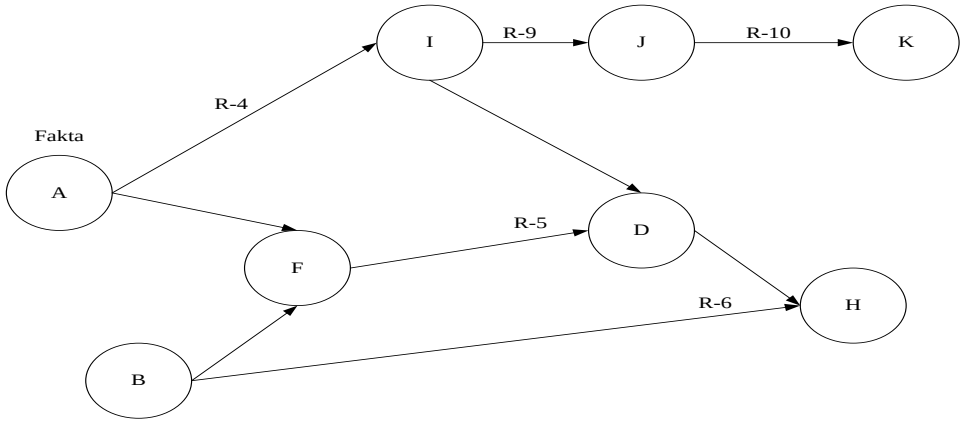
Sumber : Nita Merlina (2012:21)

Gambar II.1.

Cara Kerja Mesin Inferensi *Backward Chaining*

2. *Forward Chaining*

Forward Chaining adalah pendekatan data-driven yang dimulai dari informasi yang tersedia atau dari ide besar, kemudian mencoba menarik kesimpulan. Berikut adalah gambar dari cara kerja mesin inferensi *forward chaining*.



Sumber : Nita Merlina (2012:21)

Gambar II.2.

Cara Kerja Mesin Inferensi *Forward Chaining*.

C. Metode Pengembangan Sistem

Menurut Kusrini dan Koniyo (2007:44), "Metode pengembangan sistem yang digunakan didasarkan pada pendekatan-pendekatan sistem terstruktur, moduler dan berkembang".

Adapun pendekatan-pendekatan tersebut adalah sebagai berikut:

1. Pendekatan Sistem (*System Approach*)

Pendekatan sistem memperhatikan sistem informasi sebagai suatu kesatuan yang terintegrasi untuk masing-masing kegiatan atau aplikasinya.

Pendekatan sistem ini juga menekankan pada pencapaian sasaran keseluruhan organisasi, tidak hanya pada sasaran sistem informasi itu saja.

2. Pendekatan Terstruktur (*Structure Approach*)

   Pendekatan terstruktur dilengkapi dengan alat-alat (*tools*) dan teknik-teknik (*techniques*) yang dibutuhkan dalam pengembangan sistem sehingga hasil akhir dari sistem yang dikembangkan adalah sistem yang strukturnya didefinisikan dengan baik dan jelas.

3. Pendekatan Moduler (*Moduler Approach*)

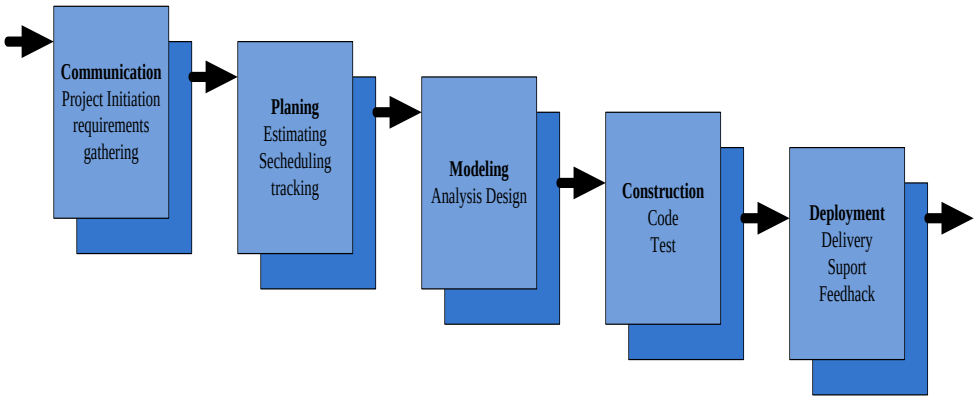
   Pendekatan moduler berusaha memecah sistem yang rumit yang menjadi beberapa bagian atau modul yang sederhana sisten lebih dipahami dan dikembangkan.

4. Pendekatan Berkembang (*Evolutionary Approach*)

   Pendekatan berkembang menerapkan teknologi canggih hanya untuk aplikasi-aplikasi yang diperlukan pada saat itu saja dan akan terus dikembangkan pada periode-periode berikutnya, memenuhi kebutuhan sesuai perkembangan teknologi.

D. Model *Waterfall*

Dalam penulisan skripsi ini, penulisan menggunakan proses *Waterfall Model* sebagai pola pengembangan sistem. Menurut Pressman (2010:39), "*Waterfall Model* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software*". Adapun tahapannya sebagai berikutnya:



Sumber : Pressman (2010:39)

Gambar II.3.

The Waterfall Model

1. *Communication*

   Pada tahap ini akan dilakukan inisiasi proyek, seperti menganalisis masalah yang ada dan tujuan yang akan dicapai. Selain itu dilakukan juga *requirements gathering*, dimana akan dikumpulkan *requirement* dari *user* melalui analisis kuesioner.

2. *Planning*

   Tahap ini merupakan tahap dimana akan dilakukan estimasi mengenai kebutuhan-kebutuhan yang diperlukan untuk membuat sebuah sistem. Selain itu, penjadwalan dalam proses pengerjaan juga ditentukan pada tahap ini.

3. *Modeling*

   Kemudian mulai masuk pada tahap perancangan dimana perancang menerjemahkan kebutuhan sistem kedalam representasi untuk menilai kualitas sebelum tahap selanjutnya dikerjakan. Tahap ini lebih difokuskan

pada atribut program, seperti struktur data, arsitektur perangkat lunak dan detail prosedur.

4. *Construction*

   Tahap ini merupakan tahap dimana perancangan diterjemahkan ke dalam bahasa yang dimengerti oleh mesin. Setelah itu dilakukan *testing* atau pengujian terhadap sistem yang telah dibuat.

5. *Deployment*

   Setelah proses pengkodean dan pengujian selesai, dilakukan pengiriman yang artinya implementasi kepada masyarakat luas. Pada tahap ini juga dilaukan pemeliharaan, perbaikan dan pengembangan agar sistem tersebut tetap dapat berjalan sebagaimana mestinya.

E. Konsep Dasar Pemprograman

Program adalah kata, ekspresi, pernyataan atau kominasinya yang disusun dan dirangkai menjadi satu kesatuan prosedur yang berupa urutan langkah untuk menyelesaikan masalah yang diimplementasikan dengan menggunakan bahasa pemprograman sehingga dapat dieksekusi oleh komputer. Bahasa pemprograman merupakan prosedur atau tata cara penulisan program. Pemprograman merupakan proses mengimplementasikan urutan langkah untuk menyelesaikan suatu masalah dengan menggunakan suatu bahasa pemprograman.

Menurut Kusrini dan Andri (2007:281), “Pemprograman yaitu suatu bentuk pemprograman yang terorganisir, yang programnya mudah dipahami dan dapat dimodifikasi secara benar”. Ciri teknik pemprograman terstruktur antara lain:

1. Mengandung algoritma pemecahan masalah yang tepat, benar, sederhana, standar dan efektif.
2. Memiliki struktur logika dan struktur program yang benar dan mudah dipahami serta menghindari penggunaan instruksi GOTO.
3. Membutuhkan biaya *testing*, pemeliharaan dan pengembangan yang rendah.
4. Memiliki dokumentasi yang baik.

   Sedangkan standar program yang baik antara lain:

   a. Standar teknik pemecahan masalah standar teknik dalam pemecahan masalah diantaranya dengan:

      1) Taknik *Top-Down*

         Suatu masalah yang komplek dibagi-bagi kedalam beberapa tingkatan kelompok masalah hingga sub bagian yang paling kecil, kemudian disusun langkah-langkah untuk menyelesaikannya secara detail.

      2) Teknik *Bottomi-Up*

         Teknik pemecahan masalah yang sudah mulai ditinggalkan karena sulit melakukan standarisasi proses dari prosedur-prosedur yang sudah ada untuk digabungkan menjadi satu kesatuan.

   b. Standar Penyusunan Program

      Terdapat beberapa standar dalam penyusunan program, yaitu:

      1) Kebenaran logika dan penulisan
      2) Waktu minimum untuk penulisan program
      3) Kecepatan maksimum eksekusi program

4) Ekspresi pengunaan memori

5) Kemudahan merawat dan mengembangkan program'

6) *User friendly*

7) Probabilitas

8) Pemprograman modular

c. Standar Perawatan Program

1) Dokumentasi

2) Penulisan instruksi

F. Peralatan Pendukung Sistem (*Tools System*)

Merupakan alat yang digunakan untuk menggambarkan bentuk logika model dari suatu sistem dengan menggunakan simbol-simbol, lambang-lambang, diagram-diagram yang menunjukan secara tepat arti dan fungsinya. Adapun peralatan pendukung sistem (*tools system*) yang dijelaskan sebagai model sistem yang akan dirancang adalah sebagai berikut:

1. UML (*Unfied Modeling Language*)

Menurut Nugroho (2010:6), "UML (*Unfied Modeling Language*) adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Pemodelan (*modeling*) sesungguhnya digunakan untuk penyederhanaan permasalahan-permasalahan yang kompleks sedemikian rupa sehingga lebih mudah dipelajari dan dipaham".

> Menurut Nugrorho (2009:4) menyatakan bahwa:
> UML (*Unfied Modeling Language*) adalah metodologi kolaborasi antara metoda-metoda Booch, OMT (*Object Modeling Technique*), serta OOSE (*Object Oriented Software Enggineering*) dan beberapa metoda lainnya, merupakan metodologi yang paling sering digunakan saat ini untuk analisa

dan perancangan sistem dengan metodologi berorientasi objek mengadaptasi maraknya penggunaan bahasa pemprograman berorientasi objek (OOP).

Itu merupakan definisi yang sederhana. Bahkan, UML adalah hal yang berbeda bagi beberapa orang yang berbeda. Hal ini berasal baik dari sejarahnya sendiri dan dari pandangan yang berbeda bahwa orang memiliki tentang apa yang membuat suatu proses rekayasa perangkat lunak yang efektif.

Karena program yang dibuat merupakan program terstruktur, maka pada tahapan ini penulis akan menjelaskan tentang diagram-diagramnya yaitu:

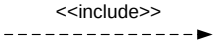
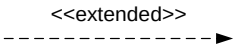
a. *Use Case Diagram*

Use case diagram dapat digunakan selama proses analisis untuk menangkap *requirements* sistem dan untuk memahami bagaimana sistem seharusnya berkerja. Selama tahap desain, *use case* diagram menetapkan perilaku (*behavior*) sistem saat diimplementasikan. Dalam sebuah model mungkin terdapat satu atau beberapa *use case diagram*. Beberapa simbol-simbol yang ada pada diagram *use case* dalam tabel II.1.

Tabel II.1.

Simbol – Simbol *Use Case diagram*

SIMBOL	NAMA	KETERANGAN
	Aktor	Seseorang atau sesuatu yang berinteraksi dengan sistem yang sedang kita kembangkan.
	Use Case	Peringkat Tertinggi dari fungsional yang dimiliki sistem.

	Relasi Asosiasi	Relasi yang terjadi antara aktor dengan use case biasanya berupa asosiasi.
	Unclude Relationship	Relasi cakupan memungkinkan suatu use case untuk menggunakan fungsionalitas yang disediakan oleh use case yang lainnya.
	Extends Relationship	Meningkatkan suatu use case memiliki keuntuntungan untuk memperluas fungsional yang disediakan use case yang lainnya.

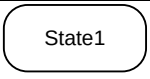
Sumber : Nugroho (2010 : 8)

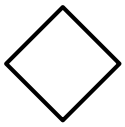
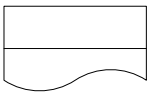
b. *Activity diagram*

Activity diagram memodelkan alur kerja (*work flow*) sebuah proses bisnis dan urutan aktivitas dalam suatu proses. Diagram ini sangat mirip dengan sebuah *flowchart* karena dapat dimodelkan sebuah alur kerja dari satu aktivitas ke aktivitas lainnya atau dari satu aktivitas kedalam keadaan sesaat (*state*). Beberapa simbol-simbol yang ada pada *activity diagram* ditunjukan dalam tabel II.2.

Tabel II.2.

Simbol – Simbol *Activity diagram*

SIMBOL	NAMA	KETERANGAN
	*Start State*	Memperlihatkan aliran kerja berawal.
	*End State*	Memperlihatkan aliran kerja berakhir.
	*State*	Menambahkan *state* suatu objek.

	*Activity*	Menggambarkan langkah-langkah dalam aliran kerja.
	*Decision*	Memperlihatkan pengambilan keputusan dari dua atau lebih langkah pada aliran-aliran kerja
	*Transition*	Memperlihatkan arah aliran-aliran kerja bergerak dari satu *activity* ke *activity* lainnya.
	*Swimlane*	Memperlihatkan orang atau badan yang bertanggung jawab untuk melaksanakan tugas-tugas tertentu pada *activity diagram.*

Sumber : Nugroho (2010 : 9)

c. *Component diagram*

Component diagram digunakan ketika ingin membagi sistem menjadi komponen dan ingin menampilkan hubungan timbal balik mereka antarmuka atau rincian dari komponen ke dalam struktur-tingkat yang lebih rendah.

d. *Deployment diagram*

Deployement diagram menunjukan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian *software* yang berjalan pada bagian-bagian *hardware*.

Ada empat macam relasi *Unified Modeling Language* (UML) yaitu:

1) Pengklasifikasian (*Classifier*)

Pengklasifikasian (*classifier*) pada prinsipnya merupakan konsep diskret dalam model yang memiliki identitas (*identity*), *state*,

perilaku (*behavior*), serta relasi dengan mengklasifikasi yang lainnya (*relationship*).

2) Asosiasi (*Association*)

Asosiasi (*Association*) pada dasarnya mendeskrpsikan koneksi diskret antara objek atau antar *instance* lain dalam sistem atau perangkat lunak yang sedang dikembangkan.

3) Generalisasi (*Generalization*)

Menggambarkan hubungan anatara *use case* yang bersifat umum dengan *use case-use case* yang bersifat lebih spesifik.

4) Realisasi (*Realization*)

Relasi realisasi (*realization*) menghubungkan elemen-elemen model, misalnya kelas ke elemen-elemen model lainnya, seperti suatu antarmuka yang menyediakan spesifikasi perilaku tetapi bukan strukturnya atau implementasinya.

2. ERD (*Entity Relationship Diagram*)

Menurut Yuhefizar (2008:17), "*Entity Relationship Diagram* digunakan untuk menggambarkan secara sistematis hubungan antar *entity-entity* yang ada dalam suatu sistem *database* menggunakan simbol-simbol sehingga lebih mudah dipaham". Simbol-simbol yang boleh digunakan adalah:

a. Persegi panjang, berfungsi untuk menyatakan suatu *entity*.
b. Elips, berfungsi untuk menyatakan *attribute*, jika diberi garis bawah menandakan bahwa *attribute*tersebut merupakan *attribute* / *field* kunci.

c. Belah ketupat, menyatakan jenis relasi.

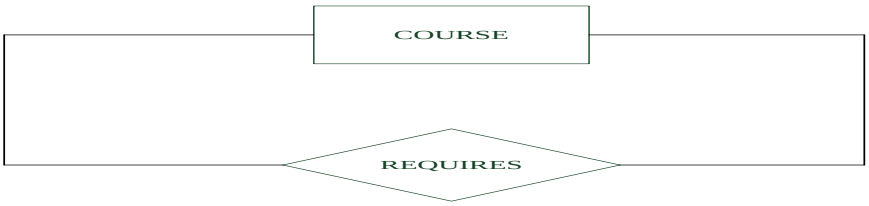
d. Garis penghubung antara relasi dengan *entity* dan antara *entity* dengan *attribute*.

1) Derajat *Relationship*.

Terdapat tiga macam derajat dari *relationship*, yaitu:

a) Derajat Satu (*Unary Degree*)

Bila satu *entity* mempunyai relasi terhadap dirinya sendiri.



Gambar II.4.

Contoh Derajat Satu (*Unary Degree*)

b) Derajat Dua (*Binary Degree*)

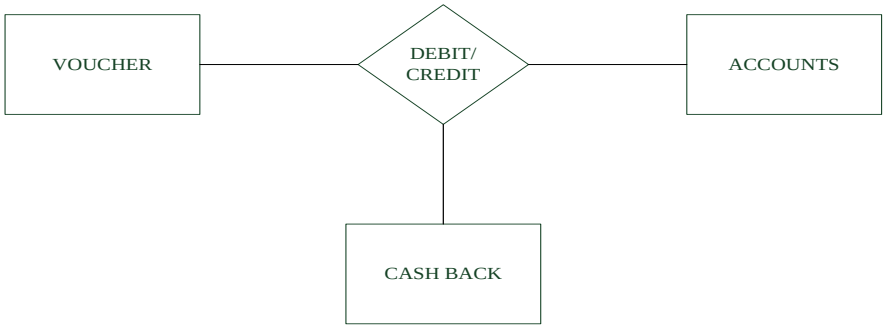
Bila satu relasi menghubungkan dua *entity*.

Gambar II.5.

Contoh Derajat Dua (*Binary Degree*)

c) Derajat Tiga (*Ternary Degree*)

Bila satu *entity* menghubungkan lebih dari dua *entity*.



Gambar II.6.

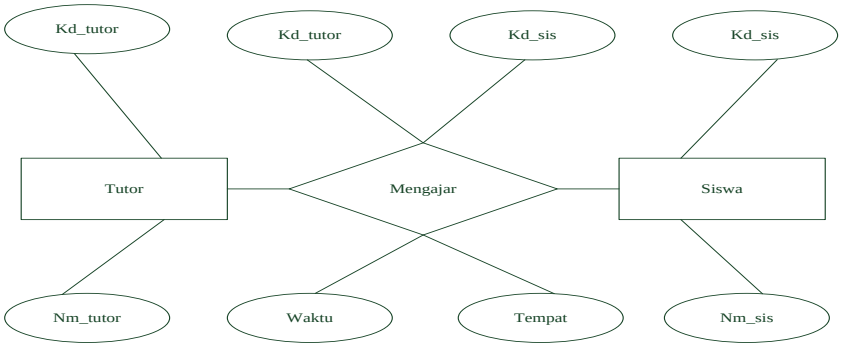
Contoh Derajat Tiga (*Ternary Degree*)

2) *Cardinality Ratio*

Terdapat empat jenis *Cardinality Ratio*, yaitu:

a) 1 : 1 (*One-To-One*)

Sebuah *entity* A diasosiasikan pada sebuah *entity* B, dan sebuah *entity* B diasosiasikan dengan paling banyak sebuah *entity* A.



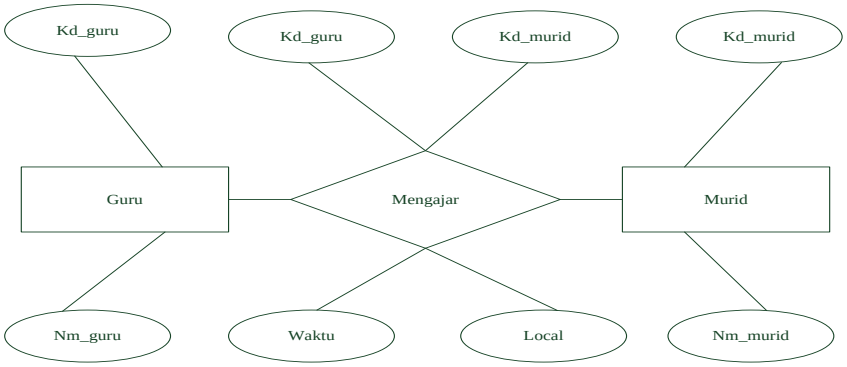
Sumber : Yuhefizar (2008:18)

Gambar II.7.

Contoh relasi 1 : 1 (*One-To-One*)

b) 1 : N (*One-To-Many*)

Sebuah *entity* A diasosiasikan dengan sejumlah *entity* B, tetapi *entity* B dapat diasosiasikan paling banyak satu *entity* A.



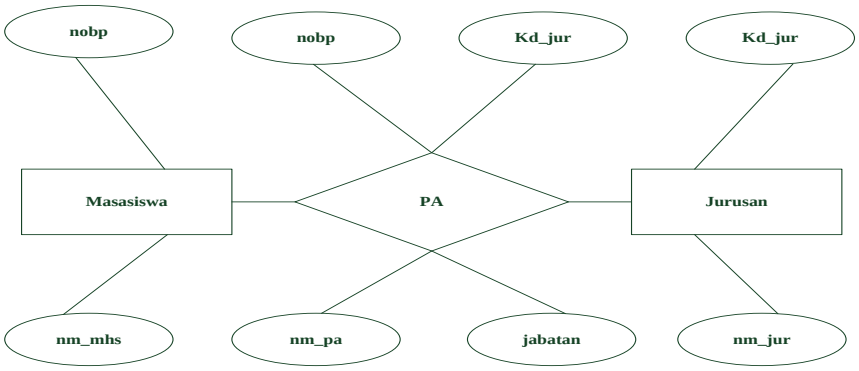
Sumber ; Yuhefizar (2008:19)

Gambar II.8.

Contoh relasi 1 : N (*One-To-Many*)

c) N : 1 (*Many-To-One*)

Suatu *entity* A dapat diasosiasikan dengan paling banyak sebuah *entity* B, tetapi *entity* B dapat diasosiasikan dengan sejumlah *entity* di A.



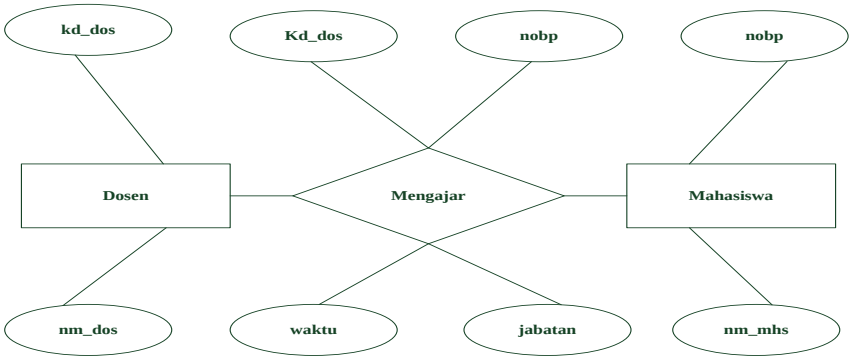
Sumber : Yuhefizar (2008:20)

Gambar II.9.

Contoh relasi N : 1 (*Many-To-One*)

d) M : N (*Many-To-Many*)

Suatu *entity* A dapat diasosiasikan dengan sejumlah *entity* B dan *entity* B dapat diasosiasikan dengan jumlah *entity* di A.



Sumber : Yuhefizar (2008:21)

Gambar II.10.

Contoh relasi M : N (*Many-To-Many*)

3) *Participation Constraint*

Terdapat dua macam *Participation Constraint*, yaitu:

a) *Total Participation*

Keberadaan suatu *entity* tergantung pada hubungannya dengan *entity* lain.

b) *Partial Participation*

Keberadaan suatu *entity* tidak tergantung hubungannya dengan *entity* lain.

4) Tahapan pembuatan ERD

Menurut Yuhefizar (2008:21) terdapat empat tahapan pembuatan ERD, yaitu:

a) Mengidentifikasi dan menetapkan seluruh *entity* yang terlibat dalam sistem *database* tersebut.

b) Menentukan *attribute-attribute* atau *field* dari masing-masing *entity* berserta kunci (*key*)-nya.

c) Mengidentifikasi dan menetapkan seluruh himpunan relasi di antar himpunan-himpunan *entity* yang ada berserta kunci tamu (*foreign key*)-nya.

d) Menentukan derajat relasi untuk setiap himpunan relasi.

2.2. Penelitian Terkait

> Menurut Sartini (2015 : 1) Menyimpulkan bahwa:
> Alat komunikasi saat ini juga semakin canggih, salah satunya handphone. Seperti barang elektronik lainnya, *handphone* pun sering mengalami kerusakan, hal yang sering terjadi adalah kerusakan pada *hardware handphone*. metode pelacakan kedepan (*forward chaining*) yaitu metode inferensi yang melakukan penalaran dari suatu masalah kepada solusinya. Dalam jurnal ini pedektesian dini menggunakan metode *forward chaining* pada mesin inferensi pakar.

> Menurut Said, jazuli (2007 : 1) Menyimpulkan bahwa:
> Dalam mengoprasikan komputer tentu sering mengalami masalah dengan sistem operasi *Windows*, seperti komputer lambat ketika membaca data, icon-icon yang hilang di deskop, sistem *crash* aplikasi atau file yang tidak dapat dijalankan, ataupun muncul pesan kesalahan yang tidak dimengerti, masalah-masalah yang muncul ini tidak jarang berakibat fatal sehingga menggangu pekerjaan atau aktifitas yang sedang dilakukan..

Berdasarkan jurnal diatas untuk kerusakan yang lebih spesifik pada perangkat keras komputer juga membutuhkan pengetahuan dan memperbaikinya. Baik itu pengetahuan gejala-gejala kerusakannya sampai cara memperbaikinya. Untuk lebih memberi kemudahan dalam penangananya, maka perlu dibuatnya sebuah program sistem pakar untuk mendiagnosis kerusakan pada perangkat keras (*hardware*) komputer dengan metode *forward chaining*.