BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Telah ada beberapa penelitian yang dilakukan terkait dengan metode *certainty factor* diantaranya adalah :

Tabel II.1 Penelitian Terkait

No	Penulis	Judul	Hasil Penelitian
1	Mohammad Arifin (2017)	Penerapan Metode Certainty Factor Untuk Sistem Pakar Diagnosis Hama Dan Penyakit Pada Tanaman Tembakau	Dalam penelitian ini, sistem pakar diagnosis hama dan penyakit pada tanaman tembakau dibangun untuk membantu mendiagnosa jenis hama atau penyakit yang menyerang tanaman tembakau, serta memberikan berbagai solusi untuk hama atau penyakit. Metode yang digunakan pada sistem pakar ini adalah metode *Certainty Factor*. Metode *Certainty Factor* dipilih karena metode ini cocok dalam proses penentuan identifikasi hama dan penyakit, dan hasil dari penerapan metode ini adalah persentase. Persentase sistem disini merupakan tingkat akurasi penentuan penyakit atau hama yang menjangkiti tanaman tembakau. Penentuan persentase dipengaruhi oleh nilai MB yang didapat dari sistem dan nilai MD yang didapat dari penilaian seorang pakar.
2	Aryu Hanifah Aji	Sistem Pakar Diagnosa	Sistem pakar diagnosa penyakit ibu hamil menggunakan metode *Certanty Factor* (CF)

	(2018)	Penyakit Ibu Hamil Menggunakan Metode *Certainty Factor* (CF)	yang dapat membantu mengenali penyakit selama kehamilan berlangsung berdasarkan gejala-gejala yang dirasakan ibu hamil serta tempat rujukan yang harus dituju oleh pasien. Metode CF memiliki kinerja sistem yang mampu berjalan sesuai kebutuhan fungsional dan hasil presentase akurasi tinggi. Selain itu metode CF dapat mengambarkan tingkat keyakinan seorang pakar terhadap masalah yang sedang dihadapi. Berdasarkan hasil pengujian, diperoleh hasil 100% fungsionalitas sistem pakar diagnosa penyakit ibu hamil berjalan sesuai dengan daftar kebutuhan sistem dan sistem mempunyai tingkat akurasi sebesar 100%.
3	Puji Sari Ramadhan, Usti Fatimah Sitorus Pane	Sistem E-*Healthcare* Untuk Mendiagnosa Penyakit Inflamasi Dermatitis Imun Anak Dengan Menggunakan Metode *Certainty Factor*	Kurangnya pengetahuan masyarakat serta tidak tercukupi tenaga ahli spesialis dibidang imunologi, mengakibatkan sulitnya penanganan terhadap pasien anak yang menderita penyakit Inflamasi Dermatitis Imun. Melihat fenomena yang terjadi maka diperlukan sebuah Sistem *E-Healthcare* yang mampu menerapkan metode *Certainty Factor* untuk mendiagnosa jenis penyakit Inflamasi Dermatitis Imun pada anak berdasarkan gejalagejala klinis yang terjadi, proses penerapannya dengan terlebih dahulu mengumpulkan basis pengetahuan, kemudian melakukan penelusuran *inferensi Forward Chaining* terhadap *rule-rule* yang ada dan selanjutnya melakukan proses perhitungan

			metode *Certainty Factor* untuk mengetahui probabilitas dan jenis penyakit Inflamasi Dermatitis Imun pada anak. Dengan adanya Sistem *E-Healthcare* ini dapat memberikan kemudahan Dalam pengambilan kesimpulan untuk dijadikan diagnosa awal.

II.2. Landasan Teori

II.2.1. Sistem Pakar

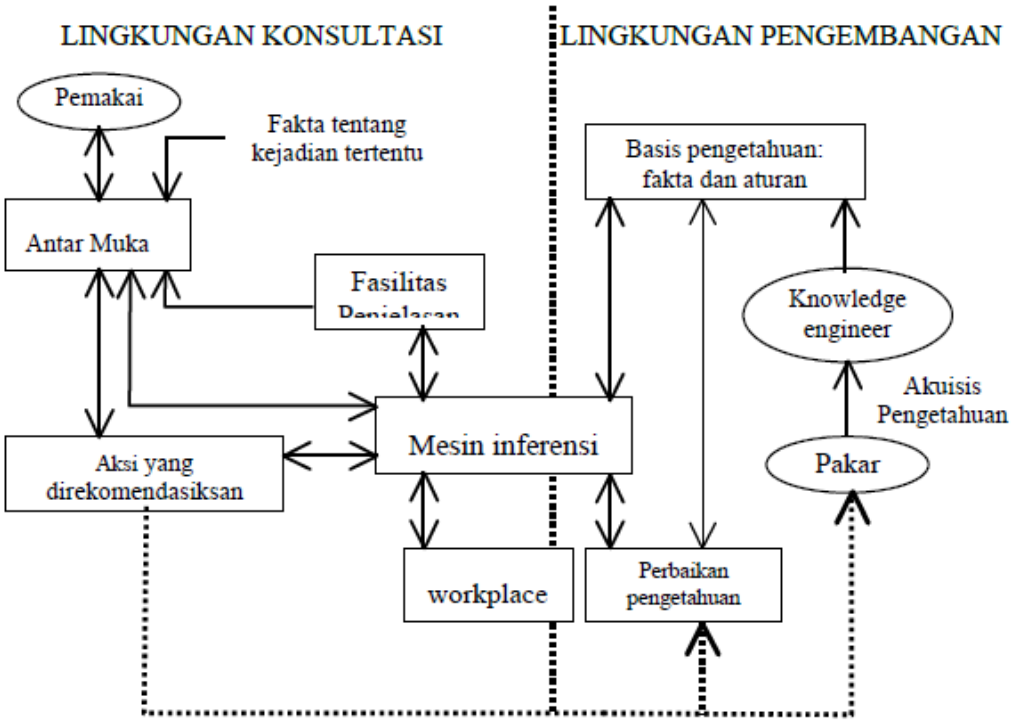
Sistem Pakar (*expert system*) adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer agar komputer dapat menyelesaikan masalah seperti yang biasa dilakukan oleh para ahli. Sistem pakar yang baik dirancang agar dapat menyelesaikan suatu permasalahan tertentu dengan meniru kerja dari para ahli. Dengan sistem pakar ini, orang awam pun dapat menyelesaikan masalah yang cukup rumit yang sebenarnya hanya dapat diselesaikan dengan bantuan para ahli (Muliadi, 2017 : 209).

Sistem pakar merupakan suatu bagian ilmu-ilmu artificial intelligence untuk dibuat suatu program aplikasi diagnosa penyakit pada manusia yang terkomputerisasi serta berusaha menggantikan dan menirukan proses penalaran dari seorang ahlinya atau pakar dalam memecahkan masalah spesifikasi yang dapat dikatakan duplikat dari seorang pakar karena pengetahuan ilmu tersebut tersimpan di dalam suatu sistem database. Secara umum sistem pakar adalah sistem yang berusaha mengadopsi pengetahuan manusia ke komputer yang dirancang untuk memodelkan kemampuan menyelesaikan masalah seperti layaknya pakar. Sistem Pakar (*Expert System*) adalah salah satu cabang AI yang

membuat penggunaan secara luas *knowledge* yang khusus untuk penyelesaian masalah tingkat manusia yang pakar. Komputer berbasis sistem pakar adalah program komputer yang mempunyai pengetahuan yang berasal dari manusia yang berpengetahuan luas atau pakar dalam domain tertentu, di mana pengetahuan di sini adalah pengetahuan manusia yang sangat minim penyebarannya, mahal serta susah didapat. (Khairani Puspita : 2018)

II.2.1.1. Struktur Sistem Pakar

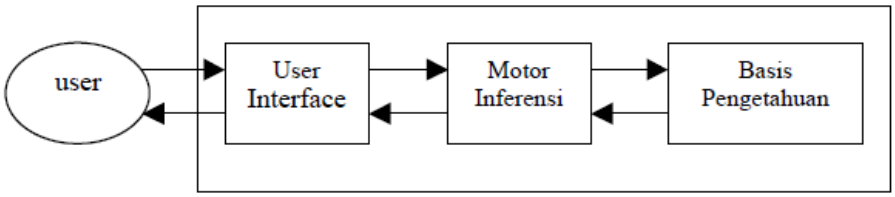
Struktur Sistem pakar memiliki 2 bagian utama antara lain lingkungan pengembangan (*development environment*) yaitu bagian yang digunakan untuk memasukkan pengetahuan pakar ke dalam lingkungan sistem pakar dan lingkungan konsultasi (*consutation environment*), yaitu bagian yang digunakan oleh pengguna yang ukan pakar untuk memperoleh pengetahuan. Struktur sistem pakar dalam dua bagian tersebu dapat di lihan pada Gambar II.1. berikut : (Yossi Octavia : 2015)



Gambar II.1. Struktur Sistem Pakar

II.2.1.2. Komponen Sistem Pakar

Sistem pakar pada umumnya mempunyai tiga elemen, yaitu : basis pengetahuan *(Knowledge Base),* Mesin Inferensi *(Inference Engine),* dan Antarmuka Pemakai *(User Interface).* Dengan blok umum hubungan dari ketiga komponen sistem pakar tersebut dapat dilihat pada Gambar II.2 di bawah ini : (Yossi Octavia : 2015)



Gambar II.2. Diagram Blok umum Sistem Pakar

Dari gambar diatas, Sistem Pakar terdiri dari 3 komponen utama, yaitu: basis pengetahuan (*knowledge bases*), *motor inferensi* dan *interface.*

a. Representasi Pengetahuan *(Knowledge Base)*

Representasi pengetahuan adalah metode yang digunakan untuk mengkodekan pengetahuan dalam sebuah sistem pakar yang berbasis pengetahuan. Representasi pengetahuan dimaksudkan untuk menangkap sifat-sifat penting problema dan membuat informasi tersebut dapat diakses oleh prosedur pemecahan masalah.

b. Antarmuka Pengguna

Antarmuka pengguna adalah perangkat lunak yang menyediakan media komunikasi antara pengguna dengan sistem. *User Interface* merupakan mekanisme yang digunakan oleh pengguna dan sistem pakar untuk berkomunikasi. Antarmuka menerima informasi dari pemakai dan mengubahnya ke dalam bentuk yang dapat diterima oleh sistem. Selain itu, antarmuka menerima informasi dari sistem dan menyajikannya ke dalam bentuk yang dapat dimengerti oleh pemakai.

c. Mesin Inferensi (motor inferensi)

Mesin inferensi adalah program komputer yang memberikan metodologi untuk penalaran tentang informasi yang ada dalam basis pengetahuan dan dalam *workplace* dan untuk memformulasikan kesimpulan. (Yossi Octavia : 2015)

II.2.3. *Certainty Factor*

Certainty Factor adalah suatu metode untuk membuktikan apakah suatu fakta itu pasti ataukah tidak pasti yang berbentuk *metric* yang biasanya digunakan dalam sistem pakar. Metode ini sangat cocok untuk sistem pakar yang mendiagnosis sesuatu yang belum pasti. Metode *certainty factor* ini hanya bisa mengolah 2 bobot dalam sekaliperhitungan. Untuk bobot yang lebih dari 2 banyaknya, untuk melakukan perhitungan tidak terjadi masalah apabila bobot yang dihitung teracak, artinya tidak ada aturan untuk mengkombinasikan bobotnya, karena untuk kombinasi seperti apapun hasilnya akan tetap sama. Misalnya, untuk mengetahui apakah seorang pasien tersebut menderita penyakit jantung atau tidak, dilihat dari hasil perhitungan bobot setelah semua keluhan-keluhan diinputkan dan semua bobot dihitung dengan menggunakan metode *certainty factor*. Pasien yang divonis mengidap penyakit jantung adalah pasien yang memiliki bobot mendekati +1 dengan keluhan-keluhan yang dimiliki mengarah kepada penyakit jantung. Sedangkan pasien yang mempunyai bobot mendekati -1 adalah pasien yang dianggap tidak mengidap penyakit jantung, serta pasien yang memiliki bobot sama dengan 0 diagnosisnya tidak diketahui atau *unknown* atau bisa disebut dengan netral.

$$CF(H, E) = MB(H, E) - MD(H, E) \quad \text{....……..(1)}$$

Dimana :

- CF (H, E) : *Certainty Factor* dari hipotesis H yang dipengaruhi oleh gejala(evidence) E. Besarnya CF berkisar antara -1 sampai 1. Nilai -1

menunjukkan ketidakpercayaan mutlak sedangkan nilai 1 menunjukkan kepercayaan mutlak.

- MB (H, E) : Ukuran kenaikan kepercayaan (*Measure Of Increased Belief*) terhadap hipotesis H yang dipengaruhi oleh gejala E.
- MD (H, E) : Ukuran kenaikan ketidakpercayaan (*Measure Of Increased Disbelief*)terhadap hipotesis H yang dipengaruhi oleh gejala E.Bentuk dasarrumus *certainty factor*, adalah sebuah aturan JIKA E MAKA H seperti ditunjukkan oleh persamaan 2 berikut:

CF (H, e) = CF (E, e) * CF (H, E)	....……..(2)

Dimana :

- CF (H, e) : *certainty factor* hipotesis yang dipengaruhi oleh *evidence* e.
- CF (E, e) : *certainty factor evidence* E yang dipengaruhi oleh *evidence* e.
- CF (H, E) : *certainty factor* hipotesis dengan asumsi *evidence* diketahui dengan pasti, yaitu ketika CF(E, e) = 1. Jika semua *evidence* pada *antecedent* diketahui dengan pasti maka persamaannya akan menjadi:

CF (E, e) = CF (H, E)	....…….……...(3)

Dalam aplikasinya, CF(H,E) merupakan nilai kepastian yang diberikan oleh pakar terhadap suatu aturan, sedangkan CF(E,e) merupakan nilai kerpercayaan yang diberikan oleh pengguna terhadap gejala yang dialaminya (Hasibuan, dkk, 2017 : 31 – 33).

II.2.4. *Unified Modeling Language* (UML)

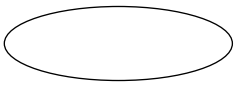
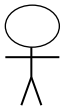
Unified Modelling Language (UML) merupakan salah satu bentuk *language* atau bahasa, mneurut pencetusnya UML didefinisikan sebagai bahasa *visuai* untuk menjelaskan, memberikan spesifikasi, merancang, membuat model, dan mendokumentasikan aspek aspek dari sebuah sistem. Perancangan sistem pada UML adalah sebagai berikut:

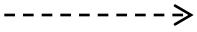
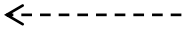
Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut :

1. *Use case* Diagram

   Use Case atau diagram *use case* merupakan pemodelan untuk kelakuan *(behavior)* sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dari dengan sistem informasi yang akan dibuat. Secara kasar, *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi itu.

Tabel II.2. Tabel Simbol *Use Case Diagram*

Gambar	Keterangan
	Use Case, Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor
	Actor, Menspesifikasikan himpuan peran yang pengguna mainkan ketika berinteraksi dengan *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.
	*Include,* Hubungan dimana objek anak (*descendent*) berbagi perilaku dan struktur data dari objek yang ada di atasnya objek induk (*ancestor*).
	Menspesifikasikan bahwa *use case* sumber secara *eksplisit*.
	*Extend,* Menspesifikasikan bahwa *use case* target memperluas perilaku dari *use case* sumber pada suatu titik yang diberikan.

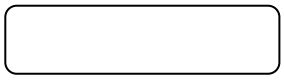
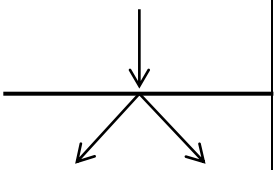
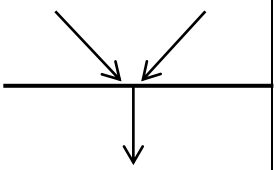
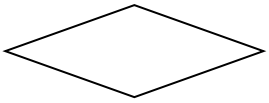
(Sumber : Ade Hendini; 2016: 109-110)

2. Diagram Aktivitas (*Activity Diagram*)

   Activity Diagram (Diagram Aktifitas) menggambarkan berbagai alir aktifitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Diagram Aktifitas merupakan *state* diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-*trigger* oleh selesainya state sebelumnya (*internal processing*). Oleh karena itu Diagram AKtifitas tidak menggambarkan *behaviour* internal sebuah sistem

(dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. Menggambarkan proses bisnis dan urutan aktifitas dalam sebuah proses. Dipakai pada *business*

Tabel II.3 Tebel Simbol *Activity Diagram*

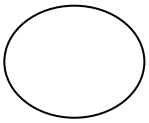
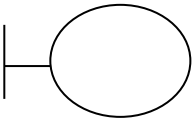
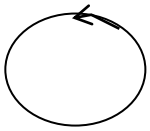
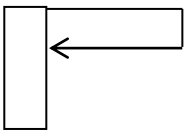
Gambar	Keterangan
	*Start point*, Bagaimana objek dibentuk atau diawali.
	*End point*, Bagaimana objek dibentuk dan dihancurkan
	*Activites*, Memperlihatkan bagaimana masing-masing kelas antarmuka saling berinteraksi satu sama lain
	*Fork* (Percabangan), Satu aliran yang pada tahap tertentu berubah menjadi beberapa aliran
	*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 : Ade Hendini; 2016: 109-110)

3. 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.4. Tabel Simbol *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 *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 : Ade Hendini; 2016: 3)

4. *Class Diagram* (Diagram Kelas)

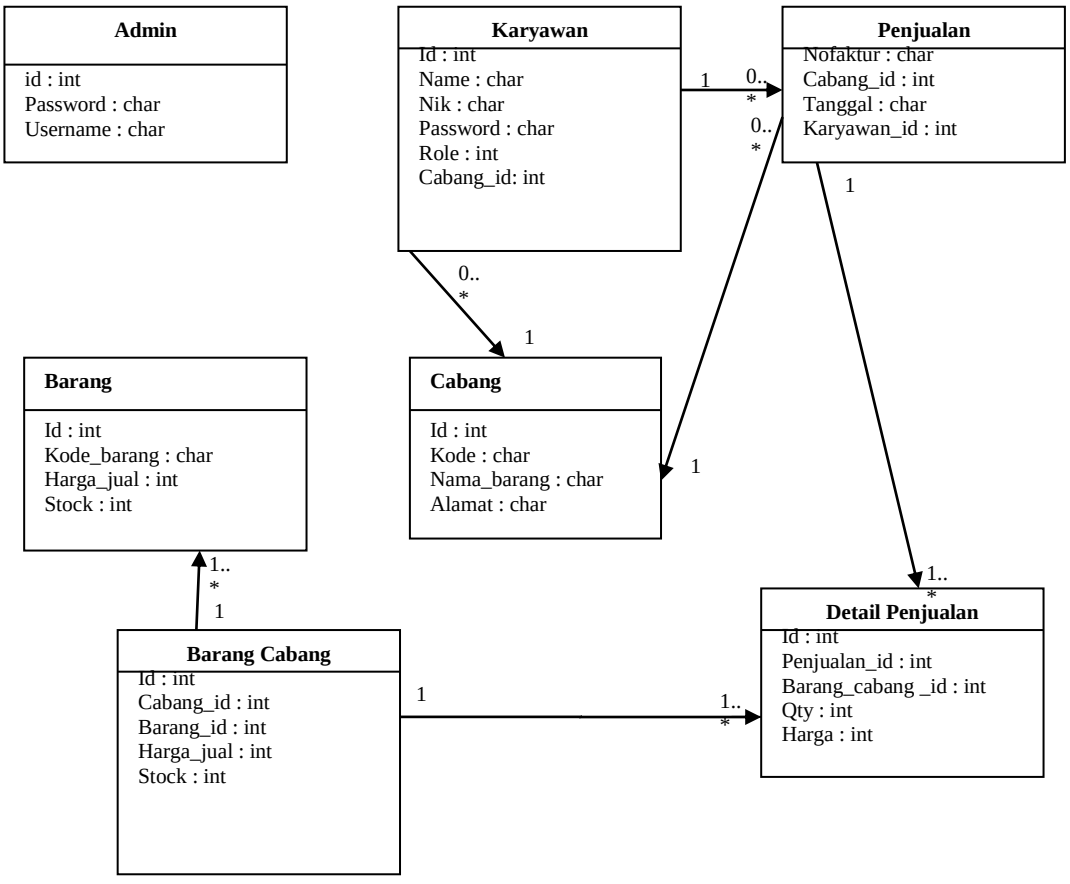
   Merupakan hubungan antar kelas dan penjelasan detail tiap-tiap kelas di dalam model desain dari suatu sistem, juga memperlihatkan aturan-aturan dan tanggung jawab entitas yang menentukan perilaku sistem. *Class diagram* juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/ Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau *cardinality.*

Tabel II.5. Tabel *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 maksimum 4

(Sumber : Ade Hendini; 2016: 111)

Adapun gambar class diagram adalah sebagai berikut :



Gambar II.2. *Class* Diagram
(*Sumber : Windu Gata : 2013*)

Adapun penjelasan dari class diagram adalah sebagai berikut Terdapat 7 tabel yaiu tabel barang cabang, tabel detail penjualan, tabel barang, tabel cabang, tabel admin, tabel karyawan dan tabel penjualan. Tabel karyawan memiliki relasi pada tabel penjualan dan tabel cabang. Tabel cabang berelasi pada tabel barang, barang cabang dan tabel detail penjualan.

II.2.5 WEB

Web adalah aplikasi yang berisikan dokuumen-dokumen multimedia (teks, gambar, suara, animasi, video) didalamnya yang menggunakan protocol HTTP *(hypertext transfer protocol)* dan untuk mengaksesnya menggunakan perangkat lunak yang disebut browser (Dedi, 2016 : 2).

II.2.6. HTML (*Hypertext Markup Language*)

HyperText Markup Language (HTML) adalah sebuah bahasa markup yang digunakan untuk membuat sebuah halaman web, menampilkan berbagai informasi di dalam sebuah Penjelajah web Internet dan formating hypertext sederhana yang ditulis kedalam berkas format ASCII agar dapat menghasilkan tampilan wujudyang terintegerasi. Dengan kata lain, berkas yang dibuat dalam perangkat lunakpengolah kata dan disimpan kedalam format ASCII normal sehingga menjadi home page dengan perintahperintah HTML. Bermula dari sebuah bahasa yang sebelumnya banyak digunakan di dunia penerbitan dan percetakan yang disebut dengan SGML(*Standard Generalized Markup Language*), HTML adalah sebuah standar yang digunakan secara luas untuk menampilkan halaman web.

HTML saatini merupakan standar Internet yang didefinisikan dan dikendalikan penggunaannya oleh *World Wide Web Consortium*(W3C). HTML dibuat oleh kolaborasi Caillau TIM dengan Berners-lee robert ketika mereka bekerja di CERNpada tahun 1989 (CERN adalah lembaga penelitian fisika energi tinggi di Jenewa). (Harison, 2017 : 43).

II.2.7. PHP *(Hypertext Preprocessor)*

PHP merupakan kependekan dari kata *Hypertext Preprocessor*. PHP tergolong sebagai perangkat lunak open source yang diatur dalam aturan *general purpose licences* (GPL). Bahasa pemograman PHP sangat cocok dikembangkan dalam lingkungan *web*, karena PHP bisa diletakan pada *script* HTML atau sebaliknya (Hikmah, 2015 : 2).

II.2.8. *MySQL (My Structure Query Language)*

MySQL adalah salah satu *Database Management System (DBMS)* dari sekian banyak *DBMS* seperti *Oracle*, *MS SQL*, *PostagreSQL*, dan lainnya. *MySQL* berfungsi untuk mengolah *Database* menggunakan bahasa *SQL*. *MySQL* bersifat *open source* sehingga bisa menggunakannya secara gratis. Pemograman PHP juga sangat mendukung dengan *database* MySQL (Hikmah, 2015 : 2).

II.2.9. Normalisasi

Normalisasi merupakan parameter digunakan untuk menghindari duplikasi terhadap tabel dalam basis data dan juga merupakan proses sebuah tabel yang

masih memiliki beberapa anomali atau ketidak wajaran sehingga menghasilkan tabel yang lebih sederhana dan struktur yang bagus, yaitu sebuah tabel yang tidak memiliki data r*edundancy* dan memungkinkan *user* untuk melakukan *insert, delete*, dan *update* pada baris (*recod*) tanpa menyebabkan inkonsistensi data. Tujuannya untuk menghindari beberapa anomali

1. *Insertion Anomaly* adalah proses melakukan penambahan *recod* baru akan tetapi mempengaruhi *user* untuk terjadinya duplikasi data.
2. *Deletion Anomaly* adalah proses melakukan penghapusan *record* akan tetapi akan menyebabkan hilangnya data yang akan dibutuhkan pada *record* lain.
3. *Modification Anomaly* adalah proses merubah data pada sebuah *record* mempengaruhi perubahan pada *record* lain karena adanya duplikasi.

Proses pernomalan tabel ada beberapa tahap yang harus dilakukan yaitu :

1. *First Normal Form* (1 NF)

   Sudah tidak ada repeating group yaitu pengulangan yang terjadi pada beberapa atribut atau kolom dalam sebuah tabel, dan juga setiap atribut harus bernilai tunggal. Atribut *multivalued*, *composite*, *derive* tidak tunggal. Setiap nilai dari atribut hanya mempunyai nilai tunggal.

2. *Second Normal Form* (2 NF)

   Untuk menjadikan tabel normal tingkat ke 2 maka sudah 1NF dan setiap atribut yang bukan *primary key* sepenuhnya secara fungsional tergantung pada semua atribut pembentuk *primary key*.

3. *Third Normal Form* (3 NF)

   Tabel sudah 2NF dan tidak memiliki *transitive dependencies*, *Transitive Dependencies* adalah ketika atribut yanng secara tidak langsung tergantung pada *primary key* dan atribut tersebut juga tergantung pada atribut lain yang bukan *primary key*.

4. *Boyce-codd Normal Form* (BCNF)

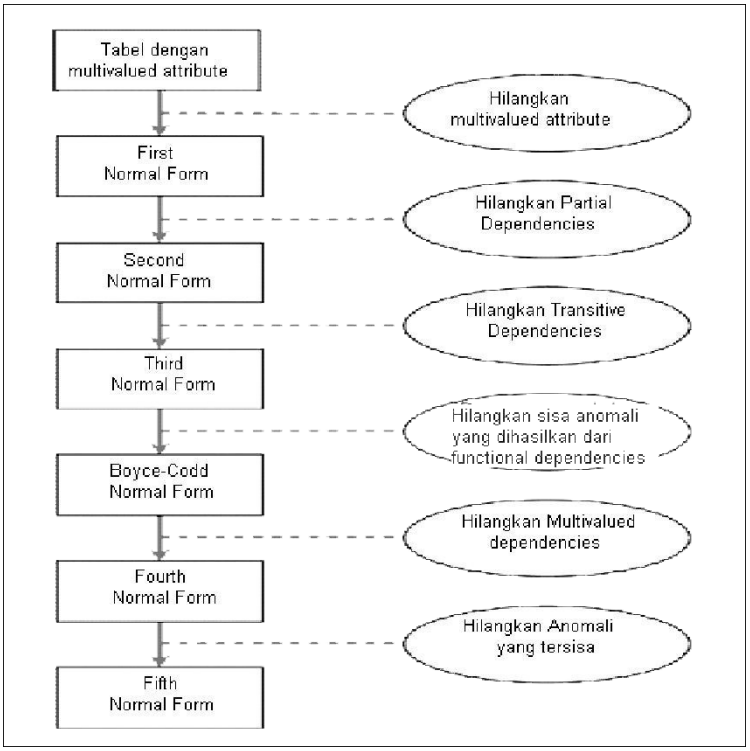
   Tabel dalam BCNF jika sudah 3NF dan semuah *determinants* adalah *candidate keys*. Perbedaan 3NF dan BCNF adalah untuk *functional dependency* A – B, 3NF memperbolehkan ketergantungan ada dalam relasi juka B adalah *primary key* dan A bukan merupakan *candidate key*. Sedangkan BCNF menuntuk untuk ketergantungan tetap ada dalam relasi, A harus menjadi *candidate key*.

5. *Fourt Normal Form* (4 NF)

   Relasi berada pada pentuk normal keempat apabila memenuhi syarat BCNF dan tidak mempunyai multivalue d*ependency*.

6. *Fifth Normal Form* (5 NF)

   Tabel bentuk normal kelima sering disebut PJNF (P*rojection Join Normal Form*), Penyebutan PJNF karena untuk suatu relasi akan berbentuk normal kelima jika tabel tersebut dapat dipecah atau diprogersikan (kemajuan) menjadi beberapa tabel dan dari progersi (kemajuan) dapat disusun kembali (*join*) menjadi tabel yang sama dengan keadaan semula. Jika penyusunan ini tidak mungkin dilakukan dikatakan pada relasi itu terdapat *join dependencies* dan dikatakan bersifat *lossy join*.(Gandung Triyono : 2013 : 19-20)



Gambar II.3. Langkah-Langkah Dalam Pernomalan Tabel
(Sumber: Gandung Triyono; 2013: 20)