BAB II

TINJAUAN PUSTAKA

II.1. Sistem

Secara sederhana, suatu sistem dapat diartikan sebagai suatu kumpulan atau himpunan dari unsur, komponen, atau *variable* yang terorganisir, saling berinteraksi, saling tergantung satu sama lain dan terpadu. Teori sistem secara umum pertama sekali diuraikan oleh Kenneth Boulding, terutama menekankan pentingnya perhatian terhadap setiap bagian yang membentuk sebuah sistem. Teori sistem mengatakan bahwa setiap unsur pembentuk organisasi itu penting dan harus mendapat perhatian yang utuh supaya manajer dapat bertindak lebih efektif. Yang dimaksud unsur atau komponen pembentuk organisasi disini bukan hanya bagian bagian yang tampak secara fisik, tetapi juga hal-hal yang mungkin bersifat abstrak atau konseptual, seperti misi, pekerjaan, kegiatan, kelompok informal, dan lain sebagainya. (Tata Sutabri : 2012 : 3).

II.2. Keputusan

Keputusan merupakan kegiatan memilih suatu strategi atau tindakan dalam pemecahan masalah tersebut. Tindakan memilih strategi atau aksi yang diyakini manajer akan memberikan solusi terbaik atas sesuatu itu disebut pengambilan keputusan. Tujuan dari keputusan adalah untuk mencapai target atau aksi tertentu yang harus dilakukan. (Kusrini : 2007 : 7).

Kriteria atau ciri-ciri dari keputusan adalah :

1. Banyak pilihan/alternatif.
2. Ada kendala atau syarat
3. Mengikuti suatu pola/model tingkah laku, baik yang terstuktur maupun tidak terstruktur.
4. Banyak input/variabel.
5. Ada faktor risiko.
6. Dibutuhkan kecepatan, ketepatan, dan keakuratan. (Kusrini : 2007 : 7).

Tahap-tahap pembuatan keputusan :

1. Identifikasi masalah.
2. Pemilihan metode pemecahan masalah.
3. Pengumpulan data yang dibutuhkan untuk melaksanakan model keputusan tersebut.
4. Mengevaluasi sisi positif dari setiap alternatif yang ada.
5. Melaksanakan solusi terpilih. (Kusrini: 2007 : 9).

II.3. Sistem Pendukung Keputusan

II.3.1. Pengertian Sistem Pendukung Keputusan

Sistem pendukung keputusan (*decision support system/DSS*) dapat didefinisikan sebagai suatu sistem informasi yang membantu mengidentifikasi kesempatan membuat keputusan atau menyediakan informasi untuk membantu pembuatan keputusan.

Menurut Turban, sistem pendukung keputusan dapat dibagi menjadi beberapa subsistem :

1. Subsistem Manajemen Data

   Subsistem menajemen data memasukkan satu database yang berisi data yang relevan untuk situasi dan dikelola oleh perangkat lunak yang disebut sistem manajemen basis data (DBMS).

2. Subsistem Manajemen Model

   Subsistem ini terdiri atas Basis Model (*Model base*), Sistem Manajemen Basis Model (*Model Base Management System*), Bahasa Pemodelan, Direktori Model, Eksekusi Model, Integrasi dan Prosesor Perintah.

3. Subsistem Antarmuka Pengguna

   Pengguna berkomunikasi dengan dan memerintahkan sistem pendukung keputusan melalui subsistem ini. Pengguna adalah bagian yang dipertimbangkan dari sistem. Para peneliti menegaskan bahwa beberapa kontribusi unik dari sistem ini berasal dari interaksi yang intensif antara komputer dan pembuat keputusan.

4. Subsistem Manajemen Berbasis Pengetahuan

   Subsistem ini dapat mendukung semua subsistem lain atau bertindak sebagai suatu komponen indepeden. Ia memberikan intelegensi untuk memperbesar pengetahuan si pengambil keputusan. Subsistem ini dapat di interkoneksikan dengan repositori pengetahuan perusahaan, yang kadang-kadang disebut basis pengetahuan organisasional.(Winnie Septiani : 57).

II.3.2. Tujuan Dari Sistem Pendukung Keputusan

1. Membantu menejer dalam pengambilan keputusan atas masalah semi terstruktur.
2. Memberikan dukungan atas pertimbangan manajer dan bukannya dimaksudkan untuk menggantikan fungsi manajer.
3. Meningkatkan evektivitas keputusan yang diambil manajer lebih dari pada perbaikan efisiensinya.
4. Kecepatan komputasi. Computer memungkinkan para pengambil keputusan untuk melakukan banyak komputasi secara cepat dengan biaya yang rendah.
5. Peningkatan produktivitas. Membangun satu kelompok pengambil keputusan, terutama para pakar, bias sangat mahal. Pendukung terkomputerisasi bias mengurangi ukuran kelompok dan memungkinkan para anggotanya untuk berada diberbagai lokasi yang erbeda0beda (menghemat biaya perjalanan).
6. Dukungan kualitas. komputer bias meningkatkan kualitas komputer yang dibuat.
7. Berdaya saing. Teknologi pengambilan keputusan bias menciptakan pemberdayaan yang signifikan dengan cara memperbolehkan seseorang untuk membuat keputusan yang baik secara cepat, bahkan jika mereka memiliki pengetahuan yang kurang.
8. Mengatasi keterbatasan kognitif dalam pemrosesan dan penyimpanan. (Kusrini : 2007 : 16)

II.4. Metode Analytical Hierarchy Process (AHP)

II.4.1. Pengertian AHP

Menurut Syaipullah dalam naskah internetnya yang berjudul Pengenalan Metode AHP (*Analytical Hierarchy Process*) menyatakan AHP merupakan suatu model pendukung keputusan yang dikembangkan oleh Thomas L. Saaty. Model pendukung keputusan ini akan menguraikan masalah multi faktor atau multi kriteria yang kompleks menjadi suatu hirarki, menurut Saaty (1993), hirarki didefinisikan sebagai suatu representasi dari sebuah permasalahan yang kompleks dalam suatu struktur multi *level* dimana *level* pertama adalah tujuan, yang diikuti level faktor, kriteria, sub kriteria, dan seterusnya ke bawah hingga level terakhir dari alternatif. Dengan hirarki, suatu masalah yang kompleks dapat diuraikan ke dalam kelompok-kelompoknya yang kemudian diatur menjadi suatu bentuk hirarki sehingga permasalahan akan tampak lebih terstruktur dan sistematis. AHP sering digunakan sebagai metode pemecahan masalah disbanding dengan metode yang lain karena alasan-alasan sebagai berikut:

a. Struktur yang berhirarki, sebagai konsekuesi dari kriteria yang dipilih, sampai pada subkriteria yang paling dalam.
b. Memperhitungkan validitas sampai dengan batas toleransi inkonsistensi berbagai kriteria dan alternatif yang dipilih oleh pengambil keputusan.
c. Memperhitungkan daya tahan *output* analisis sensitivitas pengambilan keputusan. (Syaifullah : 2010).

II.4.2. Prinsip Dasar AHP

Dalam menyelesaikan permasalahan dengan AHP ada beberapa prinsip yang harus dipahami, diantaranya adalah :

1. Membuat Hierarki

   Sistem yang kompleks bisa dipahami dengan memecahnya menjadi elemen-elemen pendukung. menyusun elemen seara hirarki, dan menggabungkannya atau mensintesisnya.

2. Penilaian Kriteria dan Alternatif

   Kriteria dan alternatif dilakukan dengan perbandingan berpasangan. Menurut Saaty (1988), untuk berbagi persoalan, skala 1 sampai 9 adalah skala terbaik untuk mengekspresikan pendapat. nilai dan definisi pendapat kualitatif dari skala perbandingan Saaty bisa diukur menggunakan tabel analisis.

3. *Synthesis of priority* (menentukan prioritas)

   Untuk setiap kriteria dan alternatif , perlu dilakaukan perbandingan berpasangan (*Pairwise Comparison*). Nilai-nilai perbandingan relatif dari seluruh alternatif kriteria bisa disesuaikan dengan *judgement* yang telah ditentukan untuk menghasilkan bobot dan prioritas.

4. *Logical Consistency* (Konsistensi Logis)

   Konsistensi memiliki dua makna. Pertama, objek-objek yang serupa bisa dikelompokkan sesuai dengan keseragaman dan relevansi. Kedua, menyangkut tingkat hubungan antar objek yang didasarkan pada kriteria tertentu. (Kusrini : 2007 : 133).

II.4.3. Prosedur AHP

Pada dasarnya, prosedur atau langkah-langkah dalam metode AHP meliputi :

1. Mendefinisikan masalah dan menentukan solusi yang diinginkan, lalu menyusun hierarki dari permasalahan yang dihadapi. Penyusunan hierarki adalah dengan menetapkan tujuan yang merupakan sasaran sistem secara keseluruhan pada level teratas.
2. Menentukan prioritas elemen
   - Langkah pertama dalam menentukan prioritas elemen adalah membuat perbandingan pasangan, yaitu membandingkan elemen secara berpasangan sesuai kriteria yang diberikan.
   - Matriks perbandingan berpasangan diisi menggunakan bilangan untuk merepresentasikan kepentingan relatif dari suatu elemen terhadap elemen yang lainnya.
3. Sintesis

   Pertimbangan-pertimbangan terhadap perbandingan berpasangan di sintesis untuk memperoleh keseluruhan prioritas. Hal-hal yang dilakukan dalam langkah ini adalah :
   - Menjumlah nilai-nilai dari setiap kolom pada matriks.
   - Membagi setiap nilai dari kolom dengan total kolom yang bersangkutan untuk memperoleh normalisasi matriks.
   - Menjumlahkan nilai-nilai dari setiap baris dan membaginya dengan jumlah elemen untuk mendapatkan nilai rata-rata.

4. Mengukur Konsistensi

   Dalam pembuatan keputusan, penting untuk mengetahui seberapa baik konsistensi yang ada karena kita tidak menginginkan keputusan berdasarkan pertimbangan dengan konsistensi yang rendah. Hal-hal yang dilakukan dalam langkah ini adalah :

   - Kalikan setiap nilai pada kolom pertama dengan prioritas relative elemen pertama, nilai pada kolom kedua dengan prioritas relatif elemen kedua, dan seterusnya.
   - Jumlah setiap baris.
   - Hasil dari penjumlahan baris dibagi dengan elemen prioritas relatif yang bersangkutan.
   - Jumlahkan hasil bagi di atas dengan banyaknya elemen yang ada, hasil disebut λ maks.

5. Hitung *Consistency Index* (CI) dengan rumus :

$$CI = (\lambda \text{ maks-n})/n \text{ …….. } 1$$

   Dimana :

   - n = banyaknya elemen

6. Hitung Rasio Konsistensi/*Consistency Ratio* (CR) dengan rumus :

$$CR = CI/IR \text{……. } 2$$

   Dimana :

   - CR = *Consistency Ratio*
   - CI = *Consistency Index*
   - IR = *Indeks Random Consistency*

7. Memerikasa konsistensi hierarki. Jika nilainya lebih dari 10%, maka penilain data judgment harus diperbaiki. Namun jika rasio konsistensi (CI/IR) kurang atau sama dengan 0,1, maka hasil perhitungan bisa dinyatakan benar. (Kusrini 2007 : 135).

II.5. *Visual Basic.Net*

Pemprograman *Microsoft Visual Basic .NET* adalah bahasa pemrograman yang dikembangkan oleh perusahaan Microsoft. *Visual Basic.NET* merupakan pengembangan dari versi sebelumnya. yaitu *Visual Basic* 6.0, yang memiliki karakteristik mudah untuk dipahami, namun andal dalam mengikuti tren teknologi perangkat lunak. Perbedaan mendasar antara *Visual Basic.NET* dengan versi-versi sebelumnya adalah kemampuan OOP (*Object Oriented programming*) yang telah ditanamkan pada *Visual Basic.NET*. saat ini *Visual Basic.NET* telah dikolaborasikan dengan beberapa jenis aplikasi, seperti aplikasi desktop dan aplikasi berbasis web. (Alexander F.K Sibero : 2010).

II.6. *SQL Server*

SQL Server adalah sebuah RDBMS (*Relational Database Management System*) yang dikembangkan oleh Microsoft. *SQL Server* dapat digunakan dalam basis data untuk kebutuhan personal, seperti basis data pada aplikasi *Windows Mobile Device* serta aplikasi yang menggunakan basis data dengan banyak *server*. *SQL Server* terdiri dari beberapa versi antara lain *Express Edition, Personal Edition, dan Enterprise Edition*. Microsoft juga terus mengembangkan

platformnya untuk dapat menghasilkan kolaborasi sistem yang andal. (Alexander F.K Sibero : 2010).

II.7. *DataBase*

Database adalah kumpulan data yang saling terkait yang diorganisasi untuk memenuhi kebutuhan dan struktur sebuah organisasi serta bisa digunakan lebih dari satu orang dan lebih dari satu aplikasi.

Ada 3 sumber data dalam system pendukung keputusan, yaitu :

1. *Data Internal*

   Data internal yang dimaksud adalah data yang sudah ada dalam suatu organisasi. Data tersebut bisa dikendalikan oleh organisasi tersebut.

2. *Data External*

   Data external adalah data yang tidak bisa dikendalikanoleh organisasi. Data tersebut berasal dari luar system.

3. *Data Private/Personal*

   Data private adalah data mengenai kepakaran/naluri dari user terhadap masalah yang akan diselesaikan. Dengan kata lain, *data private* merupakan pendapat dari user mengenai variabel yang diperlukan dalam menyelesaikan masalah atau nilai dari suatu variabel. Data tersebut bersifat subjektif. (Kusrini : 2007 : 33).

II.8. UML (*Unified Modelling Language*)

Unified Modeling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merencang dan mendokumentasikan sistem piranti lunak. UML menawarkan sejumlah standar untuk merancang sebuah sistem. Dengan menggunakan UML kita dapat membuat model untuk aplikasi piranti lunak, dimana aplikasi tersebut dapat berjalan pada piranti keras, sistem operasi dan jaringan apapun, serta ditulis dalam pemrogaman apapun. Tetapi karena UML juga menggunakan *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 prosedur dalam VB atau C.

Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantic. Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak. Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan. (Yuni Sugiarti : 2013 : 34).

II.8.1. *Use Case Diagram*

Use Case Diagram merupakan pemodelan untuk menggambarkan kelakuan (*behavior*) sistem yang akan dibuat. *Use Case Diagram* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem yang akan dibuat. Dengan pengertian yang cepat, *Use Case Diagram* digunakan untuk mengetahui fungsi apa saja yang ada didalam sebuah sistem dan siapa saja yang

berhak menggunakan fungsi-fungsi tersebut. Terdapat beberapa simbol dalam menggambarkan *Use Case Diagram* yaitu *Use Case*, Aktor dan relasi. Hal yang perlu diingat dalam *Use Case Diagram* adalah bukan menggambarkan tampilan antarmuka (*user interface*), arsitektur dari sistem, kebutuhan non fungsional, dan tujuan performansi, sedangkan untuk penamaan *Use Case* adalah nama didefinisikan sesimpel mungkin, dapat dipahami dan menggunakan kata kerja. *Use Case Diagram* adalah sebuah diagram yang menjelaskan apa yang harus dilakukan oleh sistem pada level konseptual sehingga kita akan memahami apakah keputusan yang diambil oleh sistem adalah benar atau tidak. (Yuni Sugiarti : 2013 : 41).

II.8.2. *Class Diagram*

Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. Kelas memiliki apa yang disebut atribut dan metode atau operasi.

1. Atribut merupakan variabel-variabel yang di miliki oleh suatu kelas.
2. Atribut mendeskripsikan properti dengan sebaris teks di dalam kotak kelas tersebut.
3. Operasi atau metode adalah fungsi – fungsi yang di miliki oleh suatu kelas.

Diagram kelas mendeskripsikan jenis-jenis objek dalam sistem dan berbagai hubungan statis yang terdapat diantara mereka. Diagram kelas juga menunjukkan properti dan operasi sebuah kelas dan batasan-batasan yang terdapat dalam hubungan-hubungan objek tersebut. (Yuni Sugiarti : 2013 : 57).

II.8.3. *Sequence Diagram*

Diagram sekuence menggambarkan kelakuan/prilaku objek pada *use case* dengan mendeskripsikan waktu hidup objek dan *message* yang dikirimkan dan diterima antar objek. Oleh karena itu untuk menggambarkan diagram sekuence maka harus diketahui objek-objek yang terlihat dalam sebuah *use case* beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

Banyaknya diagram sekuence yang harus digambar adalah sebanyak pendefinisian *use case* yang memiliki proses sendiri atau yang penting semua *use case* yang telah didefinisikan interaksi jalanya pesan sudah dicakup pada diagram sekuence sehingga menjadi banyak *use case* yang didefinisikan maka diagram sekuence yang harus dibuat juga harus semakin banyak. (Yuni Sugiarti : 2013 : 70).

II.8.4. *Activity Diagram*

Activity diagram menggambarkan *work flow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Yang perlu diperhatikan disini adalah bahwa diagram aktivitas menggambarkan aktivitas sistem bukan apa yang dilakukan aktor, jadi aktivitas yang dapat dilakukan oleh sistem. Diagram kelas mendukung prilaku pararel.

Diagram aktivitas juga banyak digunakan untuk mendefinisikan hal-hal sebagai berikut :

1. Rancangan proses bisnis dimana setiap urutan aktivitas yang digambarkan merupakan proses bisnis sistem yang didefinisikan.

2. Urutan atau pengelompokkan tampilan dari sistem/*user interface* dimana setiap aktivitas diaggap memiliki sebuah rancangan antar muka tampilan.
3. Rancangan pengujian dimana setiap aktivitas diaggap memerlukan sebuah pengujian yang perlu didefinisikan kasus ujiannya.

Activity Diagram menggambarkan berbagai aliran aktivitas dalam sistem yang sedang dirancang bagaimana masing-masing alir berawal, *decision* yang mungkin terjadi, dan bagaimana mereka berakhir. *Activity Diagram* juga dapat menggambarkan proses pararel yang mungkin terjadi pada beberapa eksekusi. *Activity Diagram* merupakan *state* diagram khusus, dimana sebagian besar *state* adalah *action* dan sebagian besar transisi di-*trigger* oleh selesainya *state* sebelumnya (*internal processing*). Oleh karena itu Activity Diagram tidak menggambarkan *behavior* internal sebuah sistem (dan interaksi antar subsistem). Secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. (Yuni Sugiarti : 2013 : 75).

II.9. ERD (*Entity Relationship Diagram*)

ERD adalah suatu model jaringan yang menggunakan susunan data yang disimpan dalam system secara abstrak. Jadi, jelaslah ERD ini berbeda dengan DFD yang merupakan suatu model jaringan fungsi yang akan dilaksanakan oleh sistem, sedangan ERD merupakan model jaringan data yang menekankan pada struktur-struktur dan *relationship* data. (Al-Bahra Bin Ladjamuddin. B : 2004 : 123).

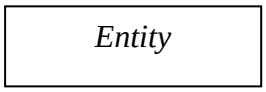
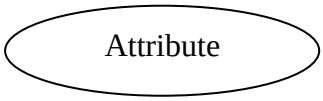
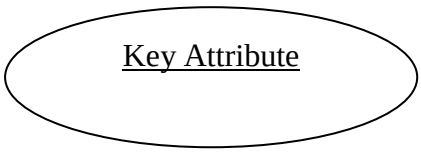
Biasanya ERD ini digunakan oleh profesional sistem untuk berkomunikasi dengan pemakai eksekutif tingkat tinggi dalam suatu organisasi. Pemakai ini lebih terrarik dengan hal-hal sebagai berikut :

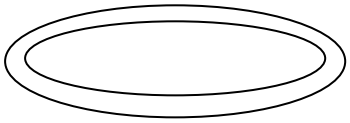
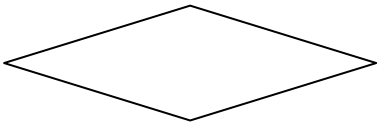
1. Data apa saja yang dibutuhkan untuk bisnis mereka ?
2. Bagaimana data tersebut berelasi dengan data lainnya ?
3. Siapa saja yang diperkenankan untuk mengakses data tersebut ? (Al-Bahra Bin Ladjamuddin. B : 2004 : 124).

II.9.1. Simbol-simbol ERD (*Entity Relationship Diagram*)

Adapaun simbol-simbol ERD (*Entity Relationship Diagram*) ditunjukkan pada tabel II.1.

Tabel II.1. Simbol-simbol ERD (*Entity Relationship Diagram*)

Simbol	**Deskripsi**
Entitas/*Entity* 	Entitas merupakan data inti yang akan disimpan; bakal tabel pada basis data
Atribut/*Attribute* 	*Attribute* adalah sifat-sifat atau karakteristik dari suatu entitas.
Atribut kunci/*Key Attribute* 	Suatu *key* attribute adalah unik, dan memiliki karakteristik pembeda dari entitas. Sebagai contoh, nomor mahasiswa mungkin menjadi

	key *attribute* mahasiswa.
Multivalued Attribute 	Suatu *multivalued attribute* memiliki lebih dari satu nilai. Sebagai contoh, suatu entitas pegawai bisa memiliki nilai pada berbagai keahlian.
Relationship 	Relationship mengilustrasikan bagaimana dua entitas berbagi informasi didalam struktur basis data.

Sumber: (Janner Simarmata : 2007 : 112)

II.10. Normalisasi

Proses normalisasi pertama kali diperkenalkan oleh E.F. Codd pada tahun 1972. Normalisasi sering dilakukan sebagai suatu uji coba pada suatu relasi secara berkelanjutan untuk menentukan apakahrelasi tersebut sudah baik atau masih melanggar aturan-aturan standar yang diberlakukan pada suatu relasi yang normal.

Proses normalisasi merupakan metode yang formal/standar dalam mengidentifikasi standar relasi bagi *primary key*. Dan dependensi fungsional antara atribut-atribut dari relasi tersebut. Normalisasi akan membantu perancang basis data dengan menyediakan suatu uji coba berurut yang dapat diimplementasikan pada hubungan individual sehingga skema relasi dapat dinormalisasi kedalam bentuk yang lebih spesifik untuk menghindari terjadinya

error atau inkonsistensi data, bila dilakukan *update* terhadap relasi tersebut dengan *anomaly*. (Al-Bahra Bin Ladjamuddin. B : 2004 : 173).

Tahapan normalisasi terdiri dari beberapa bentuk yaitu sebagai berikut:

1. Bentuk Normal Pertama (1NF/*First Normal Form*)

Pada tahap ini dilakukan penghilangan beberapa group elemen yang berulang agar menjadi satu harga tunggal yang berinteraksi diantara setiap baris pada suatu tabel, dan setiap atribut harus mempunyai nilai data yang *automatic*.

2. Bentuk Normal Kedua (2NF/*Second Normal Form*)

Bentuk normal kedua didasari atas konsep *full function dependency* (ketergantungan fungsional sepenuhnya). Bentuk normal kedua memungkinkan suatu relasi memiliki *composite key*.

3. Bentuk Normal Ketiga (3NF/ *Third Normal Form*)

Atribut bukan kunci haruslah tidak memiliki ketergantungan transistif, dengan kata lain suatu atribut bukan kunci (*non-key*) tidak boleh memiliki ketergantungan fungsional terhadap atribut bukan kunci lainnya, seluruh atribut bukan kunci pada suatu relasi hanya memiliki ketergantungan fungsional terhadap *primary key* di relasi itu saja.

4. Bentuk Normal B*oyce Codd* (BCNF/*Boyce Codd Normal Form*)

Boyce-Codd normal form didasari pada beberapa ketergantungan fungsional dalam suatu relasi yang melibatkan seluruh *candidate key* didalam relasi tersebut. Jika suatu relasi hanya memiliki satu *candidate key*, maka hasil uji normalisai sampai kebenuk normal ketiga sudah identik dengan *Boyce-Codd Normal Form* (BNCF). (Al-Bahra Bin Ladjamuddin. B : 2004 : 184).