# BAB II

# LANDASAN TEORI

Bab ini menguraikan teori-teori yang digunakan dalam pembuatan Tugas Akhir dengan judul Rancang Bangun Sistem Analisa Manajemen Resiko Pengajuan Kredit Bank Bagi Pegawai Negeri Menggunakan Metode Fuzzy Logic, antara lain meliputi:

## 2.1 Kredit

Pengertian kredit mempunyai dimensi yang beraneka ragam, dimulai dari kata kredit yang berasal dari bahasa latin ”credo”, yang berarti ”saya percaya”, yang merupakan kombinasi dari bahasa sansekerta ”cred” (yang artinya ”kepercayaan”) dan bahasa latin ”do” (yang artinya ”saya tempatkan”). Pengertian kredit juga berasal dari kata ”Kredit” yang berasal dari bahasa Yunani ”Credere” yang berarti ”kepercayaan” atau dalam bahasa latin ”Creditum” yang berarti kepercayaan akan kebenaran. Ada beberapa pengertian yang diambil dari arti kata tentang kredit diatas antara lain:

a. Kredit adalah kemampuan untuk melaksanakan suatu pembelian atau mengadakan suatu pinjaman dengan suatu janji pembayarannya akan dilakukan dengan ditangguhkan pada suatu jangka waktu tertentu yang telah disepakati (IKAPI, 1988).
b. Sedangkan pengertian yang lebih mapan untuk kegiatan perbankan di Indonesia, juga dirumuskan dalam bab I, pasal 1, 2 Undang-Undang Pokok

Perbankan No. 14 tahun 1967 yang merumuskan: “Kredit adalah penyediaan uang atau tagihan yang dapat disamakan dengan itu, berdasarkan persetujuan pinjam-meminjam antara bank dengan pihak lain dalam hal dimana peminjam berkewajiban melunasi hutangnya setelah jangka waktu tertentu dengan jumlah bunga yang telah ditentukan” (IKAPI, 1988).

## 2.2 Analisa kredit

Dalam pelaksanaan pemberian kredit, dihadapkan pada suatu masalah yang cukup kompleks yang antara lain: (a) Untuk (obyek) apa kredit itu harus diberikan?, (b) Apakah yang mengajukan kredit kiranya akan mampu untuk mengembalikannya?, (c) Berapa jumlah maksimum kredit yang layak untuk diberikan?, (d) Berapa jangka waktu pengembalian kredit?, dan (e) Dan lain sebagainya. Beberapa hal yang harus dilakukan sebelum melaksanakan kegiatan analisa kredit yaitu membahas aspek-aspek yang mempengaruhi yaitu:

a. Pemilihan pendekatan yang akan digunakan dalam melakukan analisa kredit itu sendiri yang salah satunya adalah pendekatan yang mendasarkan diri dari kemampuan pelunasan atas kredit yang diberikan (*repayment approach*).
b. Proses pengumpulan informasi yang lengkap yang akan diperlukan dalam kegiatan suatu analisa kredit yang antara lain mengenai data/informasi informal/formal yang menyangkut soal reputasi karakter yang mengajukan kredit.

## 2.3 Fuzzy Logic

Untuk menghitung gradasi yang tidak terbatas jumlahnya antara benar dan salah, Lotfi A. Zadeh mengembangkan ide penggolongan himpunan (*set*) yang ia namakan himpunan *fuzzy* (*fuzzy set*). Tidak seperti logika *boolean* yang menyatakan bahwa suatu pernyataan adalah benar atau salah, *fuzzy logic* dapat membaginya dalam derajat keanggotaan dan derajat kebenaran sehingga suatu pernyataan dapat menjadi sebagian benar dan sebagian salah pada waktu yang sama.

### 2.3.1 Konsep Utama Fuzzy

#### A Prinsip ketidakpastian

Beberapa ilmu matematika terkadang sulit untuk dipastikan, seperti teori probabilitas. Hal ini bisa diklasifikasikan berdasar tipe ketidakpastian yang dilakukan. Ada beberapa tipe ketidakpastian, dua diantaranya adalah *Stochastic Uncertainty* dan *Lexical Uncertainty*.

*Stochastic Uncertainty* berhubungan dengan arah ketidakpastian dari kejadian yang pasti. Sedangkan *Lexical Uncertainty* merupakan ketidakpastian yang diungkapkan oleh kata-kata manusia, seperti “orang yang tinggi”, “hari yang panas” dan sebagainya.

#### B Himpunan fuzzy (*Fuzzy Sets*)

Himpunan *fuzzy* terdiri atas 3 bagian, sumbu horisontal menunjukkan himpunan anggota, sumbu vertikal menunjukkan derajat dari keanggotaan, dan

garis yang menghubungkan masing-masing titik dari anggota dengan derajat keanggotaan yang tepat. Himpunan *fuzzy* akan dibahas lebih lanjut pada bagian operasi himpunan *fuzzy*

## C Fungsi keanggotaan

Derajat dimana angka teknis bernilai sesuai konsep bahasa dari kondisi variabel bahasa (*linguistic*) dinamakan sebagai derajat keanggotaan. Untuk variabel berlanjut (*continues variable*) derajat ini disebut fungsi keanggotaan.

## D Variabel linguistik

*Fuzzy* pada dasarnya menitikberatkan pada pengukuran dan penalaran tentang kekaburan atau bentuk *fuzzy* yang nampak dalam bahasa alami. Dalam *fuzzy* bentuk *fuzzy* dinyatakan sebagai variabel linguistik (disebut juga variabel *fuzzy*).

Variabel linguistik adalah bentuk yang digunakan dalam bahasa alami untuk menggambarkan beberapa konsep yang biasanya mempunyai kekaburan atau nilai *fuzzy*. Sebagai contoh dalam pernyataan “Jack adalah muda” menyatakan bahwa variabel linguistik umur mempunyai nilai linguistik muda.

Seperti halnya variabel aljabar yang berisi angka sebagai nilainya maka vaiabel linguistik menggunakan kata dan kalimat sebagai nilainya. Misalnya: jika T variabel linguistik yang berisi himpunan umur, maka isi T yang juga merupakan himpunan fuzzy adalah: *T = {sangat tua, tua , setengah baya, agak muda, muda, sangat muda}*

**E Aturan fuzzy**

Aturan dari sistem *fuzzy* (*fuzzy system*) menggambarkan pengetahuan dari sistem. Mereka menggunakan variabel linguistik sebagai bahasanya, sebagai contoh untuk mengekspresikan strategi pengendalian dari sebuah pengendali pengontrol *fuzzy*. Menjelaskan aturan *fuzzy* berarti menunjukkan, bagaimana menghitung dengan konsep linguistik.

### 2.3.2 Perhitungan fuzzy

Dalam aplikasi kontrol, komputasi *fuzzy* terdiri atas:

**A Fuzzyfikasi**

Fuzzyfikasi berarti menggunakan fungsi keanggotaan dari variabel linguistik untuk menghitung masing-masing derajat kondisi validitas dari angka-angka spesifik proses. Fuzzyfikasi yang mentransformasi masukan himpunan klasik (*crisp*) ke derajat tertentu yang sesuai dengan aturan besaran fungsi keanggotaan.

**B Inferensi fuzzy**

Dalam inferensi *fuzzy* dilakukan proses yang dinamakan evaluasi rule. Tahap ini digunakan untuk mencari derajat kebenaran (*rule strength*) dari masukan *fuzzy* yang nilai keanggotaannya telah ditentukan sebelumnya pada proses fuzzyfikasi. Struktur dasar dari sistem inferensi *fuzzy* terdiri dari basis aturan yang berisi aturan *if-then*, basis data yang mendefinisikan fungsi keanggotaan dari himpunan *fuzzy*.

**C Defuzzyfikasi**

Defuzzyfikasi yang mentransformasi hasil fuzzy ke bentuk keluaran yang *crisp*. Fungsi keanggotaan digunakan dalam menerjemahkan keluaran fuzzy ke bentuk keluaran *crisp*. Terjemahan kembali dalam defuzzyfikasi dapat menggunakan beberapa metode, dimana diantaranya adalah Center of Maximum (CoM)/Center of Gravity (CoG), Center of Area (CoA), Mean of Maximum (MoM), Mean of Maximum Bounded Sum (MoM BSUM).

### 2.3.3 Himpunan fuzzy

Teori himpunan tradisional menggambarkan dunia sebagai hitam dan putih. Ini berarti sebuah obyek berada didalam atau diluar himpunan yang diberikan. Dalam teori himpunan tradisional untuk anggota diberi nilai 1 dan untuk bukan anggota diberi nilai 0; ini disebut himpunan *crisp*. Sebagai contoh anggota himpunan orang muda dapat berisi hanya orang yang berumur kurang dari 10. Penggunaan interpretasi ini pada seseorang yang berulang tahun ke-11, maka orang tersebut bukan anggota himpunan orang muda.

Himpunan *fuzzy* memberikan nilai keanggotaan antara 0 dan 1 yang menggambarkan secara lebih alami sebuah kumpulan anggota dengan himpunan. Sebagai contoh, jika seorang berumur 5 tahun dapat diberikan nilai keanggotaan 0.9 atau jika umurnya 13 tahun nilai keanggotaannya 0.1. Dalam contoh ini “umur” adalah variabel linguistik dan “muda” adalah salah satu himpunan *fuzzy*.

Himpunan *fuzzy* dapat didefinisikan sebagai berikut: misalkan X semesta pembicaraan, dengan elemen dari X dinotasikan x. Sebuah himpunan *fuzzy* A dari X dikarakteristikkan dengan fungsi keanggotaan $\mu \wedge (x) : X \rightarrow [0,1]$

Pada *fuzzy* , kejadian atau elemen x diberikan nilai keanggotaan dengan fungsi keanggotaan $\mu$. Nilai ini mempresentasikan derajat keanggotaan elemen x pada himpunan *fuzzy* A. $\mu_{\wedge}$ (x) = Degree($x \in A$). Nilai keanggotaan dari x berada pada interval : $0 \leq \mu_{\wedge}(x) \leq 1$. Himpunan *fuzzy* adalah perluasan dari teori himpunan tradisional. Himpunan *fuzzy* menyamakan konsep keanggotaan dengan menggunakan fungsi keanggotaan $\mu$ yang menghasilkan nilai antara 0 dan 1 yang mempresentasikan derajat keanggotaan obyek x pada himpunan A.

Untuk mempresentasikan himpunan *fuzzy* dalam komputer perlu didefinisikan fungsi keanggotaannya. Sebagai contoh: orang tinggi. Dapat dinyatakan pada setiap individu, pada tingkatan mana bahwa mereka yakin seseorang itu dikatakan tinggi. Setelah mengumpulkan jawaban untuk interval ukuran tinggi, dapat disajikan tingkat rata-rata untuk menghasilkan suatu himpunan *fuzzy* dari orang-orang yang tinggi. Fungsi ini dapat digunakan sebagai suatu keyakinan (nilai keanggotaan). Bagi individu yang menjadi anggota himpunan *fuzzy* dari orang tinggi.

Dengan membentuk *fuzzy* subset untuk berbagai bentuk *fuzzy*, dianggap nilai keanggotaan dari obyek yang diberikan pada setiap himpunan. Pendekatan lain yang sering ditemukan pada praktek untuk membentuk himpunan *fuzzy* sangat berhubungan dengan interpretasi dari seorang ahli. Seperti teknik pengumpulan data, dapat ditanyakan pada pakar untuk

kepercayaannya bahwa berbagai obyek merupakan bagian himpunan yang diberikan.

### 2.3.4 Operasi himpunan fuzzy

Terdapat 3 operasi dalam himpunan *fuzzy*, yaitu:

a. Irisan *(Intersection)*

Dalam teori himpunan klasik, irisan dari dua himpunan berisi elemen-elemen yang sama dari keduanya. Dalam himpunan *fuzzy*, sebuah elemen mungkin sebagian dalam kedua himpunan. Oleh karena itu ketika mengingat irisan dari kedua himpunan, tidak dapat dikatakan bahwa sebuah elemen adalah lebih mungkin menjadi dalam irisan daripada dalam suatu himpunan asli.

b. Gabungan (*Union*)

Cara kedua dari penggabungan himpunan *fuzzy* adalah gabungannya. Penggabungan dari dua himpunan adalah terdiri dari dua himpunan adalah terdiri dari elemen-elemen yang menjadi satu atau dua himpunan. Dalam situasi ini anggota dari gabungan tidak dapat mempunyai nilai keanggotaan yang kurang dari nilai keanggotaan yang lain dari himpunan aslinya.

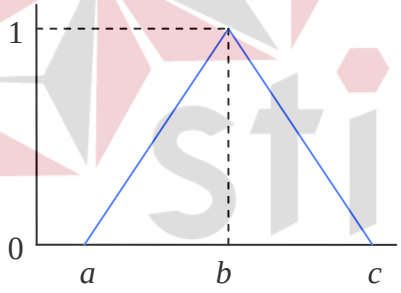
c. Komplemen (*Complement*)

Komplemen dari himpunan *fuzzy* A dinotasikan dengan (~A) dinyatakan dengan persamaan sebagai berikut : $\mu_{\sim A}(x) = 1 - \mu_A(x)$

### 2.3.5 Fungsi keanggotaan

Fungsi keanggotaan digunakan dalam mempresentasikan himpunan *fuzzy*. Dalam *fuzzy* fungsi keanggotaan yang biasa dipakai adalah fungsi keanggotaan segitiga, trapesium, Gaussian, fungsi keanggotaan S, fungsi keanggotaan lonceng dan sebagainya. Dalam sistem ini fungsi keanggotaan digunakan adalah fungsi keanggotaan segitiga :

$$\text{Triangular}(x;a,b,c) = \begin{cases} 0, & x \le a. \\ \frac{x-a}{b-a}, & a \le x \le b. \\ \frac{c-x}{c-b}, & b \le x \le c. \\ 0, & c \le x. \end{cases}$$




Gambar 2.1 Fungsi keanggotaan segitiga

### 2.3.6 Batasan (*Hedges*)

Dalam pembicaraan normal, manusia mungkin menambahkan kekaburan untuk memberikan pernyataan dengan menggunakan kata keterangan seperti sangat, agak. Kata keterangan adalah sebuah kata yang memodifikasi kata benda, kata sifat, kata keterangan lain, atau keseluruhan kalimat. Sebagai contoh, kata keterangan memodifikasi kata sifat, “orang itu sangat tinggi”.

Sebuah *hedges* memodifikasi himpunan *fuzzy* yang sudah ada secara matematis untuk menghitung beberapa kata keterangan yang ditambahkan.

## 2.4 Manajemen Resiko

Proses kegiatan manajemen resiko merupakan tugas gabungan dari departemen *underwriting* dan *loss control service*. Kegiatan ini terdiri dari tiga tingkatan kegiatan, yaitu:

a. Identifikasi resiko

   Dalam tahap ini, yang dilakukan adalah mengidentifikasi resiko apa saja yang mungkin dihadapi.

b. Evaluasi resiko

   Dalam tahap ini, faktor yang sangat penting yang menentukan diterima atau ditolaknya suatu permohonan kedit, yaitu dampak kerugian (severity) bagi pegawai yang merupakan pihak yang mengajukan.

Pengertian resiko dapat didefinisikan sebagai ketidakpastian akan terjadinya kerugian, baik kehidupan pribadi (personal), maupun kegiatan usaha (business). Bentuk dari resiko itu dapat diketegorikan sebagai berikut:

a. Resiko murni adalah bentuk resiko yang jika terjadi, dapat menimbulkan kerugian (loss) atau tidak menimbulkan kerugian (no loss/brek even).
b. Resiko spekulatif adalah bentuk resiko yang jika terjadi, dapat menimbulkan kerugian (*loss*) atau tidak menimbulkan kerugian (*no loss/brek even*), atau mendatangkan keuntungan (*gain*).

## 2.5 Penggunaan *Data Flow Diagram* (DFD)

Meskipun suatu analisa yang disebut dengan DFD mempunyai struktur tersendiri, namun sistem analisa dapat meletakkan secara bersamaan sebuah gambar yang merepresentasikan seluruh proses-proses data dalam sebuah organisasi. Pendekatan data *flow* menitik beratkan pada logika yang tersirat dari suatu sistem.

Dengan menggunakan kombinasi sistem, sistem analisa dapat membuat sebuah gambaran dari suatu proses yang sebenarnya dengan menggunakan dokumen sistem.

### 2.5.1 Keuntungan pembuatan data flow

Data *flow* mempunyai lima keuntungan utama dari penjelasan-penjelasan jalannya data dalam sistem, yaitu:

a. Kebebasan yang berasal dari kepercayaan untuk mengimplementasikan secara benar teknik sistem dari suatu sistem yang baru.
b. Memberikan pengertian dari hubungan sistem-sistem dan subsistem yang ada.
c. Komunikasi mengenai pengetahuan sistem bagi *user* melalui DFD
d. Analisa dari sebuah usulan sistem untuk menentukan jika data dan proses-proses yang ada dapat didefinisikan secara mudah.
e. Penggunaan data *flow* merupakan keuntungan tambahan yang dapat digunakan sebagai latihan bagi sistem analis, kesempatan sistem analis

menjadi lebih baik untuk mengerti tentang hubungan sistem dan subsistem yang ada di dalamnya.

Keuntungan dari kelima penggunaan data *flow* tersebut dapat digunakan sebagai *tools* yang interaktif dengan *user*. Hal yang menarik dalam penggunaan DFD adalah ditunjukannya kepada *user* gambaran-gambaran secara lengkap dari sistem. *User* dapat menanyakan guna memberikan komentar pada konsep, sistem analis dapat merubah sistem berdasarkan keinginan *user*. Keuntungan terakhir dari penggunaan DFD adalah dapat mengikuti sistem analis untuk mendeskripsikan komponen-komponen yang digunakan dalam suatu diagram. Analisa dapat ditampilkan untuk menjamin bahwa semua *output* mempunyai isi atau memperoleh data inputan dari prosesnya.

### 2.5.2 Pembuatan DFD

Untuk memulai sebuah DFD dari suatu sistem biasanya dituangkan dalam sebuah daftar dengan empat kategori yaitu *entity* luar, arus data, proses, dan penyimpanan data. Daftar ini akan membantu menentukan batasan-batasan dari suatu sistem yang akan digambarkan. Pada dasarnya daftar itu berisi elemen-elemen data yang dikarang. Elemen-elemen tersebut terdiri dari:

a. Pembuatan *context* diagram

*Context* diagram adalah level yang tertinggi dalam sebuah DFD dan hanya berisi satu proses serta merupakan representasi dari sebuah sistem. Proses dimulai dengan penomeran ke-0 dan untuk seluruh *entity* luar akan

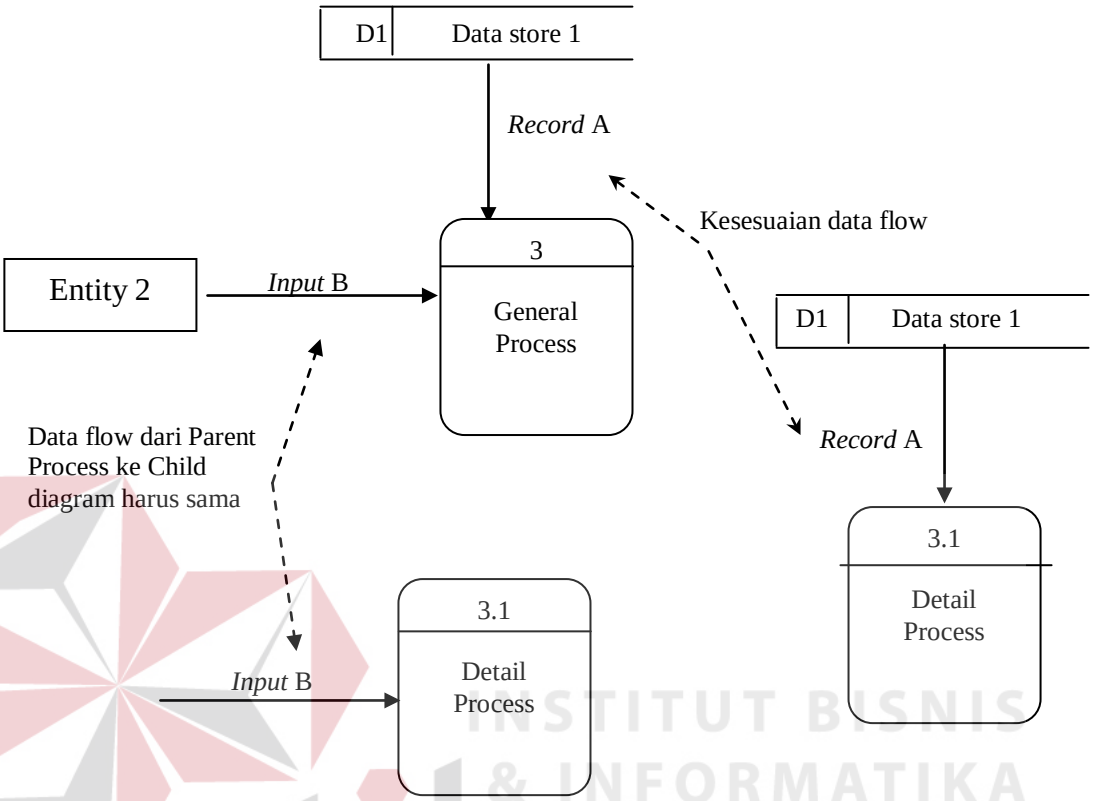
ditunjukkan dalam *context* diagram yang sama seperti data awal yang dikirim dari *entity* luar. *Context* diagram tidak berisi penyimpanan data.

b. Pembuatan diagram level 0 serta level berikutnya

Diagram level 0 dihasilkan oleh *context* diagram dan berisi proses-proses. Pengisian proses-proses yang berlebihan pada level ini akan menghasilkan sebuah diagram yang salah, sehingga sulit untuk dimengerti. Masing-masing proses diberikan penomoran dengan sebuah bentuk *integer*. Umumnya dimulai dari kiri atas dan penyelesaiannya di kanan bawah dalam sebuah bentuk diagram.

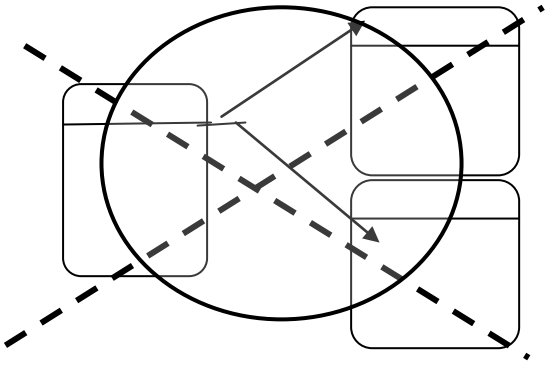
c. Pembuatan *child* diagram

*Child* diagram diberikan nomor yang sama seperti proses di atasnya (parent proses) dalam diagram level 0. Contohnya, proses 3 harus diturunkan ke diagram 3, proses pada *child* diagram menggunakan penomoran unik untuk masing-masing proses dengan mengikuti penomoran proses di atasnya. Contohnya, dalam diagram 3 proses-proses diberikan nomor 3.1, 3.2, 3.3 dan seterusnya. Konversi ini diikuti oleh analis sistem untuk menelusuri seri-seri dari proses-proses yang dikeluarkan oleh beberapa level, jika pada proses diagram level 0 digambarkan sebagai 1, 2, , dan 3 maka *child* diagram-diagramnya adalah 1, 2, dan 3 pada level yang sama. ilustrasi level detil dengan sebuah *child* DFD dapat ditunjukkan pada gambar :

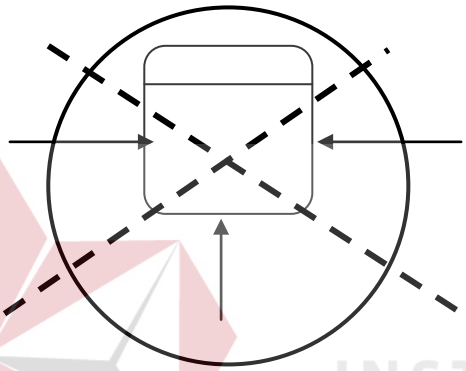



Gambar 2.2 Contoh ilustrasi detil child diagram

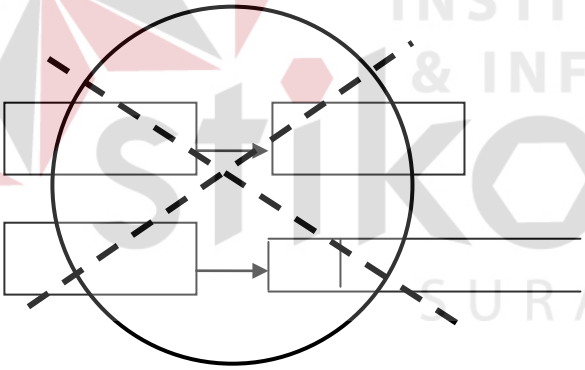
d. Pengecekan kesalahan-kesalahan pada diagram digunakan untuk melihat kesalahan-kesalahan yang terdapat pada sebuah DFD. Beberapa kesalahan-kesalahan yang umum terjadi ketika penggambaran/ pembuatan DFD, ditunjukkan pada gambar berikut, adalah :



Sebuah data *flow* tidak diperbolehkan mempunyai percabangan / memisahkan diri (*flow*) ke dalam dua atau lebih data *flow* yang berbeda



Sebuah proses harus mempunyai minimal satu *input*an data *flow* dan satu *output* data *flow*



Semua data *flow* salah satunya harus berasal atau berakhir pada sebuah proses



Gambar 2.3 Kesalahan penulisan proses dalam DFD

Adapun dari keterangan gambar di atas adalah :

a. Tidak menginputkan sebuah arus data atau arah panah langsung. Sebagai contoh adalah penggambaran proses yang menunjukkan sebuah data *flow* seperti *input* atau seperti *output*. Tiap-tiap proses pengubahan data harus menerima *input* dan *output*. Tipe kesalahan ini terjadi ketika sistem analis

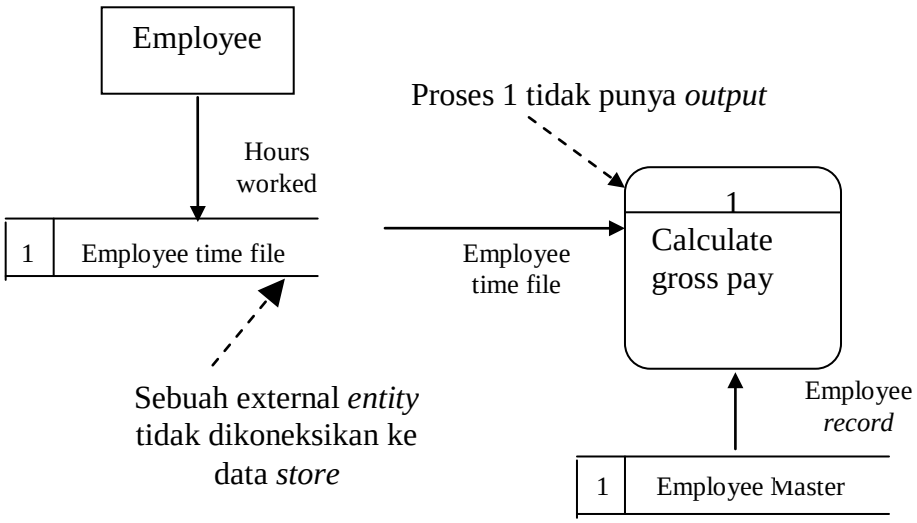
tidak memasukkan sebuah data *flow* atau meletakkan sebuah arah panah ditempat yang salah.

b. Hubungan penyimpanan data dan *entity* luar secara langsung satu sama lain. Data *store* dan *entity* tidak mungkin dikoneksikan satu sama lain ; data *store* dan *entity* luar harus dikoneksikan melalui sebuah proses.

c. Kesalahan penamaan (label) pada proses-proses atau data *flow*. Pengecekan DFD untuk memastikan bahwa tiap-tiap obyek atau data *flow* telah diberikan label. Sebuah proses haruslah di indikasikan seperti nama dari sistem atau menggunakan format kata kerja. Tiap data *flow* haruslah dideskripsikan dengan sebuah kata benda.

d. .Memasukkan lebih dari sembilan proses dalam sebuah DFD. Memiliki banyak proses akan mengakibatkan kekacauan pada diagram sehingga dapat menyebabkan kebingungan dalam pembacaan sebuah proses dan akan menghalangi tingkat komunikasi. Jika lebih dari sembilan proses dalam sebuah sistem, maka beberapa grup dalam proses dilakukan bersama-sama ke dalam sebuah sub sistem dan meletakkannya dalam sebuah *child* diagram.

e. Menghilangkan suatu arus data. Pengujian dari suatu diagram yang menunjukkan garis/arah (*flow*), dimana untuk setiap proses data *flow* hanya mempunyai *input* data, *output* kecuali dalam kasus dari detil (*child*). Setiap *child* data dari DFD, arah arus data seringkali digambarkan untuk mengidentifikasikan bahwa diagram tersebut kehilangan data *flow*. Seperti di tunjukkan pada gambar berikut:




Gambar 2.4 Contoh kesalahan pada DFD

f. Membuat ketidaksesuiaan komposisi dalam *child* diagram , dimana tiap *child* diagram harus mempunyai *input* dan *output* arus data yang sama seperti proses dilevel atasnya (*parent process*). Pengecualian untuk *rule* ini adalah kurangnya *output*, seperti kesalahan garis yang ada didalam *child* diagram.

### 2.5.3 Perbedaan DFD (Logika dan Fisik)

Tabel 2.1 Perbedaan DFD logika dan fisik

| **Disain** | **Logika** | **Fisik** |
|---|---|---|
| Gambaran model | Operasi-operasi bisnis | Bagaimana sistem akan diimplementasikan (atau bagaimana sistem dijalankan) |
| **Disain** | **Logika** | **Fisik** |

| | | |
|---|---|---|
| Apa yang ditampilkan oleh proses | Aktivitas bisnis | Program-program, modul program, dan prosedur-prosedur manual |
| Apa yang ditampilkan oleh data *store* | Koleksi-koleksi dari data yang dikesampingkan dari bagaimana data tersebut di simpan | *File-file* fisik dan *database-database* dari *file-file* manual |
| Kontrol sistem | Menunjukkan kontrol-kontrol bisnis | Menunjukkan kontrol-kontrol untuk validasi *input* data, untuk memperoleh sebuah *record* , untuk memastikan kesuksesan proses dan untuk keamanan system |

## 2.6 Desain Output (Efektif)

*Output* merupakan sebuah informasi yang dikirimkan kepada *user* melalui suatu sistem informasi (seperti : Internet, Extranet, dan WWW). Beberapa data yang dibutuhkan diproses secara teliti sebelum dinyatakan layak sebagai sebuah *output*; penyimpanan data lain, dan ketika data-data tersebut dibutuhkan kembali, data-data tersebut berupa *ouput* yang membutuhkan proses yang sedikit. *Output* dapat diambil dari beberapa bentuk yaitu : bentuk laporan yang dihasilkan printer dan tampilan layar komputer, mikrofon (suara).

Oleh sebab itu, penggunaan *output* sangat penting guna menjamin pemakaian dan penerimaan dari suatu sistem informasi. Ada beberapa obyek dimana sistem analis mencoba untuk mencapai uatu desain *ouput* yang tepat.

Obyek-obyek tersebut antara lain:

a. Desain output untuk melayani sebuah tujuan tertentu.

b. Pembuatan *output* yang disesuaikan bagi kebutuhan *user*.

c. Pengiriman sejumlah *output*.

d. Pengelolaan distribusi *output*.

e. Pengelolaan waktu *output*.

f. Penyesuaian metode *output* yang paling efektif.

## 2.7 Desain Input (Efektif)

Bentuk-bentuk desain secara khusus mungkin dapat digunakan jika suatu analisa sistem dapat disesuaikan dengan bentuk desain secara lengkap dan bermanfaat. Hal itu juga penting untuk mengenalkan secara dini desain yang digunakan, arus data atau bentuk-bentuk yang tidak dibutuhkan pada sumber-sumber suatu organisasi dan hal itu harus dihilangkan. Untuk mendesain *form-form* yang baik dan berguna, ada beberapa hal dari desain *form* yang harus diterapkan antara lain :

a. Membuat *form-form* tersebut mudah dalam pengisiannya.

b. Memastikan *form-form* tersebut sesuai dengan tujuan untuk masing-masing desain.

c. Desain *form* digunakan untuk menjamin kelengkapannya.

d. Mempertahankan *form-form* yang menarik.

## 2.8 Desain database

Desain *database* merupakan gambaran atau deskripsi dari file-file yang digunakan serta merupakan sebuah pendefinisian normal dan merupakan

gambaran dari pusat penyimpanan dari data tertentu yang digunakan dalam beberapa aplikasi yang berbeda. Disain *database* terdiri dari :

### 2.8.1 Bentuk database dan file

Ada dua bentuk pendekatan pada proses penyimpanan data dalam sebuah sistemkomputer. Metode pertama adalah untuk menyimpan *file-file* tunggal, masing-masing dengan bentuk unik untuk berbagai macam aplikasi. Pendekatan yang kedua adalah untuk menyimpan data dalam sebuah sistem komputer dengan melibatkan pembuatan sebuah *database*.

### 2.8.2 Database

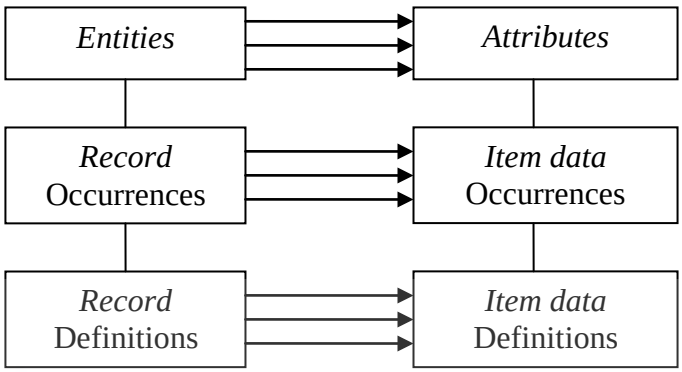
*Database* bukan hanya merupakan sebuah koleksi dari suatu *file-file*. Meskipun, sebuah *database* merupakan sebuah pusat sumber data yang disimpan oleh beberapa *user* dari sebuah aplikasi-aplikai yang bervariasi. Inti dari sebuah *database* adalah DBMS (*Database Management* Sistem), dimana diikuti dengan kreasi, modifikasi, dan perubahan (*update*) dari *database*.

### 2.8.3 Konsep-konsep data

Konsep-konsep data merupakan hal yang sangat penting untuk dimengerti bagaimana data dipresentasikan kembali sebelum memutuskan penggunaan *file* atau *database*.

a. Realita, data dan metadata

Hubungan antara realita, data dan meta data yaitu didalamnya terdapat *entity* dan attribut (realita), *record* dan *item data* (data), definisi *record* dan definisi *item data* (meta data). Ditunjukkan pada gambar berikut:




Gambar 2.5 Hubungan antara realita, data dan metadata

*Entity* → Merupakan beberapa obyek atau kejadian tentang dimana dapat mencocokkan koleksi data sebagai sebuah entity. *Entity* juga dapat berupa sebuah kejadian atau unit dari satu waktu.

*Relationship*→ Merupakan hubungan antara entity, adapun tipe-tipe dari *relationship* antara lain (1) *one to one* (1:1), (2) *one to many* (1:M), dan (3) *many to many* (M:N).

*Attribute* → Merupakan sebuah karakteristik dari sebuah *entity*.

*Record* → Merupakan suatu kumpulan dari item-item data yang secara umum merupakan penjelasan umum dari *entity*.

b. Metadata

Metadata merupakan data dimana dijelaskan tentang data *file* atau *database*. Metadata menjelaskan pemberian nama dan menunjukkan panjang dari masing-masing item data. Metadata juga menjelaskan panjang dan komposisi dari tiap-tiap *record*.

c. Kelompok *file*

Sebuah *file* berisi grup-grup dari *record* yang digunakan untuk melengkapi informasi untuk suatu operasi-operasi, perencanaan, manajemen, dan pembuatan keputusan. Tipe-tipe dari *file* yang digunakan antara lain:

| | | |
|---|---|---|
| *1. File* master | : | berisi *record-record* dari sebuah kelompok *entity*. |
| *2. File* tabel | : | berisi data yang digunakan. |
| *3. File-file* transaksi | : | digunakan untuk mengisi perubahan (*update*) sebuah file master dan laporan-laporan. |
| *4. Work file* | : | digunakan untuk menjalankan program agar lebih efektif. |
| *5. File* laporan | : | memudahkan untuk menjalankan program (ketika tidak ada printer) |

### 2.8.2 Normalisasi

Normalisasi merupakan perubahan dari *user* yang ditampilkan secara lengkap dan simpanan data untuk ukuran dari yang terkecil, serta merupakan

struktur-struktur data yang stabil. Normalisasi dibutuhkan untuk mengorganisir data dan menghindari redudansi data (data *double*). Ada tiga bentuk normalisasi, yaitu :

a. *First Normal Form* (1NF)

   Langkah pertama ini terdapat pada sebuah relasi normalisasi yang digunakan untuk menghilangkan (menghapus) grup-grup yang berulang.

b. *Second Normal Form* (2NF)

   Pada bentuk normal yang kedua, seluruh attribut yang ada akan difungsikan tergantung pada PK (*Primary Key*).

c. *Third Normal Form* (3NF)

   Sebuah relasi penormalisasian yang ketiga adalah jika seluruh yang bukan kunci (PK) dari semua attribut seluruhnya difungsikan bergantung pada PK dan attribut tersebut bukan transitif ketergantungan (tanpa kunci).

### 2.8.4 Denormalisasi

Merupakan sebuah proses yang menangani model data logika dan pengirimannya ke dalam sebuah model data fisik untuk lebih mengefisienkan tugas-tugas tertentu yang dibutuhkan. Denormalisasi dapat diselesaikan dengan sebuah penomeran yang berbeda.

## 2.9 Desain User Interface

Bagaimanapun baik atau buruknya (minimnya) suatu tampilan, hal itu berpengaruh pada keberadaan representasi sebuah sistem. Desain terdiri dari:

a. Tipe-tipe *user interface* (tampilan)

b. Tampilan bahasa

c. Tampilan tanya – jawab

d. Menu-menu

e. Tampilan bentuk isian / bentuk *input* dan *output*

f. Tampilan bahasa perintah

g. *Graphical User Interface* (GUIs)