

2. LANDASAN TEORI

2.1. Java Server Pages (JSP)

2.1.1. Definisi JSP

Java Server Pages (JSP) adalah teknologi berbasis Java yang berjalan di *Platform* Java. JSP merupakan bagian dari J2EE (Java 2 Enterprise Edition) yang mengontrol kepuasan dan penampilan dari halaman *web* melalui kegunaan dari *Servlets*. *Servlets* adalah program kecil yang dikhususkan dalam halaman *web* serta dijalankan pada *web server* untuk memodifikasi halaman *web* sebelum disampaikan/dikirim kepada *user* yang memintanya. Teknologi JSP ditujukan sebagai *Servlet application program interface* (API). JSP yang dikenal dengan program Java, dijalankan dengan *web server*, biasanya *client* adalah *web browser*. *Template* yang digunakan biasanya HTML atau XML yang mengandung *link* ke *Java Servlet*, diberi nama akhir *file* berupa .JSP. (Rickyanto, 1997).

2.1.2. Penulisan *Script* JSP

Penulisan kode JSP menggunakan *tag-tag* yang memiliki aturan yang sama dengan XML. Misalnya, apabila ada *tag* pembuka, maka harus ada *tag* penutup. Saat pertama kali *file* JSP diproses, JSP akan dikompilasi menjadi *Servlets* yang kemudian disimpan di memori *server* sehingga proses pada pemanggilan berikutnya dapat berlangsung lebih cepat. Penulisan kode/*script* JSP selalu diawali oleh tanda `<%` dan diakhiri oleh tanda `%>`, jadi kode/*script* JSP ditulis didalam `<% ..... %>`.

Berikut ini adalah contoh penulisan *script*/kode JSP:

```
<HTML>
<HEAD>
<TITLE>Contoh JSP</TITLE>
</HEAD>
<BODY BGCOLOR="ffffcc">
<CENTER>
<H2>Tanggal dan Waktu</H2>
<%Java.util.Date tanggal = new java.util.Date( );
out.println("Tanggal hari ini adalah: "+tanggal);%>
```

```

</CENTER>
</BODY>
</HTML>

```

2.1.3 Elemen-Elemen JSP

JSP merupakan bahasa pemrograman yang bersifat *embedded* pada kode HTML maupun WML, namun juga dapat berdiri sendiri dimana *sintaks* JSP diatur sesuai dengan spesifikasi JSP. Sesuai teknologi pemrograman *web*, JSP memiliki bermacam-macam elemen yang dapat digunakan dalam suatu halaman JSP yang mengacu pada *file* program yang berisikan kode-kode JSP.

Ada tiga komponen/elemen utama dalam JSP, yaitu:

a. Direktif

Direktif merupakan salah satu macam elemen utama dan sering digunakan dalam pemrograman JSP, yang terdiri dari tiga macam, yaitu:

1) Direktif Page

Direktif page `<%@ page ... %>` digunakan untuk mendefinisikan atribut yang penting bagi keseluruhan kode dalam halaman tersebut (untuk mengatur *setting* halaman JSP).

Contoh:

```
<%@ page ATRIBUT %>
```

Berikut ini adalah tabel atribut pada *direktif page*:

Tabel 2.1. Tabel Atribut pada Direktif Page

No.	Atribut	Deskripsi	Default
1	<i>Language</i>	Mendefinisikan bahasa <i>scripting</i> yang digunakan. Sementara ini hanya Resin (www.caucho.com) yang mendukung <i>Javascript</i> , selain Java.	Java
2	<i>Import</i>	Untuk mengimpor <i>package</i> atau <i>class</i> Java sebagaimana program Java umumnya.	
3	<i>Session</i>	Menentukan apakah halaman JSP menggunakan HTTP <i>session</i> .	"true"

Tabel 2.1. Tabel Atribut pada *Direktif Page* (Lanjutan)

4	<i>ContentType</i>	Mendeskripsikan <i>encoding</i> karakter yang digunakan serta tipe MIME sebagai <i>response</i> dari halaman JSP. Penggunaan: “TYPE= tipe MIME” atau “TYPE=tipe MIME; charset=CHARSET” CHARSET adalah karakter <i>encoding</i> yang digunakan.	<i>DefaultType</i> adalah ”text/html” dan nilai <i>default</i> CHARSET adalah ISO-8859-1
5	<i>IsThread-Safe</i>	Mendefinisikan tingkat keamanan mengenai masalah <i>threading</i> dari halaman JSP. Jika “false”, <i>request</i> akan diproses sebagai <i>single</i> Thread, berurutan sesuai urutan datangnya <i>request</i> .	“true”
6	<i>Buffer</i>	Menentukan model <i>buffering</i> untuk output <i>stream</i> ke <i>client</i> . Apabila nilainya diisi <i>none</i> , maka tidak terjadi <i>buffering</i> .	<i>Default</i> adalah dilakukan <i>buffering</i> maks 8 kb.
7	<i>Info</i>	Mendefinisikan string informasi yang dapat diperoleh dari implementasi metode <i>Servlet.getServletInfo()</i> .	
8	<i>IsErrorPage</i>	Mengindikasikan apakah halaman JSP ini merupakan halaman <i>error</i> dari halaman JSP lain.	“false”
9	<i>ErrorPage</i>	Mendefinisikan URL dari <i>file</i> JSP lain yang dipanggil, apabila terjadi <i>error</i> saat <i>file</i> JSP diproses.	

Tabel 2.1. Tabel Atribut pada *Direktif Page* (Lanjutan)

10	<i>AutoFlush</i>	Apabila "true", <i>output buffer</i> akan diflush/dikeluarkan setelah <i>buffer</i> penuh/full. Apabila "false", pesan <i>exception</i> akan dikeluarkan untuk mengindikasikan jika <i>buffer</i> telah full.	"true"
11	<i>Extends</i>	Digunakan untuk men-generate <i>superclass</i> .	

Sumber: Tabel Atribut *Java Server Pages* (Rickyanto, 2002).

2) Direktif Include

Direktif *include* `<%@ include ... %>` digunakan untuk menyisipkan isi *file* lain dalam *file* JSP (untuk menggabungkan halaman atau prosedur-prosedur yang sering digunakan ke dalam program JSP lainnya).

Contoh:

```
<%@include file="namafile.html" %>
```

3) Direktif Taglib

Direktif *Taglib* `<%@ taglib uri="taglibraryURI" prefix="tagprefix" %>` digunakan untuk penggunaan *tag library* atau *tag* tambahan.

Contoh penggunaan:

```
<%@ taglib uri=http://serverku.com/tagku prefix="asik" %>
```

```
<asik:proses>
```

```
—
```

```
</asik:proses>
```

Berikut ini adalah tabel atribut pada *direktif taglib*:

Tabel 2.2. Tabel Atribut pada *Direktif Taglib*

Atribut	Deskripsi	Nilai Default
<i>uri</i>	URI (<i>Uniform Resource Identifier</i>) yang mengidentifikasi <i>tag library descriptor</i> .	Tidak mengkhususkan suatu nilai karena kompilasi <i>error</i> .

Tabel 2.2. Tabel Atribut pada *Direktif Taglib* (Lanjutan)

<i>tagprefix</i>	Mendefinisikan <i>prefix string</i> dalam <i>prefix: tagname</i> yang digunakan untuk mendefinisikan <i>custom tag</i> .	Tidak mengkhususkan suatu nilai karena kompilasi <i>error</i> .
------------------	--	---

Sumber: Tabel Atribut *Java Server Pages* (Allamaraju, 2000).

b. Elemen *Scripting*

Elemen *Scripting* terdiri dari tiga macam, yaitu:

1) Skriptlet (*Scriptlet tag*)

Tag `<% ... %>` digunakan untuk deklarasi, ekspresi dan kode lain (untuk menulis proses dalam JSP sehingga menghasilkan nilai-nilai yang diinginkan).

Sintaksis:

`<% pernyataan kode java yang sah %>`

Contoh:

`<jsp:scriptlet> kode skriptlet </jsp:scriptlet>`

2) Deklarasi (*Declaration tag*)

Tag `<% ... %>` digunakan untuk mendeklarasikan variabel atau metode, baik tipe maupun nilai awal variabel.

Sintaksis:

`<%! Deklarasi variabel dan metode java %>`

Contoh:

`<jsp:declaration> kode deklarasi </jsp:declaration>`

3) Ekspresi (*Expression tag*)

Tag `<% ... %>` digunakan untuk ekspresi dalam Java dan menampilkannya sebagai *String* pada *browser* (untuk mencetak isi variabel).

Sintaksis:

`<%= ekspresi java yang akan dievaluasi %>`

Contoh:

`<jsp:expression> kode ekspresi </jsp:expression>`

c. *Action*

Action merupakan elemen JSP yang berupa *tag-tag*. Ada dua macam *action*, yaitu:

1) Tag Action Standar

Tag Action Standar adalah *tag* yang didefinisikan dalam spesifikasi JSP. Ada beberapa macam *Tag Action Standar*, yaitu:

- `<jsp:usebean>`

Tag ini digunakan untuk mengasosiasikan *Java bean* dalam *file* JSP sehingga kita dapat menggunakan obyek yang terdapat dalam *Java bean* untuk diaplikasikan dalam kode JSP.

Sintaksis:

```
<jsp:useBean id="namaid" scope="page" detail bean />
```

Dimana detail bean termasuk salah satu dari:

- ❑ `class="nama class"`
- ❑ `class="nama class" type="nama tipe"`
- ❑ `beanName="nama bean" type="nama tipe"`
- ❑ `type="nama tipe"`

Contoh:

```
<jsp:useBean id="namaid" scope="page" class="java.lang.String" >
```

- `<jsp:setproperty>`

Tag ini sebagai pelengkap penggunaan *tag* `<jsp:usebean>`.

Sintaksis:

```
<jsp:setProperty name="namabean" detail properti />
```

Dimana detail properti termasuk salah satu dari:

- ❑ `property="**"`
- ❑ `property="nama properti"`
- ❑ `property="nama properti" param="nama param"`
- ❑ `property="nama properti" value="nama value"`

Contoh:

```
<jsp:setProperty name="namabean" property="**"/>
```

- `<jsp:getproperty>`

Tag ini digunakan untuk mendapatkan nilai properti dari *bean* dan merupakan pelengkap dari *tag* `<jsp:setproperty>`.

Sintaksis:

```
<jsp:getproperty name="namabean" property="properti"/>
```

- `<jsp:include>`

Tag ini digunakan untuk mengikutsertakan *file* lain dalam JSP saat *request* dilakukan.

Sintaksis:

```
<jsp:include page="satu.html" flush="true" />
```

- `<jsp:param>`

Tag ini sebagai pelengkap *tag* `<jsp:params>`, `<jsp:plugin>` maupun `<jsp:forward>`. *Translation error* akan terjadi apabila digunakan di tempat lain.

Sintaksis:

```
<jsp:param name="nama param" value="nilai param" />
```

- `<jsp:params>`

Tag ini digunakan dalam `<jsp:plugin>` saja. Jika digunakan di tempat lain, akan terjadi *translation error*.

- `<jsp:fallback>`

Tag ini digunakan sebagai *subelemen* dari `<jsp:plugin>` yang berfungsi untuk menampilkan pesan, apabila proses *plugin* mengalami *error*.

- `<jsp:plugin>`

Tag ini digunakan untuk menghasilkan kode-kode yang sesuai target *browser* dalam menyisipkan komponen *Java bean* atau *Applet* secara otomatis.

Sintaksis:

```
<jsp:plugin type="bean|applet" code="kode obyek"
    codebase="Codebase obyek" align="alignment"
    archive="daftar archive" height="height" hspace="hspace"
    jreversion="jreversion" name="nama komponen"
    vspace="vspace" width="width" nspluginurl="url"
    iepluginurl="url" >
    <jsp:params>
        <jsp:param name="nama param" value="nilai param" />
        <jsp:param name="nama param" value="nilai param" />
        ....
    </jsp:params>
    <jsp:fallback> teks yang akan ditampilkan </jsp:fallback>
</jsp:plugin>
```

- `<jsp:forward>`

Tag ini digunakan untuk mem-*forward* ke *file* JSP, *Servlet* atau *file* statik lain.

Sintaksis:

```
<jsp:forward page="contoh.jsp" />
```

2) Custom Tag

Dalam bahasa pemrograman JSP, dapat membuat *tag* baru, dimana tiap *tag* memiliki fungsi dan kemampuan yang didefinisikan sendiri. *Tag* inilah yang disebut *custom tag*.

2.1.4. Web Server JSP

Agar *file-file Java Server Pages* yang berbasis Java dapat dijalankan, maka diperlukan *web server* yang mampu memproses Java, atau minimal JSP *engine* yang dapat terintegrasi dengan *web server*. Banyak *web server* yang dapat dijadikan pilihan sebagai *web server* JSP, diantaranya adalah Blazix, ServletExec, Jrun, WebLogic, WebSphere, Apache Tomcat ataupun Lite Web Server.

Namun untuk perancangan dan pembuatan tugas akhir ini, *web server* yang digunakan adalah Blazix Web Server. Blazix diproduksi oleh Desiderata Software. Instalasi Blazix cukup mudah, dan cara kerja *web server* ini tidak menyulitkan. Dengan ukuran *file* instalasi yang hanya 1,5 *megabyte*, *web server* ini tidak menghabiskan banyak tempat saat diinstalasi di dalam komputer. Blazix merupakan *web server standalone*, maka tidak perlu diintegrasikan ke *web server* lain. Saat dijalankan, *web server* lain dapat dimatikan. Untuk men-*download file* instalasi yang resmi dapat dilakukan di lokasi <http://www.blazix.com/>.

2.2. HTML (*Hypertext Markup Language*)

2.2.1. Definisi HTML

HTML adalah serangkaian konstruksi standar yang dapat membangun halaman *web*. HTML menggunakan perintah yang disebut sebagai *tag* untuk menetapkan bagaimana sebuah teks, grafik, video dan suara ditempatkan dalam dokumen dan untuk menciptakan *link*(rangkaihan) dinamik ke dokumen lain dan obyek yang sudah disimpan dalam komputer yang sama ataupun komputer yang

jauh. Disini pengguna hanya memerlukan sebuah kata kunci yang perlu di klik, yang dengan segera akan memindahkan dokumen ke dokumen yang lain. HTML didesain untuk menspesifikasikan ke organisasi logika dari dokumen teks, dengan *extentions* penting untuk *hypertext links* dan interaksi *user*.

2.2.2. Elemen dasar HTML

Ada empat macam elemen dasar yang ada dalam HTML, yaitu:

a. Jenis Dokumen (HTML)

HTML *tag* terdapat pada awal dan akhir dari *file* HTML. Tag ini adalah tag tentang Web browser dimana HTML pada dokumen berawal dan berakhir.

Strukturnya sebagai berikut:

```
<HTML>
....desain web....
</HTML>
```

b. Judul (*TITLE*)

Elemen ini harus selalu ditempatkan dalam struktur *header*/mukadimah. Diantara *title tag* terdapat judul dari dokumen kita. Bagian ini akan timbul pada bagian atas dari *title bar*, dan juga pada *history list*.

Strukturnya sebagai berikut:

```
<TITLE>....judul web....</TITLE>
```

c. Mukadimah (*Header*)

HEAD tag berisi tentang semua informasi *header* dokumen, keterangan umum, seperti judul dan sebagainya.

Strukturnya sebagai berikut:

```
<HEAD>
.....
</HEAD>
```

d. Batang Tubuh (*BODY*)

Batang tubuh atau sering disebut *BODY*, diletakkan setelah struktur *HEAD*. Diantara *BODY tag*, terdapat isian yang dapat ditampilkan dalam *browser window*. Semua teks, grafik, *link* dan sebagainya diletakkan diantara *BODY tag*. Jadi *BODY tag* ini mengandung isi dari halaman HTML.

Strukturnya sebagai berikut:

```
<BODY>
.....
```

</BODY>

Struktur HTML secara keseluruhan adalah sebagai berikut:

```
<HTML>
<HEAD>
<TITLE>Document Title</TITLE>
</HEAD>
.....
<BODY>
.....
</BODY>
</HTML>
```

2.2.3. Karakter khusus HTML

Dalam dokumen HTML terdapat beberapa karakter ASCII, yaitu:

a. *The ampersand character (&)*

Karakter ini digunakan untuk mengindikasikan *entity* dan referensi karakter.

b. *The left and right angle brackets (<>)*

Karakter ini digunakan untuk menunjukkan *markup tag*.

c. *The double quotation mark (“”)*

Karakter ini digunakan untuk menandai *literal string* dengan *markup tag*.

Karakter khusus dan referensi *entity* terdapat dalam tabel 2.3. berikut ini:

Tabel 2.3. Tabel Karakter Khusus dan Referensi Entity

No.	Character	Character Reference	Entity Reference
1	Left angle bracket (<)	<	<
2	Right angle bracket (>)	>	>
3	Ampersand sign (&)	&	&
4	Double quotation sign (“”)	"	"

Sumber: Tabel *Special Characters in HTML* (Graham, 1995).

2.3. Macromedia Dreamweaver MX

Macromedia Dreamweaver MX merupakan program aplikasi yang dapat mengedit HTML secara visual dan mengelola *website* serta *pages*. Kemampuan yang disediakan program aplikasi Macromedia Dreamweaver MX antara lain *Interface* Macromedia Dreamweaver MX, tampilan panel – panel, dokumen baru,

teks editor yang terintegrasi, *panel CSS style*, *templates*, petunjuk kode, *panel snippets*, *tag editor* dan *web developer*.

Macromedia Dreamweaver MX dilengkapi dengan kemampuan *client-side scripting*. Beberapa *script server side* dapat dikenali oleh Macromedia Dreamweaver MX, misalnya JSP, ASP, PHP, ASP.Net dan CFML.

Berikut ini adalah kemampuan-kemampuan yang disediakan oleh program aplikasi Macromedia Dreamweaver MX:

□ *Interface Macromedia Dreamweaver MX*

Macromedia Dreamweaver MX tampil dengan *interface* standar yang lebih konsisten dengan produk-produk *author Web Macromedia* lainnya. *Interface* ini meliputi tampilan panel, penamaan menu, ikon, *shortcut keyboard*, *launcher bar* dan yang lainnya.

□ *Tampilan Panel-Panel*

Panel-panel di dalam Macromedia Dreamweaver MX ditempelkan pada jendela kerja sehingga memudahkan *user* bekerja. Selain itu tampilan panel-panel tersebut bisa dimaksimalkan (ditampilkan secara lengkap) maupun diminimalkan untuk menghemat area kerja.

□ *Dokumen Baru*

Macromedia Dreamweaver MX memberikan fasilitas baru di dalam pembuatan dokumen baru, yaitu kotak dialog *New Document*, yang memungkinkan *user* memilih atau menentukan *layout* halaman maupun kode yang akan digunakan.

□ *Teks Editor Yang Terintegrasi*

Fasilitas *Code View* maupun *Code Inspector* tampil lebih terintegrasi. *Code View* menyediakan fitur-fitur seperti *autoediting*, *live syntax coloring* untuk HTML dan JavaScript, *toolbar* dengan kode yang lebih terorientasi. Fitur-fitur ini memudahkan penulisan kode sehingga semudah seperti bekerja dengan *Design View* (bekerja secara visual).

□ *Panel CSS Style*

Panel CSS Style tampil lebih lengkap dengan *Style sheet* yang memudahkan pengeditan halaman. Selain itu panel *CSS Style* lebih

memudahkan *user* di dalam membedakan *style* yang ditentukan secara lokal dengan *style* yang ditentukan secara eksternal.

□ *Templates*

Fasilitas *Templates* yang disediakan Macromedia Dreamweaver MX memungkinkan *user* lebih mengontrol *layout* halaman.

□ *Petunjuk Code*

Pada saat akan melakukan pengkodean, menu *pop-up* akan muncul memberikan petunjuk tentang kode yang sesuai dengan *item* yang dipilih *user*.

□ *Panel Snippets*

Panel *Snippets* memungkinkan penyimpanan, pengaturan, dan penggunaan kembali kode-kode.

□ *Tag Editor*

Pada jendela *Code View*, *Tag Editor* memungkinkan pengeditan properti dari *tag* individual termasuk *HTML tag*, *CFML tag* dan *ASP.NET tag*.

□ *Web Developer*

Macromedia Dreamweaver MX memberikan fitur-fitur baru untuk *Web Developer* yang di dalamnya termasuk *ColdFusion MX Support*, *ASP.NET Support*, *PHP Support*, *Web Services Introspections*.

(Wahana, 2002).

2.4. World Wide Web

World Wide Web adalah koleksi sederhana dokumen multimedia yang terhubung dengan *hyperlinks* dokumen yang hampir sama seperti laporan tertulis. Dimana *World Wide Web* dapat berfungsi sebagai alat komunikasi untuk menyampaikan informasi., yang menghasilkan dokumen dalam bentuk kata, angka, gambar, suara (misalnya musik), animasi, atau yang lainnya yang dikemas dalam bentuk *file* komputer. Dengan kata lain dapat menciptakan dokumen multimedia, yaitu dokumen yang menggunakan media multiplikasi/metode dari komunikasi. (Nelson, 1996).

2.5.Database

2.5.1. Definisi Database

Kata '*database*' adalah kata yang paling banyak digunakan sekarang ini sekaligus merupakan kata yang memiliki banyak kesalahpahaman makna. Banyak orang yang menganggap *database* adalah sebuah *file*/arsip-arsip saja. Namun sebenarnya *database* merupakan sekumpulan informasi yang dapat diatur dan diakses sesuai dengan struktur logika dari informasi tersebut *Database* adalah jantung (inti) dari banyak informasi, yang merupakan kumpulan dari komputer data yang diintegrasikan, dan sistem informasi yang berbasis *web* juga tidak berbeda. (Stanek, 1996).

2.5.2. Tipe Database

Tipe sistem *database* yang digunakan untuk menyimpan dan mengolah data, ada 2 macam, yaitu:

a) *Relational*

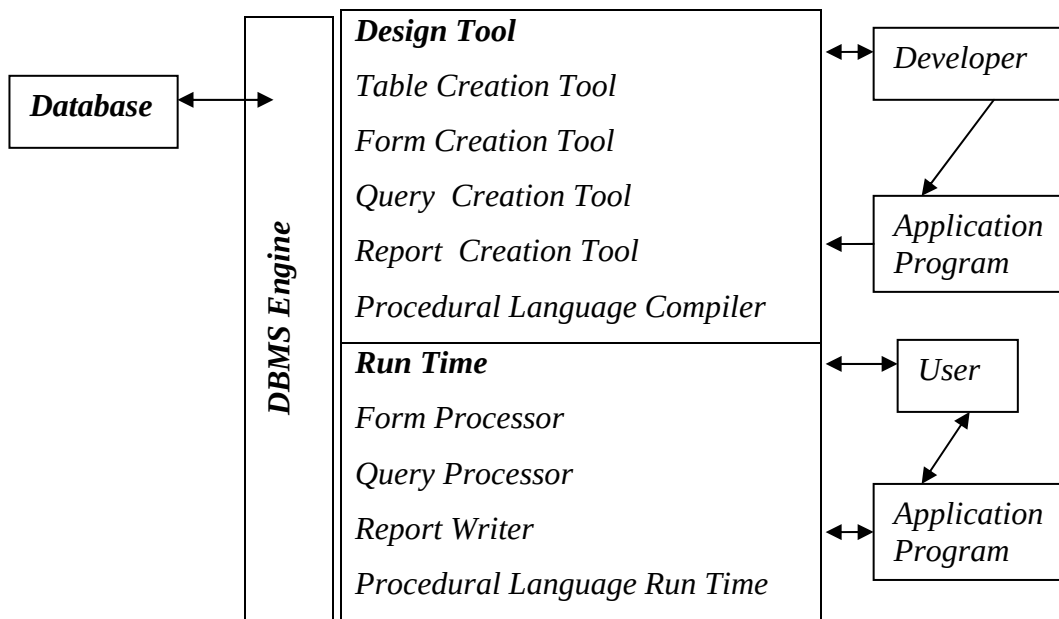
Menyimpan dan mengambil data didasarkan pada *attribute relationship*.

b) *Object-oriented*

Menyimpan dan mengambil data didasarkan pada *object-attributes*.

2.5.3. Komponen Sistem Database

Database diproses oleh DBMS, dimana digunakan oleh pengembang/developer dan user, baik langsung maupun tak langsung melalui *application program*. Untuk lebih jelasnya, komponen sistem *database* digambarkan pada gambar 2.1.



Gambar 2.1. Komponen sistem database

Sumber: Tabel Komponen sistem database. (Kroenke, 1995).

2.6. Data Flow Diagram

2.6.1 Definisi DFD

Data Flow Diagram adalah model sistem untuk mengekspresikan suatu proses serta aliran data yang ada antar proses tersebut. (Jalote, 1994).

2.6.2 Elemen dasar/symbol DFD

DFD mengekspresikan suatu sistem dalam bentuk gambar simbol/elemen dasar tertentu yang mengilustrasikan bagaimana suatu data mengalir, dan menghubungkannya dengan proses yang lain. Meskipun secara tidak langsung menekankan pada data, namun sebenarnya inti dari sistem tersebut terletak pada prosesnya.

DFD mempunyai 4 elemen dasar/symbol, yaitu:

a. *Data Source* dan *Destination* (entitas)

Entitas atau sering disebut sebagai terminator berfungsi sebagai input dan output sistem, yang menggambarkan asal dan tujuan dari sistem tersebut. Yang termasuk didalamnya adalah seseorang (misalnya manager),

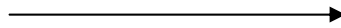
sekelompok orang serta departemen baik didalam ataupun diluar sistem yang ada. Simbol dari entitas/terminator digambarkan pada gambar 2.2.



Gambar 2.2. Simbol Entitas

b. *Data flow* (aliran data)

Data flow berfungsi untuk menggambarkan arah aliran data dari sistem. Simbol dari *data flow* digambarkan pada gambar 2.3.



Gambar 2.3. Simbol Data Flow

Dalam satu entitas ataupun proses tidak selalu harus satu *data flow* saja, tetapi bisa lebih dari satu *data flow*, yang berasal dari suatu entitas ataupun proses, serta dapat juga lebih dari satu *data flow* yang menuju ke satu entitas ataupun proses.

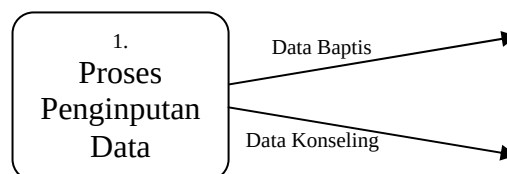
Berikut ini adalah contoh-contoh penggambaran *data flow* dalam DFD:

- Gambar 2.3.1. adalah gambar *data flow* yang jumlahnya hanya satu *data flow* saja, baik itu *data flow* yang menuju ke suatu entitas ataupun proses, maupun juga *data flow* yang berasal dari suatu entitas ataupun proses.



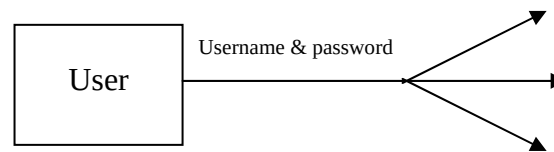
Gambar 2.3.1. Single Flow

- Gambar 2.3.2. adalah gambar *data flow* yang jumlahnya lebih dari satu dalam suatu entitas ataupun proses.



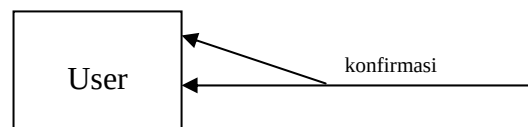
Gambar 2.3.2. Flow lebih dari satu

- Gambar 2.3.3. adalah gambar *data flow* yang bercabang (*diverge*) dari suatu proses ataupun entitas.



Gambar 2.3.3. Diverge Flow

- Gambar 2.3.4. adalah gambar *data flow* yang menyatu (*converge*) jadi satu, dari beberapa *data flow* yang berbeda.



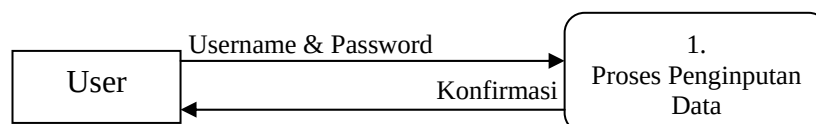
Gambar 2.3.4. Converge Flow

- Gambar 2.3.5. adalah gambar *data flow* yang memiliki dua arah dalam satu *data flow*, disebut dengan *two-way flow*.



Gambar 2.3.5. Two Way Flow dengan satu data flow

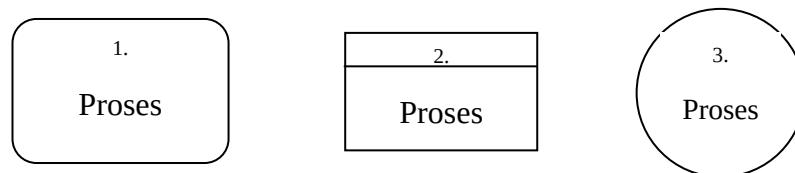
- Gambar 2.3.6. adalah gambar dua *data flow* yang memiliki dua arah. *Data flow* ini sama dengan gambar 2.3.6 diatas, hanya digambarkan dalam dua *data flow*.



Gambar 2.3.6. Two Way Flow dengan dua data flow

c. *Transformation Process* (proses)

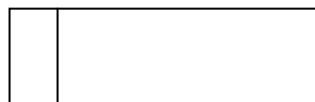
Transformation Process berfungsi untuk menggambarkan proses dari sistem yang menerima data dan mengeluarkan data lain yang ada didalam sistem (mengubah *input* menjadi *output*). Simbol dari *transformation processes* digambarkan pada gambar 2.4.



Gambar 2.4. Simbol *Transformation Processes*

d. *Data Store*

Data Store berfungsi sebagai media penyimpanan data. Yang dapat menyimpan dan mengeluarkan data dari suatu sistem jika diperlukan. Simbol dari *data stores* digambarkan pada gambar 2.5.



Gambar 2.5. Simbol *Data Store*

2.7. *Entity Relationship Diagram*

2.7.1. Definisi ERD

Entity Relationship Diagram (ERD) adalah diagram yang menggambarkan hubungan antara *entity key* (*entity*) yang satu dengan yang lain. (McLeod, 2000).

2.7.2. Tipe *Entity*

Ada beberapa tipe *entity*, yaitu *environmental element*, *resource* ataupun transaksi yang sangat penting bagi perusahaan sehingga dimodelkan dalam data. Contoh *entity* adalah *customer*, pegawai, daftar presensi dan sebagainya.

Setiap *entity* memiliki atribut didalamnya. Atribut adalah karakteristik dari setiap *entity* tersebut. Dan setiap atribut memiliki nilai atribut (*Attribute value*), yaitu nilai dari setiap atribut tersebut untuk setiap *entity*. Contoh atribut dari master_jemaat adalah id_jemaat, nama, alamat dan sebagainya. Sedangkan

contoh nilai atribut dari id_jemaat adalah jemaat001, jemaat002, dan sebagainya. Id_jemaat sebagai *identifiernya*. *Identifier* adalah atribut yang mengidentifikasi sebuah *entity* secara unik, dimana tidak boleh ada data yang sama dalam satu *entity*. Dalam ERD, *identifier* ditulis dengan garis bawah.

2.7.3. Simbol *Entity*

Entity digambarkan dalam simbol persegi empat, dimana nama *entity* ditulis didalamnya, contoh seperti pada gambar 2.6.



Gambar 2.6. Contoh *Entity*

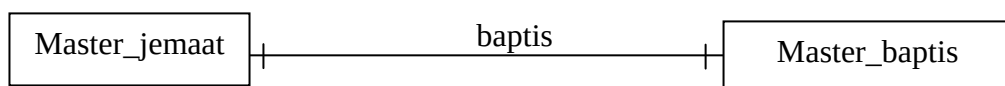
Setiap *entity* memiliki relasi dengan *entity* yang lain. Hubungan yang terjadi antara dua buah *entity* disebut *relationship*. Dalam ERD, *relationship* digambarkan dengan simbol belah ketupat.

2.7.4. Hubungan/relasi ERD

Ada 3 macam hubungan/relasi yang terjadi antara dua *entity*, yaitu:

a. *One to One Relationship*

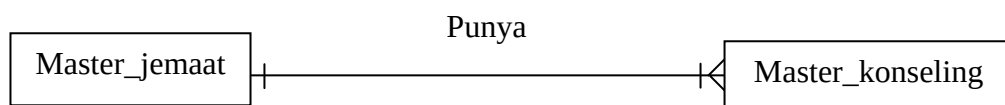
Hubungan antara satu *entity* dengan satu *entity* yang lain yang berbeda. Simbol dari *one to one relationship* digambarkan pada gambar 2.7.



Gambar 2.7. *One to One Relationship*

b. *One to Many relationship*

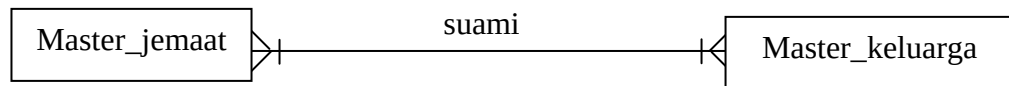
Hubungan dimana satu anggota *entity* mempunyai hubungan dengan beberapa anggota *entity* lain yang berbeda. Simbol dari *one to many relationship* digambarkan pada gambar 2.8.



Gambar 2.8. *One to many relationship*

c. *Many to Many relationship*

Hubungan dimana beberapa *entity* mempunyai hubungan dengan beberapa anggota *entity* yang lain pada *entity* yang berbeda. Simbol dari *many to many relationship* digambarkan pada gambar 2.9.



Gambar 2.9. *Many to many relationship*

Hubungan antara satu *entity* dengan satu *entity* yang lain yang berbeda.

Ada 2 macam, yaitu:

1. *Obligatory*

Bila semua anggota dari suatu *entity* harus berpartisipasi atau mempunyai hubungan dengan *entity* yang lain.

2. *Non Obligatory*

Bila semua anggota *entity* tidak harus mempunyai hubungan dengan anggota *entity* yang lain.

2.8. Sistem Informasi Manajemen

Salah satu sumber daya yang tersedia bagi seorang manager adalah informasi, dimana informasi ini dapat dikelola seperti sumber daya yang lain yang membentuk suatu sistem informasi sesuai dengan konsep dasar informasi. Agar suatu sistem dapat dikenal dengan baik, maka sistem tersebut harus dipelajari. Sistem didefinisikan sebagai kumpulan dari beberapa elemen yang berinteraksi untuk mencapai tujuan tertentu. Suatu sistem mempunyai susunan dasar, antara lain: *input*, *output*, transformasi, mekanisme pengendalian, dan tujuan. (McLeod, 2000).

Sistem informasi manajemen adalah sistem informasi pada tingkat manajemen dari suatu organisasi yang menyajikan fungsi dari perencanaan, pengawasan, dan proses pengambilan keputusan dengan menyediakan ringkasan rutin dan laporan pengecualian. Dimana sistem ini dapat memperlengkapi

manager dengan laporan-laporan ataupun akses *on-line* tentang keadaan organisasi sekarang ini dan juga catatan sejarah organisasi. Sistem informasi manajemen ini juga menyediakan jawaban dari pertanyaan rutin yang sudah ditetapkan sebelumnya dan sudah disediakan sebuah prosedur yang sudah didefinisikan sebelumnya sebagai jawabannya. Jadi sistem informasi manajemen menyediakan laporan-laporan berkala dan sudah terdefinisi yang meringkaskan informasi dengan database untuk mempermudah penyediaan informasi.

2.9. MySQL

MySQL adalah suatu *software* sistem manajemen *database* yang menggunakan standar SQL (*Structured Query Language*). SQL adalah bahasa *query* generasi keempat yang sudah distandarisasi dan banyak ditemukan dalam lingkungan *database*. Struktur dasar dari SQL yang terstandarisasi adalah

SELECT ... FROM ... WHERE

Kata *select* digunakan untuk menuliskan atribut yang ingin ditampilkan dalam sebuah *query*.

Kata *from* digunakan untuk menuliskan nama-nama tabel relasi yang dipakai dalam *query*.

Kata *where* digunakan sebagai predikat (persyaratan dalam *query*) yang terdiri dari atribut dari tabel relasi yang ada pada *klausula from*.

Sebagai *database server* yang memiliki konsep *database modern*, MySQL memiliki banyak sekali keistimewaan yang digunakan untuk menambahkan, mengakses dan memproses data yang disimpan di komputer. Berikut ini adalah keistimewaan yang dimiliki oleh MySQL.

- ❑ MySQL dapat berjalan stabil pada berbagai sistem operasi (*portability*).
- ❑ MySQL didistribusikan secara *open source* (gratis), sehingga dapat digunakan *oleh user* tanpa dipungut biaya.
- ❑ MySQL dapat digunakan secara bersamaan oleh beberapa *user* tanpa mengalami masalah (*multiuser*).
- ❑ MySQL memiliki kecepatan yang baik dalam menangani *query* sederhana.

- ❑ MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed/unsigned integer, float, double, char, varchar, text, blob, date, time, datetime, timestamp, year, set* serta *enum*.
- ❑ MySQL memiliki operator dan fungsi yang mendukung penggunaan perintah *SELECT* dan *WHERE* dalam *query*.
- ❑ MySQL memiliki beberapa lapisan sekuritas, izin akses *user* dengan perizinan yang mendetail serta *password* yang terenkripsi.
- ❑ MySQL mampu menangani *database* dalam skala besar.
- ❑ MySQL mampu mendeteksi pesan kesalahan pada *client* dalam banyak bahasa (namun Bahasa Indonesia belum termasuk didalamnya).
- ❑ MySQL memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE*, dibandingkan *database* lain.

Dengan kata lain, MySQL memiliki konektivitas, kecepatan, dan sekuriti yang baik, sehingga cocok digunakan sebagai *database* dalam internet. (Rickyanto, 2002).