2. TEORI DASAR

2.1. HTML

Hypertext Markup Language (HTML) merupakan *standard* bahasa yang digunakan untuk menampilkan dokumen *website*, yang dapat dilakukan dengan HTML, yaitu (Karrow, 2000):

- Mengontrol tampilan dari halaman *website* dan isi-nya
- Mempublikasikan dokumen secara *online* sehingga bisa diakses dari seluruh dunia
- Membuat *online form* yang bisa digunakan untuk menangani pendaftaran, transaksi secara *online*
- Menambahkan objek-objek seperti *image, audio, video* dan juga *java applet* dalam dokumen HTML.

2.1.1. Tag-tag HTML

Command HTML biasanya disebut *TAG*, *TAG* digunakan untuk menentukan tampilan dari dokumen HTML.

```
<BEGIN TAG> </END TAG>
```

Contoh:

Setiap dokumen HTML diawali dan diakhiri dengan *tag* HTML.

```
<HTML>
. . .
<HTML>
```

Tag tidak *case sensitive*, sehingga dapat digunakan <HTML> atau <html> keduanya menghasilkan *output* yang sama. Bentuk dari *tag* HTML sebagai berikut:

```
<ELEMENT ATTRIBUTE = value>
```

Element - nama *tag*

Attribute - atribut dari *tag*

Value - nilai dari atribut.

Contoh:

```
<BODY BGCOLOR=lavender>
```

BODY merupakan elemen, **BGCOLOR** (*background*) merupakan atribut yang memiliki nilai **lavender**.

2.1.2. Struktur HTML Document

Dokumen HTML bisa dibagi menjadi tiga bagian utama:

2.1.2.1.HTML

Setiap dokumen HTML harus diawali dan ditutup dengan *tag* HTML

```
<HTML> </HTML>
```

tag HTML *membe*ri tahu *browser* bahwa yang di dalam kedua *tag* tersebut adalah dokumen HTML.

2.1.2.2.*HEAD*

Bagian *header* dari dokumen HTML diapit oleh *tag* <HEAD></HEAD> di dalam bagian ini biasanya dimuat *tag* TITLE yang menampilkan judul dari halaman pada *title*-nya *browser*.

Selain itu *Bookmark* juga mengunakan *tag* TITLE untuk *membe*ri *mark* suatu *website*. *Browser* menyimpan "titile" sebagai *bookmark* dan juga untuk keperluan pencarian (*searching*) biasanya *title* digunakan sebagai *keyword*.

Header juga memuat *tag* META yang biasanya digunakan untuk menentukan in*Form*asi tertentu mengenai dokumen HTML, dan dapat juga digunakan untuk menentukan *author name*, *keywords*, dan lainyan pada *tag* META.
Contoh:

```
<META name="Author" contents="Bocah Gunung">
```

Author dari dokumen tersebut adalah "Bocah Gunung"

Atribut http-equiv dapat di gunkan untuk meletakkan nama HTTP *Server* atribut untuk menciptakan HTTP header.
Contoh:

```
<META http-equiv="Expires" content="Wed, 7 May
    2003 20:30:40 GMT">
```

yang akan menciptakan HTTP header:

Expires: Wed, 7 May 2003 20:30:40 GMT

Sehingga jika dokumen-dokumen di *cache*, HTTP akan megetahui kapan untuk mengubah dokumen tersebut pada *cache*.

2.1.2.3.*BODY*

Dokumen body di gunakan untuk menampilkan *text*, *image link* dan semua yang akan di tampilkan pada halaman *website.*

```
<html>
     <head>
          <title>Welcome to HTML</title>
     </head>
     <body bgcolor="laveder">
          <p>Document HTML yang Pertama</p>
     </body>
</html>
```

2.2. CSS

Style Sheets merupakan fitur yang sangat penting dalam membuat *Dynamic* HTML. Meskipun bukan merupakan suatu keharusan dalam membuat *website*, akan tetapi penggunaan *style sheets* merupakan kelebihan tersendiri.

Suatu *style sheet* merupakan tempat untuk mengontrol dan memanage *style-style* yang ada. *Style sheet* mendeskripsikan bagaimana tampilan dokumen HTML di layar.

Style sheet juga dapat disebut sebagai *template* dari dokumen-dokumen HTML yang menggunakanya. Dapat juga dibuat efek-efek spesial di *website* dengan menggunakan *style sheet*. Sebagai contoh, dapat digunakan untuk membuat *style sheet* yang mendefinisikan *style* untuk <H1> dengan *style bold* dan *italic* dan berwarna biru. Atau pada *tag* <P> yang akan di tampilkan dengan warna kuning dan menggunakan huruf verdana dan masih banyak lagi yang bisa dilakukan dengan menggunkan *style sheet*.

Teknologi *Cascading Style Sheet* (CSS) di-*support* pada hampir semua *Web Browser*. Karena CSS telah distandartkan oleh *World Wide Web Consortium* (W3C) untuk di gunakan di *Web Browser*.

2.2.1. *Inline Styles*

Ada dua cara untuk dapat merubah *style* dari halaman website yaitu dengan:

? Merubah *inline style*

? Menulis *script* untuk merubah *style*.

Dengan menggunakan *inline style,* maka *dynamic style* dapat dibuat tanpa harus menambahkan *script* ke dalam *website*. *Inline styles* merupakan *style* yang bisa dipasang pada elemen *website* tertentu saja.

Jika ingin menambahkan *style* pada `<H1>` dengan warna merah, harus dilakukan pengesetan attribut `STYLE` dari *tag* `<H1>`.

```
<H1 STYLE=”color:red”>
```

Jika ingin menggunakan *script* untuk memodifikasi *inline style*, maka dapat dilakukan dengan menggunakan *Style Object. Style Object* men-*support* semua *property* yang di-*support* CSS untuk *style*. Untuk menggunakan *property* pada *script*:

? Hilangkan tanda hubung “-” dari *property* CSS Style

? Ganti huruf setelah tanda hubung menjadi Capital.

Contoh:

font-weight menjadi **fontWeight**

text-align menjadi **textAlign**

2.2.2. Komentar dalam *Style Sheets*

Komentar biasanya digunakan oleh *programmer* untuk memudahkan mengingat kembali *script* yang sudah ditulis sebelumnya. Komentar di CSS hampir sama dengan komentar di C atau C++ yaitu dengan menggunakan:

```
/* isi Comments */
```

Contoh:

```
/* H1 elements akan menjadi biru */
H1 { color:blue;}
/* H1 elements akanmenjadi biru */
Tags.H1.color = “blue”;
```

2.2.3. Pemakaian elemen *style*

Pemakaian elemen *style* dapat dibuat dengan mengatur warna huruf dan latar belakang. Dapat dikerjakan dengan menggunakan elemen *style* untuk mengatur karakter kode *tag* dokumen.

```
<style type="text/css">
      body { color: black; Background: white; }
</style>
```

Pernyataan yang ditulis antara kode *tag* `<style>` dan `</style>` menunjukkan perintah pengaturan *style*.

2.2.3.1.*Link ke sheet* lainnya

Apabila menginginkan *style* yang sama untuk halaman HTML yang lain, disarankan mempergunakan *sheet-sheet* terpisah namun satu dan lainnya terhubung dengan cara *link*. Dapat dibuat dengan mengikuti cara berikut ini:

```
<link rel="stylesheet" href="style.css">
```

Kode *tag* untuk *link* ini ditempatkan dibagian "`head`" dari dokumen. Perintah `rel` perlu diatur dengan pernyataan "`stylesheet`" agar supaya *browser* dapat menemukan perintah `href` sebagai penunjukan ke alamat *website* (URL) *sheet*.

2.2.3.2.Mengatur tepi halaman *(page margin)*

Halaman *website* akan tampil cantik bila dituliskan dalam margin yang lebih lebar. Dapat mengatur sisi kiri dan kanan dengan memakai karakter "`margin-left`" dan "`margin-right`".

Contoh:

```
<style type="text/css">
  body { margin-left: 10%; margin-right:10%; }
</style>
```

Perintah di atas dituliskan dengan tujuan agar tampilan halaman *website* di layar *monitor* memiliki batas halaman kiri 10% dari lebar layar *monitor* dan batas halaman kanan 10% dari lebar layar *monitor*.

2.2.3.3. Mengatur *indent* kiri dan kanan

Agar halaman *website* tampil lebih cantik bisa juga diberikan *indent* (*spasi*) dari *margin* kiri beberapa huruf sebelum menuliskan awal kalimat.

Contoh:

```
<style type="text/css">
      body { margin-left: 10%; margin-right: 10%; }
      h1 { margin-left: -8%;}
      h2,h3,h4,h5,h6 { margin-left: -4%; }
</style>
```

2.2.3.4. Mengatur jarak penulisan dari tepi atas dan bawah halaman

Program *browser* biasanya mengerjakan batas atas dan bawah, paragraf dan lain-lain dengan baik. Namun ketika ingin membuat ruang di atas atau bawah halaman *website*, atau terdapat keinginan membuat *spasi* yang khusus, barulah disini diperlukan cara untuk mengaturnya.

Properti "margin-top" menentukan ruang sebelah atas dan properti "margin-below" menentukan ruang sebelah bawah halaman *website*. Bila hendak mengatur semuanya dengan *heading* h2, cukup dilakukan dengan menuliskan dengan perintah HTML sebagai berikut:

```
h2
{
     margin-top: 8em;
     margin-bottom: 3em;
}
```

Kode *em* merupakan unit penting dalam mengatur ukuran tinggi *font* (huruf). Ini lebih mudah bila dibandingkan dengan pengaturan *pixel* atau titik (*point*). Unit ini akan sangat berguna pada pembuatan huruf besar.

Satuan titik (*point*) umumnya dipergunakan oleh *program word processor* misalnya dituliskan ukuran huruf 10 pt. Sayangnya untuk ukuran titik yang sama, menghasilkan ukuran huruf yang berbeda pada pemakaian *program browser* yang berbeda pula.

Untuk mengatur ruang sebelah atas bagian *heading* halaman *website*, sebaiknya dibuat dengan nama *style* untuk *heading* tersebut. Dalam penulisan HTML-nya cukup dengan menggunakan atribut *class*.

Contoh:

```
<h2 class="subsection">Getting started</h2>
```

Kemudian pengaturan ruangnya ditulis dengan perintah berikut:

```
h2.subsection
{
     margin-top: 8em;
     margin-bottom:3em;
}
```

Pengaturan ini dimulai dengan nama *tag*, sebuah titik dan kemudian nilai dari atribut *class*. Hati-hati dalam menempatkan ruang sebelum atau sesudah titik tersebut. Bila pengaturan tersebut tidak *member*ikan hasil. Ada cara lain untuk mengatur *style* elemen tertentu, tetapi atribut *class* tetap bersifat fleksibel.

Pada saat sebuah "heading" diikuti dengan sebuah paragraf, nilai untuk batas bawah (margin-bottom) untuk *heading* tersebut tidak ditambahkan dengan nilai batas atas (margin-top) paragraf.

2.2.3.5.*Indent* pada baris pertama

Kadang-kadang terdapat keinginan untuk membuat inden pada baris pertama tiap paragraf dapat dilakukan dengan cara berikut:

```
P
{
     text-indent: 2em;
     margin-top: 0;
     margin-bottom:0;
}
```

Cara tersebut akan membuat inden pada baris pertama paragraf sejauh 2 *em* dan *member*ikan jarak antar paragraf.

2.2.4. Mengatur *Form*at *Font*

*Form*at *font* dapat diatur dengan mengubah model huruf, mengatur ukuran huruf, mengatur jenis huruf, menambahkan *border* dan latar belakang.

2.2.4.1.Model Huruf

Model yang umum dipakai adalah teks miring atau tebal. Umumnya *program browser* mempergunakan *tag* em untuk huruf *italic* dan *tag strong* untuk

huruf tebal. Misalnya ketika ingin menuliskan kode em agar huruf tampil berbentuk *italic* dan tebal dan menuliskan kode *strong* untuk huruf tebal *uppercase*, perintahnya dituliskan sebagai berikut:

```
em
{
     font-style: italic;
     font-weight: bold;
}
strong
{
     text-transForm: uppercase;
     font-weight:bold;
}
```

bila gagal dapat ditambahkan perintah ini:

```
h2
{
     text-transForm: lowercase;
}
```

2.2.4.2. Mengatur ukuran huruf

Kebanyakan program *browser* mempergunakan huruf yang lebih besar untuk *heading* yang penting sifatnya. Bila menimpa ukuran *default*-nya, akan menempuh resiko yaitu huruf menjadi tampak lebih kecil khususnya bila mempergunakan ukuran yang telah ditambahkan dengan ukuran titik. Sehingga disarankan untuk melakukan pengaturan ukuran huruf dengan ukuran yang sama.

Contoh berikut mengatur ukuran *heading* dalam persen relatif terhadap ukuran teks normal.

```
h1 { font-size: 200%; }
h2 { font-size: 150%; }
h3 { font-size: 100%; }
```

2.2.4.3. Mengatur jenis huruf.

Bisa saja jenis huruf yang favorit tidak bisa ditampilkan oleh berbagai jenis *browser*. Untuk mengatasi hal ini dapat dilakukan dengan menuliskan beberapa jenis huruf yang tidak dapat ditampilkan oleh hampir semua *browser*. Ada beberapa jenis huruf generik yang dijamin cocok, sehingga disarankan untuk mengakhiri daftar

perintah HTML dengan salah satu jenis huruf berikut: serif, sans-serif, cursive, fantasy, atau monospace, contoh:

```
body
{
     font-family: Verdana, sans-serif;
}
h1,h2
{
     font-family: Garamond, Times New Roman,serif;
}
```

Dalam contoh ini *heading* penting akan ditampilkan dalam bentuk huruf Garamond, bila gagal maka akan ditampilkan dalam bentuk Times New Roman, dan bila juga masih tidak dapat tampil, maka akan tampil sebagai huruf serif. Teks paragraf akan ditampilkan dengan huruf Verdana atau bila masih tidak tampil juga, maka *browser* masih dapat menampilkannya dengan jenis huruf standar sans-serif.

2.2.4.4. Cara menghindari masalah huruf dan batas tepi halaman web.

Pertama pergunakan elemen p untuk mencegah teks pada body tulisan yang tidak dapat ditampilkan dengan baik. Contoh:

```
<h2>Spring in Wiltshire</h2>
Blossom on the trees, bird song and the sound of
Lambs bleating in the fields.
```

Teks yang ditulis mengikuti sebuah *heading* dapat menimbulkan akibat tampilan jenis huruf yang berbeda pada beberapa jenis *program browser* maka disarankan segera menutup teks pada paragraf tersebut. Contoh:

```
<h2>Spring in Wiltshire</h2>
<p>Blossom on the trees, bird song and the sound
of lambsbleating in the fields.</p>
```

Kedua pergunakan selalu elemen pre ketika menuliskan pengaturan jenis huruf yang digunakan. Contoh:

```
pre
{
     font-family: monospace;
}
```

Ketiga pergunakan elemen p dan ul pada waktu mengatur jenis huruf untuk *heading*. Khususnya ketika sedang melakukan pengaturan *border* atau warna

halaman *website* dengan elemen div. Beberapa jenis *program browser* tidak dapat melakukan pengaturan huruf dengan baik dan cenderung lupa sehingga huruf *heading* tampak menjadi huruf standar saja. Untuk menghindari hal tersebut dapat menuliskan perintah HTML sebagai berikut:

```
h1,h2,h3,h4,h5,p,ul
{
     font-family: sans-serif;
}
```

Menambahkan *border* dan latar belakang.

Menambahkan *border* disekitar *heading* dapat dilakukan dengan mudah, daftar (*list*), atau paragraf atau sekelompok *heading*, *list* dan paragraf secara tertutup dengan mempergunakan elemen div.

Contoh:

```
div.box
{
     border: solid;
     border-width: thin;
     width:100%
}
```

Catatan: tanpa *property* "width" beberapa *browser* akan menempatkan tepi kanan terlalu jauh ke arah kanan. Untuk mencegah hal ini dapat dilakukan dengan menuliskan perintah HTML-nya sebagai berikut:

```
<div class="box">
      The content within this
      DIV element will be enclosed
      in a box with a thin line around it.
</div>
```

Ada sedikit jenis *border* yaitu: dotted, dashed, solid, double, groove, ridge, inset dan outset. Lebar border diatur dengan mempergunakan *property* "border-width". Nilai dari *property* ini yaitu *thin*, *medium* dan *thick* yang tampak setipis ukuran 0.1em. *Property* "border-color" memungkinkan untuk mengatur warna.

Sebuah efek yang cantik dapat dilakukan dengan *member*ikan warna latar belakang kotak dengan warna tebal atau dengan hamburan gambar ("tile image").

Untuk melakukan hal tersebut perlu mempergunakan *property* "*background*". Dapat dilakukan dengan mengikuti perintah berikut ini.

```
div.color
{
      Background: #FEFEFF;
      padding: 0.5em;
      border: none;
}
```

Tanpa pengaturan *property border*, biasanya *program browser* hanya akan menampilkan warna standar saja. *Property padding member*ikan beberapa ruangan di antara tepi-tepi daerah berwarna dan teks yang terdapat di dalamnya.

Mengatur nilai *property padding* dapat dilakukan dengan menambahkan *padding-left, padding-top, padding-right* dan *padding-bottom*. Pengaturan ini dituliskan misalnya sebagai: `padding-left: 1em.`

Jika dalam pembuatan *website* hanya dibutuhkan *border* pada satu sisi halaman *website* saja dapat dilakukan dengan pengontrolan tiap sisi *border* dengan *member*ikan keterangan *border-left, border-top, border-right* dan *border-bottom*.

Contoh:

```
p.changed
{
     padding-left: 0.2em;
     border-left: solid;
     border-right: none;
     border-top: none;
     border-bottom: none;
     border-left-width: thin;
     border-color: red;
}
```

Susunan perintah diatas *member*ikan efek pada tampilan *website* dengan *border* berwarna merah disisi sebelah kiri.

Mengatur warna

Contoh berikut adalah perintah pengaturan warna.

```
body
{
```

```
    color: black;
    Background: white;
}
strong
{
    color: red
}
```

Model pengaturan diatas *member*ikan warna hitam teks (default) dan latar belakang putih, tetapi memiliki elemen *strong* pada warna merah. Ada 16 buah nama warna standar, selain itu dapat juga dengan mempergunakan nilai desimal untuk warna merah, hijau, biru, dan masing-masing memiliki interval antara 0 sampai 255. Misalnya rgb (255, 0, 0) akan *member*ikan warna merah di layar monitor. Serta dapat juga mempergunakan angka *hexadesimal* yang dimulai dengan karakter # yang diikuti enam angka *hexadesimal* sebagai pengaturan warna. Sebuah pengubah juga diberikan dibawah ini agar dapat melakukan pengubahan nilai dari RGB ke nilai *hexadesimal*.

Mengatur warna *link*

Untuk mengatur warna *link hypertex* dapat mempergunakan, dengan warna yang berbeda untuk *link* yang belum pernah diakses, *link* yang pernah diakses dan *link* yang kemudian akan diakses serta *link* yang aktif. Bahkan untuk pengaturan warna dapat dilakukan ketika kursor *mouse* berada diatas daerah yang akan di*link*. Hal ini dapat dituliskan dalam bentuk perintah seperti berikut:

```
/* untuk warna link yang belum diakses */
:link { color: rgb(0, 0, 153) }

/* untuk warna link yang telah diakses */
:visited { color: rgb(153, 0, 153) }

/* untuk warna link ketika link diklik */
:active { color: rgb(255, 0, 102) }

/* untuk warna link ketika mouse diatasnya*/
:hover { color: rgb(0, 96, 255) }
```

Pada *website* biasanya ingin ditampilkan *link hyperteks* tanpa garis bawah dapat dilakukan dengan *member*ikan *property textdecoration* atau *none*.

Contoh:

```
a.plain { text-decoration: none }
```

Contoh berikut juga menampilkan *link* yang tidak bergaris bawah.

```
This is <a class="plain" href="what.html"> not underlined
</a>
```

Kebanyakan orang ketika mereka melihat garis bawah dibawah sebuah *link* selalu mengira itu adalah bagian teks yang diberi *link*. Umumnya orang *member*ikan warna biru pada teks yang diberi *link* ke halaman atau alamat internet lain. Jika warna latar belakang menyebabkan teks yang diberi *link* menjadi sulit terbaca maka sebaiknya tidak digunakan warna link yang mendekati warna *background*.

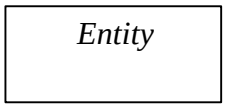
2.3. Entity Relationship Diagram (ERD)

ERD adalah representasi grafik dari data yang terdapat pada suatu sistem atau organisasi. ERD digunakan untuk mengidentifikasi data yang akan diambil, disimpan, dan digunakan dengan tujuan untuk mendukung aktivitas bisnis suatu organisasi. Selain itu, ERD juga digunakan untuk mengidentifikasi data yang dibutuhkan untuk melakukan *monitoring* jalannya sistem. ERD mempunyai tiga komponen utama, yaitu (Ramez, 2003):

2.3.1. *Entity*

Entity berupa orang, tempat, konsep, atau kejadian yang dianggap penting bagi perusahaan atau organisasi. Dengan kata lain, *entity* adalah semua hal yang datanya dianggap penting oleh perusahaan.

Simbol *entity* adalah



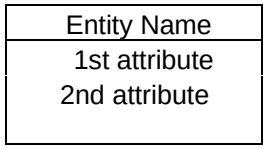
Gambar 2.1. *Entity*

Sumber: Ramez, Elmasri. (2003). *Fundamentals Of Database System*, p.72

2.3.2. *Attribute*

Attribute adalah elemen atau karakteristik yang dimiliki oleh sebuah *entity*.

Simbol *attribute* adalah

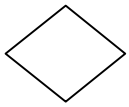


Gambar 2.2. *Attribute*

Sumber: Ramez, Elmasri. (2003). *Fundamentals Of Database System*

Relationship

Relationship adalah hubungan antara *entity*. Simbol *relationship* adalah



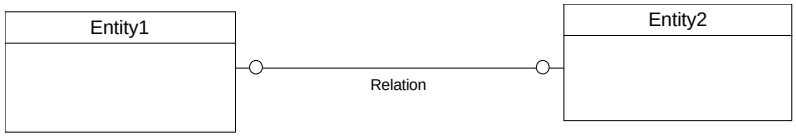
Gambar 2.3. *Relationship*

Sumber: Ramez, Elmasri. (2003). *Fundamentals Of Database System*, p.72

Terdapat beberapa jenis *relationship*, yaitu:

? *Cardinality*, mengGambarkan jumlah *entity* yang muncul dalam suatu relasi. Ada 2 nilai *cardinality*, yaitu 1 (*one*) atau N (*many*). Bentuk relasi yang dihasilkan adalah:

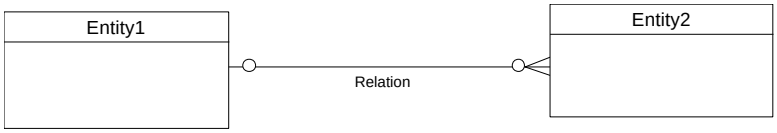
1. *One-to-one relationship*



Gambar 2.4. *One-to-one relationship*

Sumber: Ramez, Elmasri. (2003). *Fundamentals Of Database System*

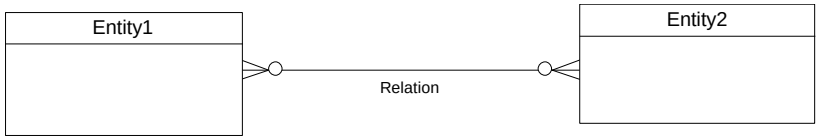
2. *One-to-many relationship*



Gambar 2.5. *One-to-many relationship*

Sumber: Ramez, Elmasri. (2003). *Fundamentals Of Database System*

3. *Many-to-many relationship*

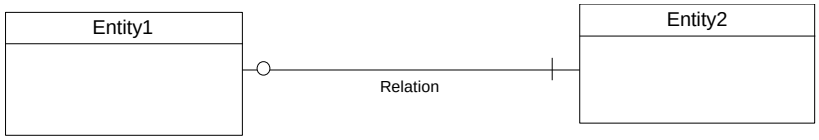


Gambar 2.6. *Many* to *many relationship*

Sumber: Ramez, Elmasri. (2003). *Fundamentals Of Database System*

? *Mandatory*, berfungsi untuk menandai apakah semua anggota suatu *entity* harus berelasi dengan anggota *entity* lain atau tidak. Bila semua anggota harus berelasi maka diberi simbol "|" atau disebut juga *mandatory/obligatory* sedangkan bila semua anggota tidak harus berelasi maka diberi simbol "o" atau disebut *non mandatory / non obligatory.*

Berikut ialah contoh *mandatory entity* 1 *to entity* 2 dengan model relasi *one-to-one relationship.*



Gambar 2.7. *Mandatory*

Sumber: Ramez, Elmasri. (2003). *Fundamentals Of Database System*

Dari Gambar tersebut berarti semua anggota dari *entity* 1 harus berelasi dengan anggota dari *entitiy* 2 tetapi anggota dari *entitiy* 2 tidak harus berelasi seluruhnya dengan anggota dari *entitiy* 1.

2.4. MySQL

MySQL dipublikasikan sejak tahun 1996, akan tetapi sebenarnya dikembangkan sejak tahun 1979. MySQL telah memenangkan penghargaan *Linux Journal Reader's Choice Award* selama tiga tahun. MySQL sekarang tersedia di bawah lisensi *open source*, tetapi ada juga lisensi untuk penggunaan MySQL yang bersifat komersial. Keunggulan dari MySQL ini adalah:

? Bersifat *open source.*

? Sistem *software*-nya tidak *member*atkan kerja *server* atau komputer karena dapat bekerja di *background*.

MySQL adalah *database client-server* yang merupakan aplikasi *Relational Database Management Server* (RDBMS) yang bersifat *open source*. MySQL menggunakan *Structured Query Language* (SQL) yang merupakan bahasa standar pemrograman *database*. Dengan menggunakan MySQL data dapat diakses oleh banyak pemakai secara bersamaan dan terdapat fasilitas untuk pembatasan akses para pemakai berdasarkan *previlage* (hak akses) yang diberikan. Untuk kemudahan pemakainya dalam proses pengubahan, penambahan, serta penghapusan data maka dapat digunakan fasilitas yang lebih *user friendly* yaitu MySQL *Front*.

Perintah-perintah MySQL

Pada MySQL terdapat beberapa perintah. Perintah-perintah pada MySQL ini hampir sama dengan *database server* yang lain. Perintah-perintah pada MySQL itu antara lain adalah sebagai berikut:

1. Create *database* digunakan untuk membuat *database* pada *database server*. Sintaksnya adalah:

```
create database database_name
```

   Database_name adalah nama *database* yang akan dibuat.

2. *Use database* digunakan untuk menunjuk pada *database* yang akan digunakan. Sintaksnya adalah:

```
use database_name
```

   *Database*_name adalah nama *database* yang akan digunakan.

3. Create table digunakan untuk membuat tabel pada suatu *database*. Sintaksnya adalah:

```
create table table_name
(     column_1 column_type column_attributes,
      column_2 column_type column_attributes,
      primary key (column_name),
      index index_name(column_name)
)
```

   Tabel_name adalah nama tabel yang akan dibuat.

Column_1 adalah nama kolom yang akan dibuat dalam tabel.

Column_type ✍ tipe dari kolom tersebut, dapat berupa: *char, varchar, tinytext, text, mediumtext, longtext, enum, int, tinyint, mediumint, bigint, float, double, decimal, date, datetime, timestamp, time, year*.

Column_attributes ✍ atribut kolom dapat berisi *null, not null*.

Tipe kolom yang digunakan adalah:

? *varchar(length)* ✍ digunakan untuk menyimpan karakter dengan maksimum panjang 255. variabel length digunakan untuk membatasi panjang karakter.

? *int* ✍ digunakan untuk menyimpan numerik.

? *date* ✍ digunakan untuk menyimpan tanggal dengan *Form*at YYYY-MM-DD.

? *Enum* ✍ digunakan untuk menyimpan karakter tertentu saja misalnya: *enum('T','F')* berarti hanya dapat menyimpan karakter T dan F saja.

4. *Insert* digunakan untuk menambahkan record pada tabel. Sintaksnya adalah:

```
insert into table_name (column1, column2,…)
Values (value1, value2,…)
```

Table_name ✍ nama tabel yang akan ditambahkan *record*-nya.

Column1, column2 ✍ kolom yang akan ditambahkan.

Value1,value2 ✍ data yang akan ditambahkan.

5. *Update* digunakan untuk mengubah *record* yang sudah ada pada tabel. Sintaksnya adalah:

```
update table_name set column1=value1, column2=value2
where column = value
```

Table_name adalah nama tabel yang akan diubah *record*-nya.

Column1, column2 adalah kolom yang akan ditambahkan.

Value1, value2 adalah data yang akan digantikan.

6. *Drop table* digunakan untuk menghapus tabel. Sintaksnya adalah:

```
drop table table_name
```

Table_name adalah nama tabel yang akan dihapus.

7. *Show table* digunakan untuk menampilkan tabel yang telah dibuat dalaam *database* yang aktif. Sintaksnya adalah:

```
show tables
```

8. *Show Field* digunakan untuk menampilkan seluruh *field* dalam suatu tabel. Sintaksnya adalah:

```
show field from table_name
```

Table_name adalah nama tabel yang akan ditampilkan *field*-nya.

9. *Alter table* digunakan untuk menambahkan, mengubah, dan menghapus *field* dalam suatu tabel. Sintaksnya adalah:

? Untuk menambahkan

```
alter table table_name add column column1 column_type
column_attributes
```

Table_name ✍ nama tabel yang akan ditambahkan *field*-nya.

Column1 ✍ nama *field* baru, *column_type* adalah tipe kolom dan *column attributes* adalah atribut kolom.

? Untuk mengubah

```
alter table table_name change column1 column2
column_type column_attributes
```

Table_name ✍ nama tabel yang akan diubah field-nya.

Column1 ✍ nama field yang akan diubah,

Column2 ✍ nama field baru column_type tipe kolom

Column attributes ✍ atribut kolom dari *field* baru.

? Untuk menghapus

```
alter table table_name drop column column1
```

Table_name ✍ nama tabel yang akan dihapus *field*-nya.

Column1 ✍ nama *field* yang akan dihapus.

2.5. J2ME

Java dikeluarkan oleh sebuah perusahaan yang bernama *Sun Microsystem* dalam tiga edisi yaitu (Topley, 2001):

? *Standard Edition* (J2SE): Didesain untuk dijalankan pada *personal computer*.

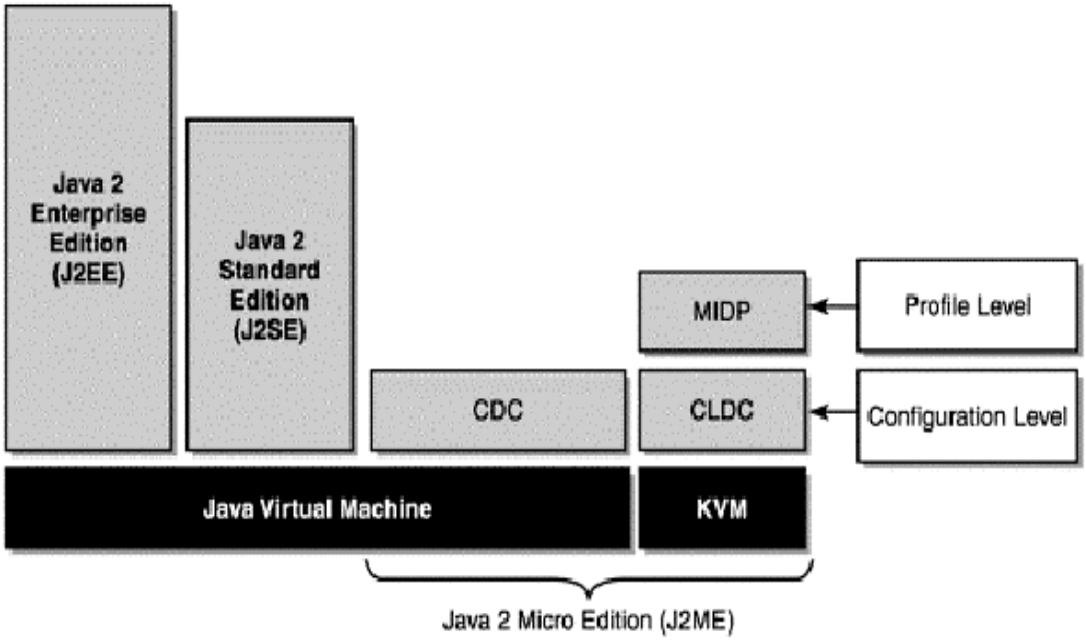
? *Enterprise Edition* (J2EE): Dengan fitur *built in* untuk *Servlet*, JSP, and XML, edisi ini didesain untuk aplikasi berbasis *server*.

? *Micro Edition* (J2ME): Didesain untuk *device* yang memiliki keterbatasan memori, tampilan dan proses seperti *handphone* atau PDA (*Personal Digital Assistant*)

Sebagian orang telah mengenal *applet* sebagai aplikasi *java* yang berjalan pada internet yang bersifat *client side* atau *servlet* yang bersifat *server side*. Sedangkan untuk aplikasi pada edisi J2ME diberi nama MIDlet yang didefinisikan dari *profile* dan *configuration* untuk menjangkau bermacam-macam pengguna dan *device* yang beragam.

J2ME merupakan kumpulan dari spesifikasi yang mendefinisikan sekumpulan *platform*, setiap spesifikasi akan dapat mengakomodasi kebutuhan sejumlah *platform* dalam *scope* tertentu yang sama. Setiap subset dari *Java programming environment* untuk beberapa perangkat tertentu didefinisikan ke dalam satu atau lebih *profiles*, dimana setiap *profiles* merupakan pengembangan kemampuan lebih lanjut dari *configuration*. Penentuan *configuration* dan *profiles* suatu *device* bergantung pada lingkungan kerja *device* tersebut dan tujuan pemasaran dari *device* tersebut.

Berikut ini adalah Gambar diagram J2ME *platform* dibandingkan dengan *platform-platform* lain yang menggunakan *Java* sebagai programming language.



Gambar 2.8. Perbandingan J2ME dengan *Java* 2 *Platform* Lain

Sumber: Muchow, John W. (2001). *Core J2ME Technology & MIDP*. Prentice-Hall Inc,p.2

Configuration

Configuration adalah spesifikasi yang mendefinisikan *software environment* pada beberapa *varian device* yang tergantung dari:

? Tipe dan jumlah memori yang tersedia

? Tipe *processor* dan kecepatannya

? Koneksi *network* yang didukung

Configuration digunakan sebagai *platform* minimal untuk suatu *device* tanpa adanya fitur tambahan. Pada J2ME *configuration* dibagi menjadi:

? *Connected Limited Device Configuration* (CLDC)

1. Memiliki memori 128 *kilobytes* untuk menjalankan *java.*
2. Memiliki memori 32 *kilobytes* untuk pengalokasian memori pada saat *runtime*.
3. Keterbatasan *user interface*.
4. Biasanya memakai tenaga baterai.

   Memiliki *network connectivity* yang bersifat *wireless*, terbatas dan dengan *bandwith* kecil.

? *Connected Device Configuration (*CDC*)*

1. Memiliki minimal 128 *kilobytes* untuk menjalankan *java*.
2. Memiliki minimal 256 *kilobytes* untuk pengalokasian memori pada saat *runtime*.
3. Memiliki *network connectivity* yang bersifat *persistent* dan dengan *bandwith* besar.

Sebagai contohnya, pada CLDC 1.0 tidak dikenal tipe data *float* sedangkan pada CLDC 1.1 mulai diperkenalkan tipe data *float*. Tipe data *float* akan sangat banyak membantu *programmer* untuk melakukan perhitungan dengan hasil yang lebih tepat.

Profile

Profile menambahkan *class* tambahan yang mendukung fitur fitur yang diperlukan pada *device* tertentu atau pada segmen pasar yang berbeda. Kedua *configuration* yang ada mempunyai satu atau lebih *profile* dimana kadang suatu *profile* digunakan untuk mendukung *profile* yang lainnya. Beberapa *profile* yang ada antara lain:

? *Mobile Information Device Profile* (MIDP)

Profile ini menyediakan kemampuan untuk *networking*, penyimpanan data, dan *component user interface*. Karena MIDP didefinisikan untuk lingkungan yang serba terbatas maka *user interface* dan koneksi *networking* yang didukung sangatlah sederhana.

? PDA *profile* (PDAP)

PDA *profile* didefinisikan untuk PDA yang memiliki spesifikasi sedikit lebih tinggi dari pada MIDP. *Application Programming Interface* (API) yang disediakan digunakan pada segmen komputer kecil seperti Palm.

MIDP versi 1.0, sekalipun cukup membantu dalam membuat *software* yang bebas *platform*, ternyata masih kurang memuaskan para pengembang *software handphone*. Perkembangan teknologi *handphone* yang begitu pesatnya memacu para pengembang *software* untuk membuat *software* yang khusus digunakan untuk suatu merk dan tipe tertentu. MIDP 1.0 awalnya ditujukan untuk mengatasi problematika

ini, namun kenyataannya masih belum cukup dapat mendukung penggunaan fitur fitur khusus *handphone*.

Pada pertengahan November 2003, Sun mengeluarkan versi baru yaitu versi 2.0 yang lebih lengkap dibandingkan versi sebelumnya. Di samping masih mengandalkan beberapa fitur lama yang cukup stabil, misalnya dalam penanganan *user interface* dan *record management* dalam versi yang terbaru, beberapa fitur lama diperbarui dan diperlengkapi. Penambahan fitur yang sangat membantu dalam versi 2.0 adalah:

- *Multimedia*: memungkinkan untuk memainkan nada sederhana dengan menggunakan perintah playTone() yang disediakan pada *package javax*.microedition.media.Manager
- *Game* API: sangat membantu dalam pembuatan *game*.
- *Secure Networking*: pada versi sebelumnya masih sangat sederhana.

Record Management Store (RMS)

RMS merupakan tempat penyimpanan data pada MIDlet (*Mobile Information Device Application*). Data tidak dapat langsung disimpan pada suatu *file* karena *java* tidak diberi *permission* untuk mengakses *resource handphone* secara langsung. Dengan adanya *class* `javax.microedition.rms` maka suatu data dapat disimpan secara *persistent*.

Record Store merupakan *class* yang menangani kumpulan dari *record* dan seluruh akses ke *record* haruslah melalui perantaraan *class Record Store*. *Class* ini menjamin penyimpanan data yang *atomic* tanpa ada kemungkinan untuk terjadinya data *corrupt*.

Saat sebuah *record* dibuat, *Record Store* akan *member*i nomor pada setiap *record* dengan nomor *unique identifier* yang disebut sebagai *record* ID. Penambahan *record* pertama kali akan diberi ID 1, yang kedua dengan ID 2, dan seterusnya. *Record* ID bukanlah suatu *index*, penghapusan sebuah *record* tidak akan mengakibatkan *renumbering existing record*.

Tiap *Record Store* memiliki nama yang unik untuk membedakannya dengan *Record Store* yang lain. Pada MIDP 1.0, *Record Store* tidaklah dapat untuk

digunakan bersama sama antar MIDlet yang berbeda. MIDP 2.0 sudah mendukung penggunaan *Record Store* yang sama oleh MIDlet yang berbeda.

Jumlah penyimpanan data pada tiap *handphone* tidaklah sama dan bervariasi. Setiap MIDlet yang menggunakan RMS haruslah menspesifikan jumlah minimum dari penyimpanan yang diperlukan pada JAR (*Java Archive*) *manifest* dan *application descriptor*. Pengaturan nilai minimum yang berlebihan akan dapat mengakibatkan penolakan untuk *install* pada *device* karena *space* yang tersedia tidak sebanyak yang dibutuhkan.

Kecepatan akses pada *persistent* memori lebih lama dari pengaksesan data yang disimpan pada variabel. Pada beberapa *platform*, proses *writing* bisa membutuhkan waktu yang lama sehingga untuk *performance* yang lebih bagus digunakan *thread* yang berbeda dari *thread* utama.

2.6. JSP

Defenisi JSP

Java Server Pages (JSP) adalah teknologi *server-side web pages* berbasis *Java* yang berjalan di *Platform Java.* JSP merupakan bagian dari *Java 2 Enterprise Edition* (J2EE). JSP merupakan merupakan pengembangan dari *servlet* yang *member*ikan kemudahan dalam membuat halaman web yang dapat dikonversi menjadi *servlet* pada saat dijalankan sehingga membuat *developer* lebih mudah dalam membuat *web pages* yang menggunakan HTML dan *Java* secara bersamaan dalam satu halaman.

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 *Servlet* 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>
```

2.6.1.1. *SESSION*

Session merupakan sebuah identifier dari koneksi yang terjadi yang menyimpan data data tentang koneksi yang terjadi, identifier disimpan di *server* dimana *client* tidak bisa mengaksesnya secara langsung. Contoh penggunaan session di JSP adalah sebagai berikut:

```
<html>
<head>
      <title>UseSession</title>
</head>
<body>
<%
      // mengambil nilai count dari session
      Integer count = (Integer)session.getAttribute("COUNT");

      // jika count tidak ditememukan,
      // count dibuat dan dimasukkan ke session
      if ( count == null )
      {
            count = new Integer(1);
            session.setAttribute("COUNT", count);
      }
      else
      {
            count = new Integer(count.intValue() + 1);
            session.setAttribute("COUNT", count);
      }
      out.println("<b>Hello you have visited this site: "
      + count + " times.</b>");
%>
</body>
</html>
```

2.6.1.2.Koneksi ke MySQL

Untuk melakukan koneksi dari JSP ke database MySQL dapat digunakan *mysql-connector*. Cara untuk melakukan koneksi dari JSP ke *database* MySQL dengan *mysql-connector* yaitu (Leonardo,2003):

```
String con=
"jdbc:mysql://localhost:3306/ta?user="+"root"
+"&password="+"1";
try
{
      // melakukan pengecekan terhadap databse mysql
      Class.forName("com.mysql.jdbc.Driver");
      try
      {
            // mebuka koneksi ke MySQL
            Connection myConn =
            DriverManager.getConnection(con);

            ...

            // menutup koneksi
            myConn.close();
      }
      catch(SQLException sqlException)
      {
            // jika terdapat error pada syntax SQl
            sqlerror=”SQLException: " +
            sqlException.getMessage()+"<br> SQLState: " +
            sqlException.getSQLState()+"<br>VendorError: " +
            sqlException.getErrorCode()+"<br>";
      }
}
catch(ClassNotFoundException err)
{
      System.out.println(“Driver Tidak Ditemukan”);
}
```

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. Berikut ini penjelasan mengenai tiga komponen/elemen utama dalam JSP, yaitu (Goodwill, 2000):

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

1. *Direktif Page* <%@ page … %>, digunakan untuk mendefinisikan atribut yang penting bagi keseluruhan kode dalam halaman tersebut (untuk mengatur *setting* halaman JSP).
2. *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).
3. *Direktif Taglib* <%@ taglib uri=”taglibraryURI” prefix=”tagprefix” %>, digunakan untuk penggunaan *tag library* atau *tag* tambahan.

Table 2.1 Table Atribut pada *Directif Page*

No.	Atribut	Deskripsi	Default
1	Language	Mendefinisikan bahasa *scripting* yang digunakan.	*Java* <%@page languange = "*java*" %>
2	Extends	Digunakan untuk men-*generate superclass*.	<%@ page extends = "com.taglib …"%>
3	Import	Untuk mengimpor *package* atau *class Java* sebagaimana program *Java* umumnya.	
4	Session	Menentukan apakah halaman JSP menggunakan *HTTP session*.	TRUE
5	Buffer	Menentukan model *buffering* untuk *output stream* ke *client*. Apabila nilainya diisi none, maka tidak terjadi *buffering*.	Default buffering maks 8 kb.
6	AutoFlush	Apabila *true*, *output buffer* akan di-*flush*/dikeluarkan setelah *buffer* penuh/*full*. Apabila *false*, pesan *exception* akan dikeluarkan untuk mengindikasikan jika *buffer* telah penuh.	TRUE

Table 2.1 Table Atribut pada *Directif Page* (Lanjutan)

7	IsThreadSafe	Mendefinisikan tingkat keamanan mengenai masalah *threading* dari halaman JSP. Jika *false*, *request* akan diproses sebagai *single Thread*, berurutan sesuai urutan datangnya *request*.	TRUE
8	Info	Mendefinisikan *string* in*Form*asi yang dapat diperoleh dari implementasi *metode Servlet.getServletInfo*().	
9	ErrorPage	Mendefinisikan *URL* dari file JSP lain yang dipanggil, apabila terjadi *error* saat file JSP diproses.	
10	IsErrorPage	Mengindikasikan apakah halaman JSP ini merupakan halaman *error* dari halaman JSP lain.	FALSE
11	Content Type	Mendeskripsikan *encoding* karakter yang digunakan serta tipe MIME sebagai *response* dari halaman JSP. Penggunaan: "TYPE= tipe MIME" atau "TYPE=tipe MIME; charset=CHARSET" CHARSET adalah karakter *encoding* yang digunakan.	DefaultType adalah *text*/html dan nilai default CHARSET adalah ISO-8859-1

Sumber: Goodwill, James. (2000). *Pure JSP*. SAMS.

b. Elemen *scripting*

Elemen s*cripting* terdiri dari tiga macam, yaitu:

1. Skriplet (*Scriptlet tag*), digunakan untuk deklarasi, ekspresi dan kode lain (untuk menulis proses dalam JSP sehingga menghasilkan nilai-nilai yang diinginkan).
2. Deklarasi (*Declaration tag*), digunakan untuk mendeklarasikan variabel atau metode, baik tipe maupun nilai awal variabel.
3. Ekspresi (*Expression tag*), digunakan untuk ekspresi dalam *Java* dan menampilkannya sebagai *string* pada *browser* (untuk mencetak isi variabel).

c. *Action*

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

1. *Tag Action* Standar : *tag* yang didefinisikan dalam spesifikasi JSP.
2. *Custom Tag* : merupakan *tag* baru, dimana tiap *tag* memiliki fungsi dan kemampuan yang didefinisikan sendiri.