BAB II

LANDASAN TEORI

2.1. Konsep Dasar Web

2.1.1. *Website*

Menurut Hidayat (2010:2) mengemukakan bahwa: *"Website* atau situs dapat diartikan sebagai kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya, baik bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing-masing dihubungkan dengan jaringan-jaringan halaman".

Hubungan antara satu halaman *web* dengan halaman *web* yang lainnya disebut *Hyperlink*, sedangkan teks yang dijadikan media penghubung disebut *Hypertext*. Halaman-halaman sebuah situs *web* diakses dari sebuah URL *(Unified Resource Locator).* URL adalah alamat sebuah halaman *web*, yaitu halaman suatu dokumen atau program yang ingin ditampilkan atau digunakan.

1. Internet

Menurut Yuhefizar (2008:1) mengemukakan bahwa: "Internet merupakan jaringan global komputer dunia, besar dan sangat luas sekali dimana setiap komputer saling terhubung satu dengan yang lainnya dari satu negara ke negara lainnya dari seluruh dunia dan berisi berbagai macam informasi, mulai dari teks, gambar, *audio*, video, dan lainnya" .

Internet itu sendiri berasal dari kata *Interconnection Networking* yang berarti hubungan dari banyak jaringan komputer dengan berbagai tipe dan jenis, dengan menggunakan tipe komunikasi seperti telepon,

satelit, dan yang lainnya. Internet telah memungkinkan komunikasi antar komputer dengan menggunakan *Transmission Control Protocol* atau *Internet Protocol* (TCP/IP) yang didukung media komunikasi, seperti satelit dan paket radio. Jadi, jarak jangkauannya tidak terbatas. TCP (*Transmission Control Protocol*) bertugas untuk memastikan bahwa semua hubungan bekerja dengan benar, sedangkan IP (*Internet Protocol*) yang mentransmisikan data dari satu komputer ke komputer yang lain. TCP/IP secara umum berfungsi memilih rute terbaik transmisi data, memilih rute alternatif jika suatu rute tidak dapat digunakan, mengatur dan mengirimkan paket-paket pengiriman data.

Menggunakan fasilitas internet, harus berlangganan ke salah satu ISP (*Internet Service Provider*) yang ada dan melayani daerah yang menggunakan layanan ini. ISP ini biasanya disebut penyelenggara jasa internet memberikan banyak sekali manfaat, ada yang bisa memberikan manfaat baik dan buruk. Baik bila digunakan untuk pembelajaran informasi dan buruk bila digunakan untuk sesuatu yang berbau pornografi, informasi kekerasan, dan hal negatif lainnya.

2. Aplikasi Berbasis Web

a. Web Browser

Menurut Limantara (2009:1) mengemukakan bahwa: "*Web browser* adalah aplikasi perangkat lunak yang memungkinkan penggunanya untuk berinteraksi dengan teks, *image*, video, *games*, dan informasi lainnya yang berlokasi pada halaman web pada *World Wide Web* (WWW) atau *Local Area Network* (LAN)". Sejarah *web browser* dimulai pada akhir tahun 80-an, ketika berbagai teknologi baru menjadi dasar pembutan *web browser* pertama di dunia, *World Wide Web*, oleh Tim Berners-Lee pada tahun 1991. *Browser* itu menggabungkan beberapa teknologi *software* dan *hardware* yang sudah eksis maupun masih baru pada waktu itu.

Diperkenalkannya *web browser* NSCA Mosaic pada tahun 1993,salah satu *web browser* gratis pertama memulai ledakan penggunaan *web server*. Marc Andreessen, pimpinan tim Mosaic di NSCA kemudian mendirikan perusahaannya sendiri, Netscape dan meluncurkan *Netscape Navigator* pada tahun 1994. Dengan cepat *Netscape Navigator* menjadi *browser* paling popular di dunia. Pada masa jayanya digunakan oleh 90% pengguna *web browser*. Hingga sekarang sudah banyak *web browser* lain yang bermunculan, diantaranya: *Internet Explorer, Mozila Firefox, Safari, Opera, Google Chrome,* dan lain-lain.

b. Web Server

Menurut Sidik (2009:6) mengemukakan bahwa: "*Web server* adalah komputer yang digunakan untuk menyimpan dokumen-dokumen *web* dimana komputer ini akan melayani permintaan dokumen *web* dari kliennya". Untuk itu kita membutuhkan beberapa *server* yang mempunyai dokumen-dokumen media ke *Browser*. *Browser web* seperti, *Netscape*, *Internet Explorer*, *Mosaic*, *Firefox*. Berkomunikasi melalui jaringan ke *Server Web* dengan menggunakan HTTP (*Hypertext Transfer Protokol*). *Browser* mengirim suatu perintah kepada *server* yaitu meminta dokumen jika ada pada *Protocol* HTTP. *Browser* akan menerima dan mengerti isi dokumen tersebut. *Server Web* juga dapat menjalankan suatu program berdasarkan informasi yang diisi pada *form* isian, seperti menjalankan aplikasi pengakses *database* dan mengirim *e-mail*.

2.1.2. Bahasa Pemrograman

1. HTML *(Hypertext Markup Language)*

Menurut Sibero (2011:19) mengemukakan bahwa: "*Hypertext Markup Language* (HTML) adalah bahasa yang digunakan pada dokumen *web* untuk pertukaran dokumen *web*". HTML dalam ilmu komputer merupakan bahasa pemformatan teks untuk dokumen-dokumen pada jaringan komputer yang

dikenal sebagai *World Wide Web* (WWW) yang sering disebut *web*. Dokumen-dokumen HTML merupakan berkas *text* yang mengandung dua buah bagian isi, yaitu segala sesuatu yang ingin ditampilkan dan diperlihatkan dalam dokumen *Web* dan *Tag* yang merupakan informasi pemformatan, yang tersembunyi dari pandangan pengguna, yang memberitahu *browser* tentang bagaimana caranya menampilkan isi dokumen ke dalam hadapan pengguna.

Beberapa *tag* dalam dokumen-dokumen HTML menentukan bagaimana teks-teks diformat. *Tag-tag* yang lain memberitahu komputer tentang bagaimana menanggapi aksi-aksi yang datang dari pengguna. Sebagai contoh, apa yang harus dilakukan komputer saat pengguna mengklikkan *mouse*-nya pada ikon tertentu. Kemudian *tag* lain yang penting adalah *link* yang mengandung *Uniform Resource Locator* (URL), yang merujuk pada dokumen lain di *server* yang sama atau komputer lain yang ada dijaringan *internet*.

Tag adalah tanda awal < dan tanda akhir > yang digunakan sebagai pengapit suatu elemen. *Tag* pada elemen pembuka diawali dengan tanda < dan diakhiri dengan tanda >. Sedangkan untuk elemen penutup diawali dengan tanda < dan / kemudian diakhiri dengan tanda >. Untuk penulisan *tag* elemen tunggal cukup menuliskan tanda < dan sebelum tanda > ditambahkan tanda /.

Elemen adalah nama penanda yang diapit oleh *tag* yang memiliki fungsi dan tujuan tertentu pada dokumen HTML. Elemen *head* dapat digunakan sebagai tempat penulisan judul dokumen, informasi mengenai dokumen dan defenisi refrensi alamat. Berikut contoh penulisan *tag* yaitu:

a. *Tag Elemen* Pembuka

Bentuk penulisannya adalah : <head>

b. *Tag Elemen* Penutup

Bentuk penulisannya adalah : </head>

c. *Tag Elemen* Tunggal

Bentuk penulisannya adalah : <input type=”text” />

Aturan-aturan penulisan pada dokumen HTML antara lain :

a. Setiap nama *tag* atau elemen pembuka diawali dengan tanda < dan diakhiri dengan tanda >.

b. Setiap nama *tag* atau elemen penutup diawali dengan tanda < dan tanda / kemudian diakhiri dengan tanda >.

c. Untuk *tag* atau elemen yang berdiri sendiri, cukup dengan menuliskan tanda < dan diakhiri *tag* atau elemen ditambahkan tanda / sebelum tanda >.

d. Penulisan nama *tag*, elemen atau atribut dapat menggunakan huruf besar maupun huruf kecil (tidak *case sensitive*).

e. Penulisan nilai pada *atribut* diawali dengan tanda “ dan di akhiri dengan tanda ”.

f. Urutan struktur dokumen setelah *tag* <html> sebaiknya dimulai <head> kemudian <body>, jika *tag* <body> mendahului *tag* <head> secara aturan tidak mengubah atau menyalahi struktur dokumen HTML.

2. PHP *(Perl Hypertext Preprocessor)*

Menurut Peranginangin (2006:2) mengemukakan bahwa: “PHP *(Perl Hypertext Preprocessor)* merupakan bahasa pemrograman berbasis *web* yang memiliki kemampuan untuk memproses dan mengelola data secara dinamis”. PHP dapat dikatakan sebagai sebuah *server-side embedded script language*, yang artinya semua sintaks dan perintah program yang ditulis akan sepenuhya dijalankan oleh *server*, tetapi dapat disertakan pada halaman HTML. PHP adalah produk *Open Source* yang dapat digunakan secara gratis tanpa harus membayar untuk menggunakannya. Bahasa PHP dapat disisipkan *(embedded)* dalam bahasa HTML dan karena bahasa *server-side*, maka PHP akan dieksekusi di *server* sehingga yang dikirimkan ke *browser* adalah hasil jadi bentuk HTML, dan kode PHP tidak akan terlihat.

Salah satu keunggulan yang dimiliki oleh PHP adalah kemampuannya untuk melakukan koneksi ke berbagai macam *software* sistem manajemen bisnis data atau *Database Management System* (DBMS), sehingga dapat

menciptakan halaman *web* yang dinamis. PHP memiliki koneksitas yang baik dengan beberapa DBMS antara lain *Oracle, Sybase, MySQL, Microsoft SQL Server, Solid, Postrgre SQL, Adabas, Filepro, Velocis, dBase, Unix dbm*, dan tidak terkecuali semua *database* ber-*interface* ODBC.

3. JQuery

Menurut Hakim (2014:3), "*jQuery* adalah *JavaScript Library*: kumpulan kode/fungsi *JavaScript* siap pakai, sehingga mempermudah dan mempercepat kita dalam membuat kode *JavaScript* ". Secara standar, apabila kita membuat kode *JavaScript*, maka diperlukan kode yang cukup panjang, bahkan terkadang sangat sulit dipahami. Disinilah peran *jQuery* sebagai *JavaScript Library*: menyederhanakan kode *JavaScript*. Hal ini sesuai dengan slogannya *"Write less, do more"*. Kemampuan yang dimiliki *jQuery* antara lain:

Mempermudah akses dan memanipulasi elemen tertentu pada dokumen. Biasanya diperlukan baris program yang cukup panjang untuk mengakses suatu elemen dokumen. Namun, *jQuery* dapat melakukannya hanya dalam beberapa baris program saja, karena *jQuery* mempunyai selector yang sangat efisien untuk mengakses suatu elemen tertentu pada dokumen yang selanjutnya bisa dimanipulasi sesuai dengan keinginan kita.

a. Mempermudah modifikasi atau perubahan tampilan halaman *web*. Biasanya untuk memodifikasi tampilan halaman *web* digunakan CSS.

Permasalahannya, CSS sangat dipengaruhi oleh *web browser* yang digunakan. Disini *jQuery* dapat menyesuaikan *style* CSS pada semua *browser*.

b. Mempersingkat Ajax *(Asynchronous JavaScript and XML)*. Kemampuan favorit dari Ajax adalah mampu mengambil informasi dari *server* tanpa melakukan *refresh* pada halaman *web*, artinya halaman *web* terlihat berganti secara otomatis. Apabila kita menuliskan kode Ajax secara manual, biasanya diperlukan baris yang cukup panjang. Namun *jQuery* dapat mempersingkatnya menggunakan Ajax *call*, perbandingannya 25 baris kode Ajax dapat disingkat menjadi 3 baris kode saja jika menggunakan *jQuery*.

c. Memiliki API *(Application Programming Interface)*. Dengan API, *jQuery* dapat memanipulasi konten pada suatu halaman *web*, seperti pengubahan teks, manipulasi gambar *(rezise, rotate, crop)*, penyusunan dan pengurutan daftar atau *list, paging*, dan lain-lain.

d. Mampu merespon interaksi antara *user* dengan halaman *web* dengan lebih cepat.

e. Menyediakan fasilitas untuk membuat animasi sekelas *Flash* dengan mudah.

4. JavaScript

Menurut Hakim (2014:4) mengemukakan bahwa: ”*JavaScript* merupakan pemrograman *web* yang berjalan di sisi klien (*browser*), sehingga *JavaScript* dapat membuat *website* lebih hidup (interaktif dan responsif). *JavaScript* bekerja pada sisi *browser*, artinya untuk menampilkan halaman *web*, *user* menuliskan alamat *web* di *address bar url*. Setelah itu”, *browser* mengambil *file* HTML (dengan *file JavaScript* yang melekat padanya jika memang ada) ke *server* yang beralamat di URL yang diketikkan oleh *user*. Selesai *file* diambil, *file* ditampilkan pada *browser*. Setelah *file JavaScript* berada pada *browser*, barulah skrip *JavaScript* tersebut bekerja.

Secara fungsional, *JavaScript* digunakan untuk menyediakan akses skrip pada objek yang dibenamkan (*embeded*). Contoh sederhana dari penggunaan *JavaScript* adalah membuka halaman *pop up*, fungsi validasi pada *form* sebelum data dikirimkan ke *server*, merubah gambar kursor ketika melewati objek tertentu, dan lain-lain. Yang harus diperhatikan dalam pengelolaan pemrograman *JavaScript* diantaranya *JavaScript* adalah bahasa pemrograman yang *case sensitive*, yang artinya *JavaScript* membedakan huruf kecil dan huruf besar. Hal ini sama seperti bahasa pemrograman Turbo C atau C++ dimana huruf “A” tidak sama dengan huruf “a”.

5. CSS (*Cascading Style Sheet*)

Menurut Castro (2007:119) mengemukakan bahwa: "CSS (*Cascading Style Sheet*) adalah instruksi-instruksi yang terkumpul pada suatu dokumen, yang menyediakan fungsi tata letak, efek, serta pengaturan lain pada halaman *web*". CSS saat ini tidak hanya melakukan *formating* halaman *web*, namun digunakan untuk membuat tampilan tata letak terlihat profesional Terdapat pula CSS *Properties* yang berguna untuk mengontrol, seperti dasar penulisan, yaitu ukuran huruf dan warna. CSS juga memiliki beberapa *properties* yang dinamis, yang membuat suatu *item* bisa tampil atau tidak, dan digunakan untuk membuat *drop-down* dan komponen interaktif lainnya. Hal lain, CSS bisa dibuat diluar halaman *web* dan dijalankan pada semua halaman *web* dalam waktu yang bersamaan, karena CSS dibuat agar fleksibel, *powerful* dan efisien.

2.1.3. Basis Data

Menurut Kusrini (2007:2), "Basis data adalah kumpulan data yang saling berelasi". Data sendiri merupakan fakta mengenai objek, orang, dan lain-lain. Data dinyatakan dengan nilai (angka, deretan karakter, atau simbol).

Sementara itu Sutisna (2008:6) mengemukakan bahwa "Agar *website* lebih dinamis, diperlukan *database* untuk pengolahan data. *Database* dapat memudahkan *user* dan *webmaster* untuk memasukkan, menghapus, mengedit, menampilkan, dan mencari data". Beberapa program *database* yang dapat digunakan untuk pembuatan *website* antara lain: Oracle, SQL Server, dan MySQL.

Dalam pembuatan *database* dibutuhkan sebuah aplikasi khusus yang menangani penyimpanan data. Pada pembuatan tugas akhir ini penulis menggunakan aplikasi *database* MySQL.

Menurut Arief (2011:151) mengemukakan bahwa: "MySQL (*My Structure Query Language*) merupakan *software* yang tergolong dalam *database server* dan bersifat *open source*". *Open source* menyatakan bahwa *software* ini dilengkapi dengan *structure code*. MySQL termasuk jenis *Relational Database Management System* (RDBMS) yaitu hubungan antar tabel yang berisi data-data pada suatu *database*. *Database* pada MySQL terdiri dari tabel-tabel, maka setiap tabel mempunyai kolom, baris, serta *record* untuk menyimpan data. Tabel-tabel tersebut di-*link* oleh suatu relasi yang memungkinkan untuk mengkombinasikan data dari beberapa tabel ketika seorang *user* menginginkan menampilkan informasi dari suatu *database*. Penggunaan MySQL biasanya dipadukan dengan menggunakan program aplikasi

PHP pada *web*, tetapi aplikasi *non-web* pun dapat mengakses data dalam *database* di MySQL.

MySQL sebagai *database* mempunyai banyak kelebihan. MySQL terkenal dengan pengolahan data yang cepat walaupun data *record* yang dimasukkan dalam jumlah yang banyak. Kelebihan lain dari MySQL adalah MySQL menggunakan bahasa *query* (permintaan) standar SQL (*Structured Query Language*). SQL adalah suatu bahasa permintaan yang terstruktur yang telah distandarkan oleh semua program pengakses *database* seperti *Oracle*, *Progres* SQL, SQL *server*. Sebuah ekspresi SQL terdiri atas 3 klausa, yaitu *select, from,* dan *where.*

2.1.4. Model Pengembangan Perangkat Lunak

Model yang digunakan pada pengembangan perangkat lunak ini menggunkan model *waterfall* (air terjun). Menurut Boehm dalam Simarmata (2010:54) mengemukakan bahwa "Kemunculan model air terjun adalah untuk membantu mengatasi kerumitan yang terjadi akibat proyek-proyek pengembangan perangkat lunak". Model pengembangan *waterfall* terbagi menjadi tiga tahapan, yaitu:

1. Analisis Kebutuhan

   Tahapan ini sangat menekan masalah pengumpulan kebutuhan pengguna pada tingkatan sistem dengan menentukan konsep sistem beserta antarmuka

yang menghubungkannya dengan lingkungan sekitar. Hasilnya berupa spesifikasi sistem.

2. Perancangan Sistem dan Perangkat Lunak

   Proses perancangan sistem ini difokuskan pada empat atribut, yaitu: struktur data, representasi antarmuka, arsitektur perangkat lunak, dan interaksi antar objek di dalam kelas.

3. Implementasi dan Pengujian Unit

   Pada tahap ini, perancangan perangkat lunak direalisasikan sebagai serangkaian program atau unit program. Kemudian pengujian unit melibatkan verifikasi bahwa setiap unit program telah memenuhi spesifikasinya.

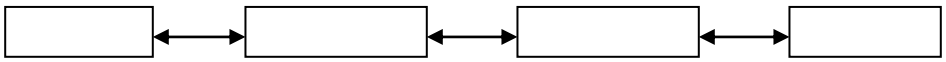
2.2. Teori Pendukung

2.2.1. Struktur Navigasi

Menurut Prihatna (2005:51) mengemukakan bahwa: "Struktur navigasi adalah susunan menu atau hirarki dari suatu situs yang menggambarkan isi setiap halaman dan *link* atau navigasi tiap halaman pada suatu situs *web*". Struktur navigasi dapat digolongkan menurut kebutuhan objek, kemudahan pemakaian dan kemudahan membuatnya yang berpengaruh terhadap waktu pembuatan situs *web.* Bentuk dasar dari struktur navigasi adalah sebagai berikut:

1. *Linear* (Satu Alur)

Linear (satu alur) merupakan struktur yang hanya mempunyai satu rangkaian cerita yang berurut. Dengan kata lain struktur ini hanya dapat menampilkan satu demi satu tampilan layar secara berurut menurut urutannya. Salah satu yang terpenting dari struktur ini adalah tidak diperkenankan terjadinya percabangan.

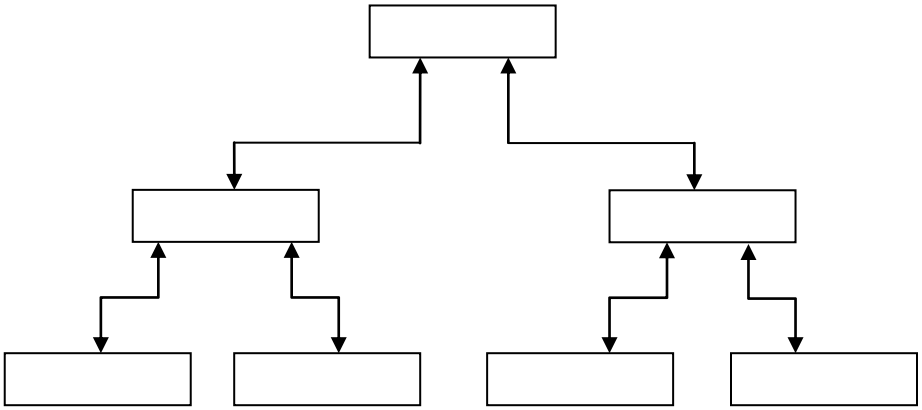


Sumber: Prihatna (2005:51)

Gambar II.1. Struktur Navigasi *Linear*

2. *Hierarchical* (Hirarki)

Struktur *Hierarchi* (bercabang) ini percabangan untuk menampilkan data berdasarkan kriteria tertentu. Tampilan pada menu pertama akan disebut sebagai *Master Page* (halaman utaman kesatu), halaman utama ini akan mempunyai halaman percabangan yang disebut *Slave Page* (halaman pendukung). Jika salah satu halaman pendukung dipilih atau diaktifkan , maka tampilan tersebut akan bernama *Master Page* (halaman utama kedua), dan seterusnya. Yang terpenting dari struktur penjejakan ini tidak diperkenankan adanya tampilan secara *linear*.

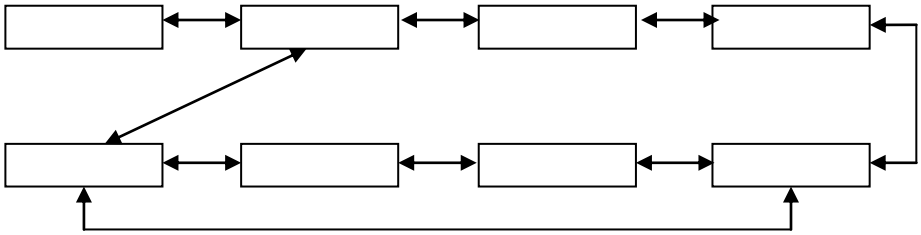


Sumber: Prihatna (2005:51)

Gambar II.2. Struktur Navigasi *Hierarchical* (Hirarki)

3. Non *Linear* (Tidak berurut)

Struktur penjejakan *Non Linear* (tidak berurut) merupakan pengembangan dari struktur penjejakan *Linear*. Pada struktur ini diperkenankan membuat penjajakan bercabang. Pemakai bebas menelusuri *website* tanpa dibatasi oleh suatu rute dimana kontrol navigasi dapat mengakses ke semua halaman manapun. Percabangan yang dibuat pada struktur *Non Linear* ini berbeda dengan percabangan yang dibuat pada struktur *Hierarchi*, karena pada percabangannya *Non Linear* ini walaupun terdapat percabangan, tetapi tiap-tiap tampilan mempunyai kedudukan yang sama tidak ada *Master Page* dan *Slave Page.*

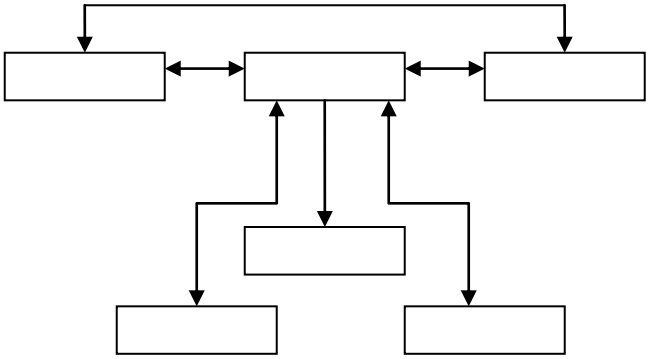


Sumber: Prihatna (2005:51)

Gambar II.3. Struktur Navigasi *Non Linear*

4. *Composite* (Campuran)

Composite (campuran) atau disebut juga struktur penjejakan bebas merupakan gabungan dari ketiga struktur sebelumnya yaitu *Linear*, *Non Linear*, dan *Hierarchi*. Jika suatu tampilan membutuhkan percabangan, maka dapat dibuat percabangan, dan bila dalam percabangan tersebut terdapat suatu tampilan yang sama kedudukannya makan dapat dibuat struktur *Linear* dalam percabangan tersebut. Penggunaan peta penjejakan bergantung kepada kebutuhan dan tujuan dari *web* yang hendak dibuat. Semakin kompleks peta penjejakan yang digunakan, maka semakin sulit pembuatan *page* dari peta penjejakan tersebut.



Sumber: Prihatna (2005:51)

Gambar II.4. Struktur Navigasi Campuran

2.2.2. *Entity Relationship Diagram*

Menurut Pratama (2014:49), "ERD (*Entity Relationship Diagram*) adalah diagram yang menggambarkan keterkaikatan antar tabel beserta dengan *field-field* di dalamnya pada suatu *database system*".

Menurut Kusnini (2007:21) mengemukakan bahwa: "Perancangan basis data dengan menggunakan model *entity relationship* adalah dengan menggunakan *Entity Relationship Diagram* (ERD)". Model ini dirancang untuk menggambarkan persepsi dari pemakai dan berisi objek-objek dasar yang disebut *entity* dan hubungan antar *entity-entity* tersebut yang disebut *relationship*.

Komponen-komponen yang digunakan dalam ERD terdiri dari:

1. *Entity*

   Entity adalah objek yang dapat dibedakan dengan yang lain dalam dunia nyata. *Entity* dapat berupa objek secara fisik seperti orang, rumah, atau

kendaraan. *Entity* dapat pula berupa objek secara konsep seperti pekerjaan, perusahaan, dan sebagainya. *Entity* digambarkan dalam bentuk persegi panjang.

2. Atribut

   Atribut adalah karakteristik dari *entity* atau *relationship*, yang menyediakan penjelasan detail tentang *entity* atau *relationship* tersebut. Nilai Atribut merupakan suatu data aktual atau informasi yang disimpan pada suatu atribut di dalam suatu *entity* atau *relationship*. Atribut digambarkan dalam bentuk oval.

3. *Relationship*

   Relationship adalah hubungan yang terjadi antara satu atau lebih *entity*. Kumpulan *relationship* yang sejenis disebut *Relationship set. Relationship* digambarkan dalam bentuk garis lurus.

   a. Derajat *Relationship*

      Derajat dari *relationship* menjelaskan jumlah *entity* yang berpartisipasi dalam suatu *relationship*. Terdapat tiga jenis derajat dari *relationship* yaitu:

      1) *Unary degree* (derajat satu), yaitu satu buah *relationship* menghubungkan satu buah *entity*.
      2) *Binary degree* (derajat dua), yaitu satu buah *relationship* yang menghubungkan dua buah *entity*.

3) *Ternary degree* (derajat tiga), yaitu satu buah *relationship* menghubungkan tiga buah *entity*.

b. Kardinalitas Relasi

Kardinalitas Relasi menunjukkan batasan jumlah keterhubungansatu *entity* dengan *entity* lainnya. Terdapat 3 macam kardinalitas relasi yaitu:

1) Relasi Satu ke Satu *(One to One)*

Hubungan relasi satu ke satu yaitu setiap entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B.

2) Relasi Satu ke Banyak *(One to Many)*

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi setiap entitas pada entitas B dapat berhubungan dengan satu entitas pada himpunan entitas A.

3) Relasi Banyak ke Banyak *(Many to Many)*

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B.

2.2.3. *Logical Record Structure* (LRS)

Menurut Priyadi (2014:40), "LRS (*Logical Record Structure*) adalah representasi dari struktur *record-record* pada tabel-tabel yang terbentuk dari

hasil antar himpunan entitas". Menentukan kardinalitas, jumlah tabel dan *Foreign Key* (FK).

Logical Record Structure dibentuk dengan nomor dari tipe *record*. Beberapa tipe *record* digambarkan oleh kotak empat persegi panjang dan dengan nama yang unik. Beda LRS dengan diagram E-R adalah nama tipe *record* berada diluar kotak *field* tipe *record* ditempatkan. *Logical Record Structure* terdiri dari *link-link* diantara tipe *record*. Link ini menunjukkan arah dari satu tipe *record* lainnya. Banyak *link* dari LRS yang diberi tanda *field-field* yang kelihatan pada kedua *link* tipe *record*. Penggambaran LRS mulai dengan menggunakan model yang dimengerti. Dua metode yang dapat digunakan, dimulai dengan hubungan kedua model yang dapat dikonversikan ke LRS. Metode yang lain dimulai dengan ER-diagram dan langsung dikonversikan ke LRS.

2.2.4. Pengujian Web

Menurut Pressman (2010:482), "tujuan dari pengujian adalah untuk menemukan dan memperbaiki sebanyak mungkin kesalahan dalam program sebelum menyerahkan program kepada *customer"*. Salah satu pengujian yang baik adalah pengujian yang memiliki probabilitas tinggi dalam menemukan kesalahan.

Menurut Pressman (2010:495), "*Black-Box testing* berfokus pada persyaratan fungsional perangkat lunak yang memungkinkan *engineers* untuk memperoleh set kondisi *input* yang sepenuhnya akan melaksanakan persyaratan fungsional untuk sebuah program". *Black-Box testing* berusaha untuk menemukan kesalahan dalam kategori berikut:

1. Fungsi yang tidak benar atau fungsi yang hilang
2. Kesalahan antarmuka
3. Kesalahan dalam struktur data atau akses *database* eksternal
4. Kesalahan perilaku (*behavior*) atau kesalahan kinerja
5. Inisialisasi dan pemutusan kesalahan

Tes ini dirancang untuk menjawab beberapa pertanyaan-pertanyaan berikut ini:

1. Bagaimana validitas fungsional diuji?
2. Bagaimana perilaku dan kinerja sistem diuji?
3. Apa kelas *input* akan membuat kasus uji yang baik?
4. Apakah sistem *sensitive* terhadap nilai *input* tertentu?
5. Bagaimana batas-batas kelas data yang terisolasi?
6. Kecepatan dan *volume* data seperti apa yang dapat ditolerir sistem?
7. Efek apakah yang akan menspesifikasikan kombinasi data dalam sistem operasi?

Menurut Pratama (2014:51) mengemukakan bahwa: “Terdapat setidaknya empat buah jenis pengujian di sisi pengembang (*blackbox*) ini ” . Keempat jenis pengujian tersebut meliputi :

1. Pengujian *Interface* (tatap muka) aplikasi.

   Pengujian *Interface* (tatap muka) aplikasi sistem informasi bertujuan untuk mengetahui fungsionalitas dari setiap elemen *interface* yang ada di setiap halaman pada aplikasi sistem informasi. Elemen ini berupa tombol (*button*) yang menjalankan aksi sesuai yang diharapkan oleh pengguna dan pengembang.

2. Pengujian fungsi dasar sistem.

   Pengujian fungsi dasar sistem bertujuan untuk mengetahui sejauh mana kinerja dari setiap fungsi dasar sistem yang ada di dalam aplikasi sistem informasi. Fungsi-fungsi ini dalam penerapannya membentuk satu atau sejumlah modul. Modul ini dapat digunakan baik di sisi pengembang maupun sebagai pengguna (misal: instalasi modul melalui akun *administrator*).

3. Pengujian *form handle* sistem.

   Pegujian *form handle* sistem bertujuan untuk mengetahui seperti apa dan sejauh mana respon oleh sistem informasi terhadap *input*-an yang diberikan oleh pengguna. Inputan yang diberikan oleh pengguna ke dalam

sistem informasi dapat berupa *input*-an bernilai (misalkan: data) maupun *input*-an kosong.

4. Pengujian keamanan sistem.

   Pengujian keamanan sistem bertujuan untuk mengetahui sejauh mana tingkat keamanan yang dimiliki oleh sistem informasi untuk dapat memberikan kenyamanan kepada para pengguna. Keamanan dicek dari sisi sistem (misalkan: SQL *Injection*), kebijakan (misalkan: ada tidaknya penanganan minimal jumlah karakter untuk *password*, otentikasi via *email*), serta *user* atau pengguna (misalkan: ada tidaknya perbedaan hak akses untuk setiap kelompok pengguna).