BAB II

LANDASAN TEORI

2.1. Konsep Dasar *Web*

Perkembangan dunia *internet* yang sangat pesat membuat banyak orang menghabiskan sebagian besar waktunya di depan perangkat yang terkoneksi dengan *internet*, mulai dari belajar sampai berbelanja semua dilakukan lewat dunia maya. Dalam membangun suatu sarana informasi terdapat sekumpulan perangkat lunak yang nantinya akan digabungkan menjadi sebuah aplikasi yang dapat mengolah data dan menghasilkan informasi yang bermanfaat.

2.1.1. Website

Menurut Yuhefizar (2009:2) Situs web (*website*) adalah keseluruhan halaman-halaman web yang terdapat dalam sebuah domain yang mengandung informasi, sebuah website biasanya dibangun atas banyak halaman web yang saling berhubungan.

Hubungan antara satu halaman *web* dengan yang lain disebut *Hyperlink*, sedangkan *text* yang dijadikan media penghubung disebut *Hypertext*.

2.1.2. Internet

Menurut Chaffey (2009:186) *internet* adalah jaringan fisik yang menghubungkan komputer di seluruh dunia. *Internet* terdiri dari infrastruktur jaringan *server* dan hubungan antara komputer yang digunakan untuk menyimpan dan pemindahan informasi antara PC klien dan *server web*.

2.1.3. Browser

Menurut Chaffey (2009:96) w*eb browser* adalah *software* seperti *Microsoft Internet Exploler* dan *Mozilla Firefox* yang bisa kita gunakan untuk mengakses informasi pada *www* yang disimpan di *web service*.

Sedangkan menurut Shelly dan Velmaart (2011:81) *web browser* atau *browser* adalah perangkat lunak aplikasi yang memungkinkan pengguna untuk mengakses dan melihat halaman *web* atau mengakses program *web* 2.0.

Berdasarkan pendapat para ahli di atas dapat di simpulkan bahwa *browser* adalah *software* untuk dapat mengakses informasi pada www atau *World Wide Web*.

2.1.4. Web Server

Menurut Kurniawan (2008:2) *Web Server* adalah sebuah perangkat lunak *server* yang berfungsi menerima permintaan HTTP atau HTTPS dari klien yang dikenal dengan *web browser* dan mengirimkan kembali hasilnya dalam halaman-halaman *web* yang umumnya berbentuk dokumen HTML. *Web server* yang dimaksud disini adalah simulasi dari sebuah *web server* secara fisik. *Web server* biasanya juga disebut HTTP server karena menggunakan protocol HTTP sebagai basisnya. Beberapa *web server* yang sering digunakan diantaranya adalah PWS, IIS, Apache dan sebagainya.

2.1.5. HTML

Menurut Shelly dan Velmaart (2011:678) HTML (*HypertextMarkup Language*) adalah bahasa format khusus yang digunakan *programmer* untuk memformat dokumen untuk ditampilkan di *web*.

Sedangkan menurut Chaffey (2009: 96) HTML atau *Hyper Text Markup language* adalah halaman standar *web* presentasi dengan menggunakan format untuk menentukan pesan dan tata letak halaman *web*.

Berdasarkan pendapat para ahli di atas, penulis menyimpulkan bahwa HTML atau *Hyper Text Markup Language* adalah bahasa pemrograman yang digunakan untuk memformat dokumen untuk ditampilkan di *web*.

```
1  <html>
2  <head>
3  <title>Belajar HTML</title>
4  </head>
5  <body bgcolor="#FF0000">
6  <h1>Selamat Datang di Dunia HTML</h1>
7  <hr>
8  Belajar HTML ternyata sangat mudah sekali.
9  Ini adalah dokumen <b>HTML</b> saya yang <i>pertama</i>.
10
11 <p>Saya senang sekali Belajar Komputer dan Internet
12 <a href="http://naughtyric.blogspot.com">disini</a>
13 Terimakasih</p>
14 </body>
15 </html>
```

Gambar II.1 Contoh HTML

2.1.6. PHP

Menurut Shelly dan Vermaat (2011:682) PHP, yang merupakan singkatan dari *Personal Home Page*, merupakan sebuah sumber bebas terbuka dari bahasa skrip. PHP, yang merupakan bahasa mirip dengan C, Java dan Perl, digunakan

terutama pada *web server* linux. Pengembang *web* membuat halaman *web* dinamis dengan memasukkan skrip PHP bersama dengan HTML atau XHTML dalam suatu halaman *web*.

2.1.7. *Cascading Style Sheet (CSS)*

Menurut Murya (2012:108) "*Cascading Style Sheet* (CSS) merupakan salah satu bahasa pemrograman *web* untuk mengendalikan beberapa komponen dalam sebuah *web* sehingga akan lebih terstruktur dan seragam". Sama halnya *styles* dalam aplikasi pengolahan kata seperti *Microsoft Word* yang dapat mengatur beberapa *style*, misalnya *heading, subbab, bodytext, footer*, *images*, dan *style* lainnya untuk dapat digunakan bersama-sama dalam beberapa berkas (*file*). Pada umumnya CSS dipakai untuk memformat tampilan halaman *web* yang dibuat dengan bahasa HTML dan XHTML.

CSS dapat mengendalikan ukuran gambar, warna bagian tubuh pada teks, warna tabel, ukuran *border*, warna *border*, warna *hyperlink*, warna *mouse over*, spasi antar paragraf, spasi antar *teks*, margin kiri, kanan, atas, bawah, dan parameter lainnya. CSS dalah bahasa *style sheet* yang digunakan untuk mengatur tampilan dokumen. Dengan adanya CSS memungkinkan kita untuk menampilkan halaman yang sama dengan format yang berbeda.

2.1.8. *JavaScript*

Menurut Andi (2012:2) "*JavaScript* merupakan skrip yang paling banyak digunakan dalam pemrograman *web* pada sisi *client* dewasa ini". Dengan adanya *JavaScript* sebuah *web* akan menjadi lebih hidup, cepat, dan tampil lebih menawan dengan sebuah animasi. Dikarenakan begitu luasnya penggunaannya

JavaScript, banyak pengembang yang membangun sebuah *library*/pustaka *JavaScript* untuk memudahkan para programmer *website*. *Library*/pustaka inilah yang selanjutnya disebut sebagai *framework*, yaitu kumpulan fungsi-fungsi *JavaScript* yang siap digunakan untuk memanipulasi DOM (*Document Object Model*). Banyak *framework* yang telah dikembangkan, antara lain YUI, *prototype*, *mootools*, dan *JQuery*.

2.1.9. *JQuery*

Menurut Utomo dan Ali Akbar (2011:62) "*JQuery* merupakan *library JavaScript* yang banyak digunakan saat ini". *JQuery* dibuat oleh John Resig pada tahun 2006. Banyak *website* yang dimanfaatkan *library* ini untuk menyederhanakan fungsi-fungsi yang ada pada *JavaScript* atau *Ajax*. Sesuai dengan slogannya *JQuery* sendiri "*Write less, do more*", menulis sedikit namun bisa mengerjakan banyak hal, sehingga dapat menghemat *coding* program, yang sebelumnya menggunakan *JavaScript* butuh beberapa baris kode, namun dengan *JQuery* hanya butuh satu dua baris saja.

Penggunaan JQuery ini sebenarnya cukup mudah karena semuanya sudah terbungkus dalam bentuk *library* dan *plugin*, dan kita hanya perlu mengetahui cara penerapannya saja didalam sebuah *website*.

2.1.10. Basis Data

Menurut Connolly dan Begg (2010:15) *database* adalah kumpulan data yang berelasi secara logikal dan sebuah deskripsi dari data tersebut yang di desain untuk memenuhi kebutuhan organisasi. *Database* adalah sebuah tempat

penyimpanan besardari data yang dapat digunakan secara terus menerus oleh banyak departemen dan *user*.

Menurut O'Brien dan Marakas (2010:173) *database* adalah kumpulan terintegrasi dari elemen data yang secara logika saling berhubungan. *Database* mengonsolidasikan berbagai catatan yang dahulu disimpan dalam file-file terpisah kedalam satu gabungan umum elemen data yang menyediakan data untuk banyak aplikasi. Data yang disimpan dalam *database* independen dari program aplikasi yang menggunakannya dan dari jenis peralatan penyimpanan tempat mereka disimpan. Jadi, *database* berisi berbagai elemen data yang mendeskripsikan berbagai entitas dan hubungan antar entitas.

Dari pendapat para ahli di atas, maka dapat disimpulkan bahwa *database* adalah suatu tempat penyimpanan yang berisi berbagai elemen data yang berelasi secara logikal yang disimpan dalam satu wadah yang dapat digunakan secara terus-menerus oleh *user* maupun banyak departemen.

2.1.11. MySQL

Menurut Arief (2011:151) MySQL (*My Structure Query Languange*) adalah salah satu jenis *database server* yang sangat terkenal dan banyak digunakan untuk membangun aplikasi web yang menggunakan database sebagai sumber dan pengelolaan datanya. MySQL bersifat *open source* dan menggunakan SQL (*Structured Query Languange*). MySQL biasa dijalankan diberbagai *platform* misalnya windows Linux, dan lain sebagainya.

MySQL merupakan DBMS yang *multithread*, multi *user* yang bersifat gratis di bawah lisensi GNU *General Public Licence* (GPL). Tidak seperti Apache yang

merupakan *software* yang dikembangkan oleh komunitas umum, dan hak cipta untuk kode sumber dimiliki oleh penulisnya masing-masing. Seperti yang telah disebutkan sebelumnya, MySQL bersifat gratis atau *open source* sehingga bisa menggunakannya secara gratis.

2.1.12. Waterfall

Menurut Pressman (2010:39) model *waterfall* adalah model klasik yang bersifat sistematis, berurutan dalam membangun *software.*

Adapun 5 langkah metode *waterfall* menurut Pressman (2010:39) adalah sebagai berikut :

1. *Communication*

   Langkah ini merupakan analisis terhadap kebutuhan *software*. Dan tahap untuk mengadakan pengumpulan data dengan melakukan pertemuan dengan *costumer*, maupun mengumpulkan data-data tambahan baik yang ada di jurnal artikel maupun dari *internet.*

2. *Planning*

   Proses *planning* merupakan lanjutan dari proses *communication (analysis requirement)*. Tahapan ini akan menghasilkan dokumen *user requirement* atau bisa dikatakan sebagai data yang berhubungan dengan keinginan *user* dalam pembuatan *software*, termasuk rencana yang akan dilakukan.

3. *Modeling*

   Proses *modeling* ini akan menerjemahkan syarat kebutuhan ke sebuah perancangan *software* yang dapat diperkirakan sebelum dibuat *coding*. Proses ini berfokus pada rancangan struktur data, arsitektur *software*,

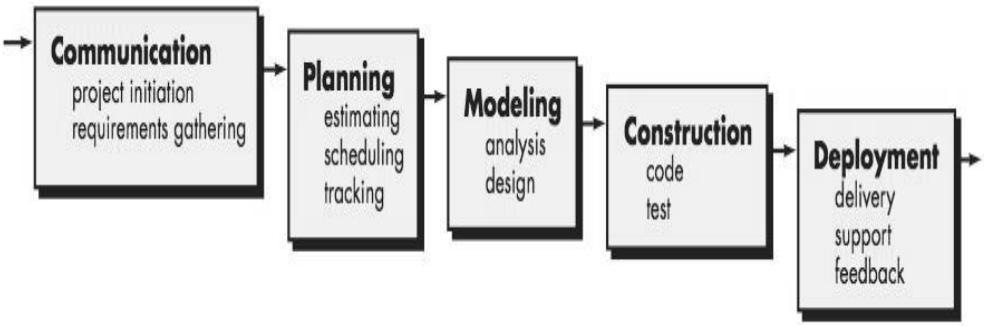
representasi *interface*, dan detail (algoritma) *procedural*. Tahapan ini akan menghasilkan dokumen yang disebut *software requirement*.

4. *Contruction*

   Contruction merupakan proses membuat kode. *Coding* atau pengkodean merupakan penerjamahan desain dalam bahasa yang bias dikenali oleh komputer. *Programmer* akan menerjemahkan transaksi yang diminta oleh *user*. Tahapan inilah yang merupakan tahapan secara nyata dalam mengerjakan suatu *software*, artinya penggunaan komputer akan dimaksimalkan dalam tahapan ini. Setelah pengkodean selesai maka akan dilakukan *testing* terhadap system yang telah dibuat tadi. Tujuan *testing* adalah menemukan kesalahan-kesalahan terhadap system tersebut kemudian bisa diperbaiki.

5. *Deployment*

   Tahapan ini bias dikatakan final dalam pembuatan sebuah *software* atau sistem. Setelah melakukan analisis, desain dan pengkodean maka system yang sudah jadi akan digunakan oleh *user*. Kemudian *software* yang telah dibuat harus dilakukan pemeliharaan secara berkala.



Sumber : Pressman (2010:39)

Gambar II.2. Metode Waterfall

2.2. Teori Pendukung

Untuk lebih memahami isi dari tugas akhir ini, maka dibutuhkan beberapa pengetahuan mengenai definisi serta uraian yang berkaitan dengan teori pendukung, sebagai berikut :

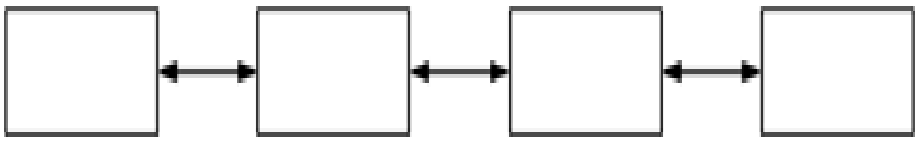
2.2.1. Struktur Navigasi

Menurut Binanto (2010:268) “Struktur navigasi adalah gabungan dari struktur referensi informasi situs *web* dan mekanisme *link* yang mendukung pengunjung untuk melakukan penjelajahan situs.

Menurut Binanto (2010:269) ada empat macam bentuk dasar dari struktur navigasi yang biasa digunakan yaitu:

1. Struktur *Navigasi Linier*

   Struktur *navigasi linier* hanya mempunyai satu rangkaian cerita yang berurut yang menampilkan satu demi satu tampilan layar secara berurut menurut urutannya. Tampilan yang dapat ditampilkan pada struktur jenis ini adalah satu halaman sebelumnya atau satu halaman sesudahnya, tidak dapat dua halaman sebelumnya atau dua halaman sesudahnya, pengguna akan melakukan navigasi secara berurutan, dalam *frame* atau *byte* informasi satu ke yang lainnya.

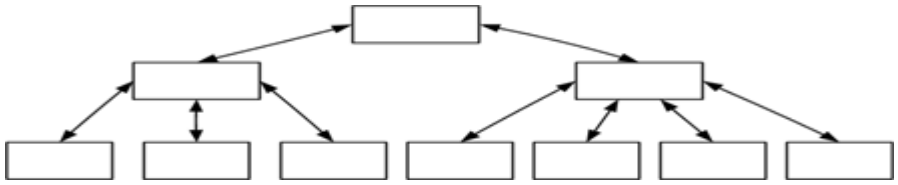


Sumber : Binanto, (2010:269)

Gambar II.3. Struktur Navigasi Linier

2. Struktur *Navigasi Hirarki*

Struktur dasar ini disebut juga struktur *linier* dengan percabangan karena pengguna melakukan navigasi disepanjang cabang pohon struktur yang terbentuk oleh logika isi.

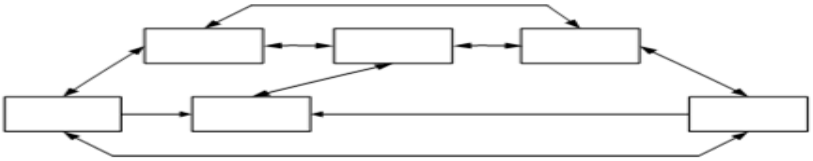


Sumber : Binanto (2010:269)

Gambar II.4. Struktur Navigasi Hirarki

3. Struktur Navigasi Tidak Berurut (*Non-Linier)*

Struktur navigasi *non-linier* merupakan pengembangan dari struktur navigasi linier. Pada struktur ini diperkenankan membuat navigasi bercabang. Percabangan yang dibuat pada struktur non-linier ini berbeda dengan percabangan pada struktur hirarki, karena pada percabangan non-linear ini walaupun terdapat percabangan tetap tiap-tiap tampilan mempunyai kedudukan yang sama yaitu tidak ada *Master Page* dan *Slave Page*, pengguna akan melakukan navigasi dengan bebas melalui isi proyek dengan tidak terikat dengan jalur yang sudah ditentukan sebelumnya.

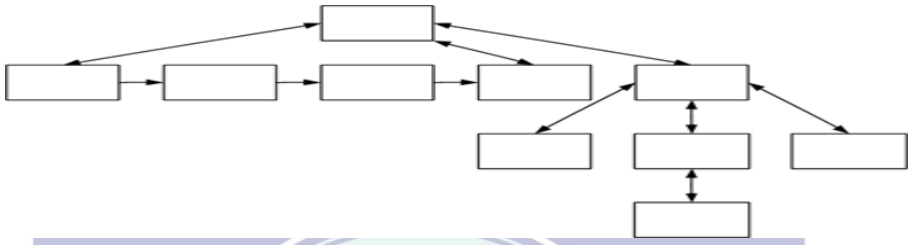


Sumber : Binanto (2010:270)

Gambar II.5. Struktur Navigasi Tidak Berurut (*Non-Linier)*

4. Struktur Navigasi Campuran (*Composite*)

Struktur navigasi pengguna akan melakukan navigasi dengan bebas (secara *non-linier*), tetapi terkadang dibatasi presentasi *linier* film atau informasi penting dan pada data yang paling terorganisasi secara logis pada suatu hirarki.



Sumber : Binanto (2010:270)

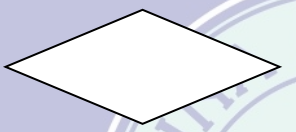
Gambar II.6. Struktur Navigasi Campuran (*Composite*)

2.2.2. ERD (*Enterprise Relationship Diagram*)

Menurut Sutanta (2011:91) "Entity Relationship Diagram (ERD) merupakan suatu model data yang dikembangkan berdasarkan objek." Entity Relationship Diagram (ERD) digunakan untuk menjelaskan hubungan antar data dalam basis data kepada pengguna secara logis. Entity Relationship Diagram (ERD) didasarkan pada suatu persepsi bahwa real world terdiri atas obyek-obyek dasar tersebut. Penggunaan Entity Relationship Diagram (ERD) relatif mudah dipahami, bahkan oleh para pengguna yang awam. Bagi perancang atau analis sistem, Entity Relationship Diagram (ERD) berguna untuk memodelkan sistem yang nantinya, basis data akan di kembangkan. Model ini juga membantu perancang atau analis sistem pada saat melakukan analis dan perancangan basis data karena model ini dapat menunjukkan macam data yang dibutuhkan dan kerelasian antardata didalamnya.

2.2.3. Komponen ERD (*Enterprise Relationship Diagram*)

Tabel II.1.
Simbol *Entity Relationship Diagram*

NO	SIMBOL	KETERANGAN	CONTOH
1.		*Entity* adalah sebuah "benda" (*thing*) atau "objek" (*object*) didunia nyata yang dapat dibedakan dari semua objek lainnya	Kumpulan orang yang berobat di rumah sakit didefinisikan sebagai pasien.
2.		*Relationship* adalah hubungan diantara beberapa *entity*.	Pasien mungkin memiliki atau mungkin tidak memiliki biaya asuransi.
3.		Atribut merupakan sebutan untuk mewakili suatu *entity*.	Dalam Entity Dokter terdapat atribut kode dokter, nama,dokter dan spesialisasi.
4.		Garis penghuung yaitu sebagai penghubung antara himpunan *relationship* dengan himpunan *entity* dan himpunan *entity* dengan *atribut*nya	Satu dokter bisa memiliki banyak pasien, satu pasien bisa jadi hanya memiliki satu dokter utama

Sumber: Kusrini (2007:21)

Menurut Brady dan Loonam J (2010:48) Atribut dapat memiliki sifat-sifat sebagai berikut:

1. Atribut Sederhana (*Simple Attribute*)

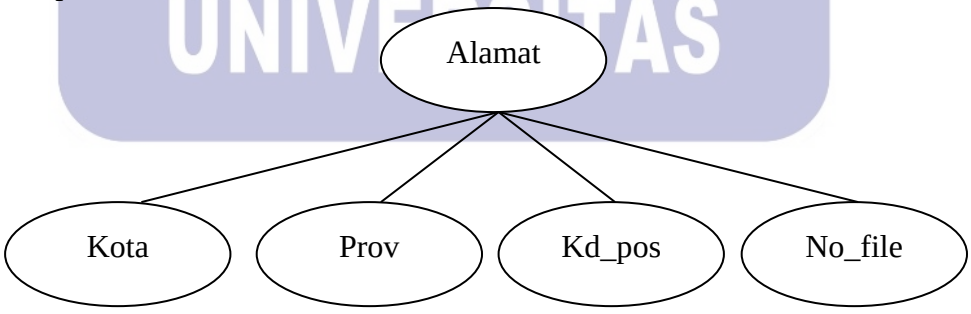
Atribut sederhana adalah atribut yang nilainya tidak dapat dibagi lagi menjadi banyak atribut yang lebih kecil. Contoh atribut sederhana harga seperti berikut:

Sumber: Brady dan Loonam J (2010:49)

Gambar II.7.Atribut Sederhana (*Simple Attribute*)

2. Atribut Komposit (*Composite Attribut*)

Atribut komposit adalah atribut gabungan yang nilainya dapat dipecah menjadi bagian yang lebih kecil. Atas sering disebut atribut yang terdiri dari beberapa atribut kecil didalamnya. Contoh atribut komposit adalah alamat seperti berikut:

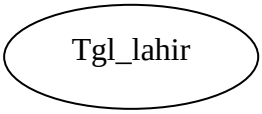


Sumber: Brady dan Loonam J (2010:49)

Gambar II.8.Atribut Komposit (*Composite Attribute*)

3. Attribute Bernilai Tunggal (*Single Values Attribute*)

Atribut bernilai tunggal adalah jenis atribt yang nilainya hanya satu dari suatu entitas. Contoh atribut bernilai tunggal adalah tanggal_lahir dari entitas mahasiswa. Telah bisa dipastikan bahwa setiap mahasiswa mempunyai satu tanggal_lahir seperti gambar berikut:

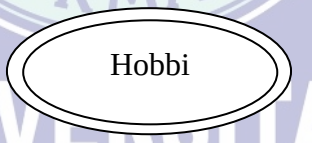


Sumber: Brady dan Loonam J (2010:50)

Gambar II.9.Attribute Bernilai Tunggal (*Single Values Attribute*)

4. Atribut Bernilai Banyak (*Multivalues Attribute*)

Atribut bernilai banyak adalah jenis atribut yang nilainya lebih dari satu dalam suatu entitas tertentu. Contoh atribut bernilai banyak adalah hobbi dimungkinkan bahwa mahasiswa memiliki lebih dari satu hobbi seperti gambar berikut:



Sumber: Yanto Brady dan Loonam J (2010:50)

Gambar II.10.Atribut Bernilai Banyak (*Multivalues Attribute*)

5. Atribut Turunan (*Derioed Attribute*)

Atribut turunan adalah jenis atribut yang nilainnya diperoleh dari atribut yang lain. Contoh atribut turunan adalah masa bati dari entitas pegawai. Atribut tanggal_masuk_kerja sudah ada nilainya.

Pada dasarnya atribut masa bakti tidak akan dijadikan suatu kolom. Atribut masa bakti akan muntul dengan bantuan query.

Sumber: Yanto Brady dan Loonam J (2010:51)

Gambar II.11.Atribut Turunan *(Derioed Attribute)*

6. Atribut Identitas (*Key Attribute*)

Atribut entitas adalah atribut yang dijadikan sebagai kunci pada susatu *table*. Sifat atribut identitas ini unik , tidak ada yang menyamai, atribut identitas terdiri dari beberapa jenis yaitu:

a. *Super Key*

Super key adalah satu atribut atau kumpulan atribut yang secara unik mengidentifikasi sebuah baris didalam relasi atau himpunan dari satu atau lebih entitas yang dapat digunakan untuk mrngidentifikasi secara unik sebuah entitas dalam set entitas.

b. *Candidate key*

Candidate key adalah atribut yang menjadi determinan yang dapat dijadikan identitas baris pada sebuah relasi. Biasanya *super key* minimum.

c. *Primary key*

Primary key adalah kadidat *key* yang dipilih untuk mengidentifikasi baris data secara unik dalam relasi.

d. *Alternative key*

Alternative key adalah kandidat *key* yang tidak terpilih sebagai *primary key* atau atribut untuk menggantikan kunci utama.

e. *Foreign key*

Foreign key adalah atribut dengan domain yang sama menjadi kunci utama sebuah relasi, tetapi pada relasi lain atribut tersebut sebagai atribut biasa.

7. *Composite Key*

Composite key adalah kunci yang terdiri dari dua atribut atau lebih. Atribut-atribut tersebut jika berdiri sendiri tidak menjadi identitas baris, tetapi bila dirangkaian menjadi satu kesatuan akan dapat mengidentifikasi secara unik.

Sumber: Brady dan Loonam J (2010:52)

Gambar II.12.*Composite Key*

2.2.4. Derajat *Relationship*

Menurut Yakub (2008:33) mengemukakan bahwa "model relasi ini berdasarkan persepsi dunia nyata diantaranya himpunan objek dan di identifikasikan secara unik, dan objeknya dapat berbentuk orang, barang, dan sebagainya".

Kardinalitas relasi menunjukan maksimum entitas yang dapat berlerasi dengan entitas pada himpunan entitas yang lain. Kardinalitas relasi yang terjadi di

antara dua himpunan entitas (misalkan A dan B) dapat berupa satu ke satu (*one to one*), satu ke banyak (*one to many*) dan banyak ke banyak (*many to many*).

Umumnya ada 4 macam derajat relasi, yaitu:

1. Satu ke satu (*One to One*)

   1-1 yaitu setiap entitas pada himpunan entitas A berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas B, dan begitu juga sebaliknya setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

Sumber : Yakub (2008:35)

Gambar II.13. Kardinalitas *One to One*

2. Satu ke banyak (*One to Many*)

   1-N berarti bahwa setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi tidak sebaliknya, dimana setiap entitas pada himpunan entitas B berhubungan dengan paling banyak dengan satu entitas pada himpunan entitas A.

Sumber : Yakub (2008:36)

Gambar II.14. Kardinalitas *One to Many*

3. Banyak ke banyak (*Many to Many*)

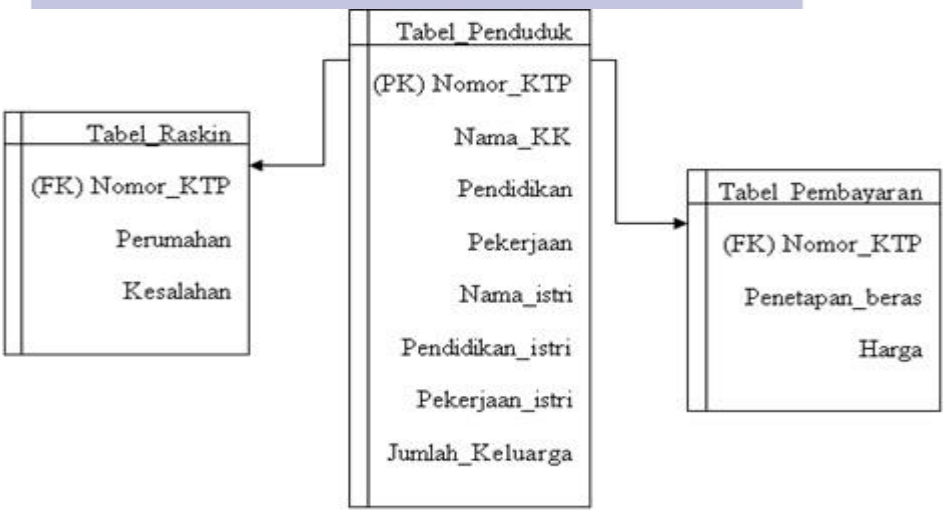
Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B dan demikian juga sebaliknya, dimana setiap entitas pada himpunan entitas B dapat berhubungan dengan banyak entitas pada himpunan entitas A.

Sumber : Yakub (2008:37)

Gambar II.15. Kardinalitas *Many to Many*

2.2.5. LRS (*Logical Record Structure*)

Menurut Hasugian dan Shidiq (2012:608) memberikan batasan bahwa LRS adalah sebuah model sistem yang digambarkan dengan sebuah *diagram*-ER akan mengikuti pola atau aturan permodelan tertentu dalam kaitanya dengan konvensi ke LRS.



Sumber: Suryati dan Bambang Eka Purnama (2010:37)

Gambar II.16.Contoh *Logical Record Structure*

2.2.6. *Black-Box Testing*

Menurut Pressman (2010:495) Pengujian kotak hitam bukan teknik alternatif untuk kotak putih. Sebaliknya, ini merupakan pendekatan pelengkap yang memungkinkan dilakukan untuk mengungkapkan kelas kesalahan yang berbeda dari yang diungkap oleh motede kotak putih.

Pengujian Kotak Hitam (*Blackbox Testing*) khusus di didesain untuk mencari kesalahan dengan melakukan ujicoba pada interface software. Pegujian Kotak Hitam (*Blackbox Testing*) mendemonstrasikan fungsi dari perangkat lunak yang beroperasi, dengan mengecek apakah input sudah bisa diterima dengan baik, dan hasil outputnya sesuai dengan apa yang diharapkan, uji coba Kotak Hitam (*Blackbox Testing*) melakukan pengecekan pada integritas informasi eksternal, pada dasarnya pengujian Kotak Hitam (*Blackbox Testing*) hanya memeriksa hasil output yang dihasilkan apakah sudah sesuai dengan apa yang diharapkan dan dinyatakan benar,namun pengujian Kotak Hitam (*Blackbox Testing*) tidak mengecek logika dari perangkat lunak, Pengujian unit merupakan pengujian secara individual terhadap semua program untuk memastikan bahwa program bebas dari kesalaan. namun jika ditemukan *error* atau kesalahan pada program, *user* akan langsung mencari kesalahannya dan proses untuk melakukan pencarian kesalahan ini dikenal dengan *debugging*.

Beberapa proses yang dilakuan dalam *Black-Box Testing* diantaranya yaitu :

1. Fungsi-fungsi yang tidak benar, baik input atau pun output, dalam hal ini hanya melihat apakah proses input dan output sudah sesuai, contohnya jika ada software yang menampilkan form input data identitas, jika user

melengkapi form maka program akan melakukan proses simpan, namun jika user tidak melengkapi form program tidak boleh melakukan proses simpan, jika perangkat lunak tidak sesuai misalnya tidak melengkapi form namun dapat tersimpan, hal ini perlu untuk diperbaiki.

2. Kesalahan interface, dalam hal ini sering terjadi pada software yang tidak diuji coba dengan baik, misalnya tampilan web dengan menggunakan framework, ada beberapa framework yang tidak mendukung dengan beberapa browser, hingga tampilan interface kamu kurang maksimal saat user memakai browser yang tidak mendukung frameword yang kamu gunakan.
3. Kesalahan dalam struktur data atau akses database, yang sering menjadi kendala, karena hal ini dapat berdampak pada akses web kamu menjadi lamban, jika kamu tidak memperhatikannya.
4. Prilaku atau kinerja kesalahan yang ada pada perangkat lunak.
5. Inisialisasi dan penghentian kesalahan pada perangkat lunak.

Tabel II.2. Contoh tabel *Black-Box Testing*

No	Rancangan Proses	Hasil Yang Diharapkan	Hasil	Keterangan
1	Mengisi form login dan klik tombol login	Masuk halaman utama	Sesuai	Jika input benar
2	Klik menu edit profil sekolah	Membuka form input profil sekolah	Sesuai	Jika input benar
3	Mengisi form profil sekolah dan klik simpan	Data tersimpan dan membuka halaman utama	Sesuai	Jika data di input dengan benar
4	Klik menu edit bidang kehalian	Membuka form input bidang keahlian	Sesuai	Jika input benar
5	Mengisi form input bidang keahlian dan klik simpan	Data tersimpan data muncul ditabel bidang keahlian	Sesuai	Jika data sudah valid dan tersimpan di database

Sumber : Pressman (2010:497)