2. LANDASAN TEORI

2.1. *Web Crawler*

Web crawler adalah program komputer yang mencari *world wide web*, dilakukan dengan cara yang sitematis dan otomatis. Beberapa sebutan lain untuk *web crawler* adalah *ants*, *automatic indexer*, *bots*, *worms*, *web spider*, *web robot* atau *web scutter*.

Proses dari *web crawler* disebut dengan *web crawling* atau *spidering*. Banyak site menggunakan *spidering* untuk menyediakan data yang *up to date*. *Crawlers* dapat digunakan untuk memelihara tugas-tugas pada *web site*, seperti mengecek *link* atau melakukan validasi kode HTML. Selain itu, *crawlers* dapat digunakan untuk mengumpulkan informasi secara spesifik dari halaman *web*, seperti melakukan *harvesting* alamat *e-mail*(biasanya untuk spam).

Web crawler merupakan salah satu tipe robot atau agen perangkat lunak. Pada umumnya, *web crawler* dimulai dari sebuah atau beberapa URL yang ingin dikunjungi, disebut *seeds*. Setelah *crawler* mengunjungi URL tersebut, maka *crawler* akan mengidentifikasi semua *hyperlinks* yang ada pada halaman tersebut dan menambahkannya pada halaman yang akan dikunjungi, biasa disebut *crawl frontier*. URL yang berasal dari *frontier* akan dikunjungi secara rekursif sehingga mencapai pada syarat tertentu (Soumen, 2003).

2.2. *Keyphrase extraction*

Keyphrase extraction merupakan proses untuk mencari daftar kata kunci dari sebuah teks. Daftar kata tersebut lebih tepat disebut frase kunci daripada kata kunci, karena lebih sering terdiri dari dua atau lebih kata dibandingkan satu kata.

Sebuah frase didefinisikan sebagai kumpulan dari satu, dua, atau tiga kata yang teratur di dalam teks, dan tidak terdapat *stopword* atau *punctuation* (tanda baca). Setelah itu lakukan *stemming* setiap frase dengan memotong tiap kata di dalam frase menjadi kata dasar.

Frase dengan empat kata atau lebih sangat jarang. Maka dari itu pada algoritma ini hanya mempertimbangkan frase dengan satu, dua, atau tiga kata (Turney, 1999).

2.3. *Parsing*

Dalam bidang tata bahasa dan linguistik, *parsing* adalah sebuah proses yang dilakukan seseorang untuk menjadikan sebuah kalimat menjadi lebih bermakna atau berarti dengan cara memecah kalimat tersebut menjadi kata-kata atau frase-frase (Soumen, 2003).

2.4. *Stopwords and Stemming*

Pada sebagian bahasa terdapat banyak kata fungsi dan kata hubung seperti *preposition* yang muncul dalam jumlah besar tetapi sangat tidak diperlukan dalam pencarian informasi. Seperti kata *a, an, the, on* pada bahasa inggris disebut *stopwords*.

Penghilangan *stopwords* diperlukan untuk dengan alasan untuk mengurangi daftar kata kunci dan meningkatkan performa.

Stemming atau *conflation* adalah perangkat lain untuk membantu sesuai dengan permintaan dengan istilah *morphological* berlainan dalam *corpus*. Proses *Stemming* membuat bentuk sebuah kata menjadi kata dasarnya. Metode umum untuk *stemming* adalah menggunakan kombinasi dari analisa *morphological* (contoh : Algoritma *Porter Stemmer*) (Soumen, 2003).

2.5. Algoritma *Porter Stemmer*

Konsonan pada kata adalah huruf selain A, E, I, O atau U, dan selain Y. Jadi, pada kata TOY huruf konsonannya adalah T dan Y, sedangkan pada kata SYZYG huruf konsonannya adalah S, Z, dan G. Apabila sebuah huruf bukan konsonan maka huruf tersebut adalah huruf vokal.

Huruf konsonan akan diwakilkan dengan C, huruf vokal diwakilkan dengan huruf V. Sejumlah huruf CCC.. dengan panjang lebih dari 0 akan diwakilkan dengan C, dan sejumlah VVV... dengan panjang lebih dari 0 akan diwakilkan dengan V. Semua kata, atau sebagian dari kata, memiliki salah satu dari bentuk berikut :

CVCV...C

CVCV...V

VCVC...C

VCVC...V

Semua bentuk diatas dapat direpresentasikan dengan sebuah bentuk yaitu [C]VCVC...[V]. Dimana tanda kurung siku diartikan bahwa huruf yang diapit boleh ada dan boleh tidak. Dengan menggunakan (VC)m untuk mewakilkan bahwa VC diulang sebanyak m kali, maka sekali lagi bentuk kata dapat dituliskan dengan [C](VC)m[V].

m akan disebut sebagai ukuran dari sebuah kata ketika merepresentasikan bentuk sebelumnya. Untuk kasus m=0 maka berarti null word. Berikut beberapa contoh :

m=0 TR, EE, TREE, BY

m=1 TROUBLE, OATS, TREES, IVY

m=2 TROUBLES, PRIVATE, OATEN, ORRERY

Aturan-aturan untuk menghapus akhiran akan dilakukan dengan bentuk (kondisi) S1→ S2, dimana bentuk ini berarti apabila sebuah kata diakhiri dengan akhiran S1 dan stem sebelum S1 memenuhi konsisi yang diberikan, maka S1 digantikan dengan S2. kondisi biasanya diberikan dengan syarat dari m. contohnya : (m>1) EMENT→null, pada contoh berarti S1 adalah ’EMENT’ dan S2 adalah null. Kondisi ini akan membuat kata ’REPLACEMENT’ menjadi ’REPLAC’, karena ’REPLAC’ termasuk dalam kata dimana m=2.

Kondisi dapat juga berisi beberapa syarat seperti pada Tabel 2.1

Tabel 2.1. Syarat Kondisi

Kondisi	Keterangan
*S	stem diakhiri dengan huruf s (hal yang sama berlaku untuk semua huruf)
v	stem mengandung huruf vokal
*d	stem diakhiri dengan konsonan kembar
*o	stem diakhiri dengan bentuk cvc, dimana c terakhir bukan huruf W, X, atau Y

Bagian kondisi dapat juga mengandung ekspresi dengan *and*, *or* dan *not*. Sehingga bentuk (m>1 and (*S or *T)) merupakan kondisi untuk *stem* yang

memiliki m>1 dan diakhiri dengan S atau T . Algoritma ini harus mengikuti langkah-langkah berurutan.

Aturan-aturan pada tahap pertama dibagi menjadi tiga bagian :

a. Menghilangkan akhiran jamak (*plurals*) pada Tabel 2.2.

Tabel 2.2. Kelompok *Rule* 1a

Suffix	Replacement	Measure Condition	Additional Condition	Example
sses	Ss	NULL	NULL	caresses → caress
ies	i	NULL	NULL	ponies → poni
ss	Ss	NULL	NULL	caress → caress
s	NULL	NULL	NULL	cats → cat

b. Menghilangkan akhiran untuk *past participles* pada Tabel 2.3.

Tabel 2.3. Kelompok *Rule* 1b

Suffix	Replacement	Measure Condition	Additional Condition	Example
eed	ee	m>0	NULL	Feed →feed
				Agreed → agree
ed	NULL	NULL	*v*	plastered → plaster
				bled → bled
ing	NULL	NULL	*v*	Motoring → motor
				sing → sing

Apabila salah satu dari aturan ke 2 dan 3 (ed dan ing) dilakukan maka aturan pada Tabel 2.4. harus dijalankan.

Tabel 2.4. Kelompok *Rule* Tambahan Untuk *Rule* 1b

Suffix	Replacement	Measure Condition	Additional Condition	Example
at	ate	NULL	NULL	conflat → conflate
bl	ble	NULL	NULL	troubl → trouble
iz	ize	NULL	NULL	siz → size
*d		NULL	*L or *S or *Z	Hopp → hop
				tann → tan
				fall → fall
*o		m=1	*o	hiss → hiss
				fail → fail
				fil →file

c. Mengganti huruf Y dengan huruf I. Dapat dilihat pada Tabel 2.5.

Tabel 2.5. Kelompok *Rule* 1c

Suffix	Replacement	Measure Condition	Additional Condition	Example
y	I	NULL	*v*	happy → happi
				sky → sky

Tahap kedua berhubungan dengan pencocokan bentuk dengan beberapa akhiran umum. Semua aturan ini hanya dijalankan apabila kondisi m terpenuhi setelah kata tersebut diubah akhirannya. Aturan-aturan pada tahap kedua ini sebagai berikut pada Tabel 2.6.:

Tabel 2.6. Kelompok *Rule* 2

Suffix	Replacement	Measure Condition	Additional Condition	Example
ational	Ate	m>0	NULL	relational → relate
tional	Tion	m>0	NULL	conditional → condition
Enci	Ence	m>0	NULL	valenci → valence
Anci	Ance	m>0	NULL	hesitanci → hesitance
Izer	Ize	m>0	NULL	digitizer → digitize
Abli	Able	m>0	NULL	conformabli → conformable
Alli	Al	m>0	NULL	radicalli → radical
Entli	Ent	m>0	NULL	differentli → different
Eli	E	m>0	NULL	vileli → vile
Ousli	Ous	m>0	NULL	analogousli → anologous
ization	Ize	m>0	NULL	vietnamization → vietnamize
Ation	Ate	m>0	NULL	predication → predicate
Ator	Ate	m>0	NULL	operator → operate
Alism	Al	m>0	NULL	feudalism → feudal
iveness	Ive	m>0	NULL	decisiveness → decisive
fulness	Ful	m>0	NULL	hopefulness → hopeful
ousness	Ous	m>0	NULL	callousness → callous
Aliti	Al	m>0	NULL	formaliti → formal
Iviti	Ive	m>0	NULL	sensitiviti → sensitive
Biliti	Ble	m>0	NULL	sensibility → sensible

Tahap selanjutnya merupakan tapah ketiga dimana tahap ini berisi tetang aturan yang membahas tentang kata-kata dengan akhiran istimewa. Semua aturan ini hanya dijalankan apabila kondisi m terpenuhi setelah kata tersebut diubah akhirannya. Aturan-aturan pada tahap ketiga ini dapat dilihat dengan lebih jelas pada Tabel 2.7. :

Tabel 2.7. Kelompok *Rule* 3

Suffix	Replacement	Measure Condition	Additional Condition	Example
Icate	ic	m>0	NULL	triplicate → triplic
Ative	NULL	m>0	NULL	formative → form
Alize	al	m>0	NULL	formalize → formal
iciti	ic	m>0	NULL	electriciti → electric
ical	ic	m>0	NULL	electrical → electric
ful	NULL	m>0	NULL	hopeful → hope
ness	NULL	m>0	NULL	goodness → good

Tahap keempat menghilangkan kata yang mengandung terlalu banyak akhiran, dimana kata tersebut harus tetap memenuhi *measure condition* setelah dilakukan penghapusan akhiran. Beberapa akhiran yang dihapuskan antara lain seperti pada Tabel 2.8. :

Tabel 2.8. Kelompok *Rule* 4

Suffix	Replacement	Measure Condition	Additional Condition	Example
al	NULL	m>1	NULL	revival → reviv
ance	NULL	m>1	NULL	allowance → allow
ence	NULL	m>1	NULL	inference → infer
er	NULL	m>1	NULL	airliner → airlin
ic	NULL	m>1	NULL	gyroscopic → gyroscop
able	NULL	m>1	NULL	adjustable → adjust
ible	NULL	m>1	NULL	defensible → defens
ant	NULL	m>1	NULL	irritant → irrit
ement	NULL	m>1	NULL	replacement → replac
ment	NULL	m>1	NULL	adjustment → adjust
ent	NULL	m>1	NULL	dependent → depend
(*S or *T)ion	NULL	m>1	NULL	adoption → adopt
ou	NULL	m>1	NULL	homologou → homolog
ism	NULL	m>1	NULL	communism → commun
ate	NULL	m>1	NULL	activate → activ
iti	NULL	m>1	NULL	angulariti → angular
ous	NULL	m>1	NULL	homologous → homolog
ive	NULL	m>1	NULL	effective → effect
ize	NULL	m>1	NULL	bowdlerize→ bowdler

Tahap kelima yang merupakan tahap terakhir berguna untuk mengecek apakah kata yang diproses diakhiri dengan huruf hidup dan membetulkan sesuai kebutuhan. Aturan-aturan pada tahap kelima ini dibagi dua sebagai berikut :

a. Menghilangkan akhiran E pada Tabel 2.9.

Tabel 2.9. Kelompok *Rule* 5a

Suffix	Replacement	Measure Condition	Additional Condition	Example
e	NULL	m>1	NULL	probate → probat
				rate → rate
e	NULL	m=1	not *o	cease → ceas

b. Menghapus 1 huruf dari kata berakhiran kembar pada Tabel 2.10.

Tabel 2.10. Kelompok *rule* 5b

Suffix	Replacement	Measure Condition	Additional Condition	Example
*d	single letter	m>1	*L	controll → control
				roll → roll

Algoritma ini sangat berhati-hati untuk tidak menghapus akhiran ketika suatu kata yang diproses terlalu pendek. Panjang dari kata ditentukan dari sebuah ukuran. Ukuran tersebut disimbolkan dengan m. Tidak ada dasar *linguistic* untuk pendekatan ini. Hanya saja setelah dilakukan obeservasi m dapat berguna cukup efektif untuk menentukan bijaksana atau tidaknya untuk menghapus suatu akhiran (Porter, Martin, 1980).

2.6. *Regular Expression*

Regular expression adalah cara untuk mencari dan memodifikasi sebuah teks. Mereka dapat mencari *pattern* di dalam sebuah *string*. Meskipun hebat, *regular expression* lebih lambat dibanding fungsi standar *string*. Jadi, bila tidak memerlukan pencarian *string* yang sulit hindarilah penggunaan *regular expression* (Matt, 2000).

Tabel 2.11. *Regular Expression*

simbol	kegunaan
*	menggantikan 0 karakter hingga tak terhingga
+	menggantikan 1 karakter hingga tak terhingga
?	menggantikan 0 atau 1 karakter
^	mencari kata yang diawali oleh *pattern*
$	mencari kata yang diakhiri oleh *pattern*

Tabel 2.11. *Regular Expression* (sambungan)

|	memberikan pilihan
()	membuat *subpattern*

2.7. *TF-IDF (Terms Frequency – Inverse Document Frequency)*

Metode Tf-Idf merupakan suatu cara untuk memberikan bobot hubungan suatu kata (*term*) terhadap dokumen. Metode ini menggabungkan dua konsep untuk perhitungan bobot yaitu, frekuensi kemunculan sebuah kata didalam sebuah dokumen tertentu dan *inverse* frekuensi dokumen yang mengandung kata tersebut. Frekuensi kemunculan kata didalam dokumen yang diberikan menunjukkan seberapa penting kata tersebut didalam dokumen tersebut. Frekuensi dokumen yang mengandung kata tersebut menunjukkan seberapa umum kata tersebut. Sehingga bobot hubungan antara sebuah kata dan sebuah dokumen akan tinggi apabila frekuensi kata tersebt tinggi didalam dokumen dan frekuensi keseluruhan dokumen yang mengandung kata tersebut yang rendah pada kumpulan dokumen.

Dalam perkembangannya metode ini memiliki 4 (empat) proses yang berbeda untuk perhitungan nilai suatu kalimat (Jonas & Kenji, 2005), antara lain :

1. Kecocokan kata-kata pada kalimat dengan daftar kata kunci/ *key word*. Idenya adalah semakin suatu kalimat memiliki nilai yang tinggi, maka kalimat tersebut semakin penting keberadaanya di dalam suatu artikel.
2. Menghitung frekuensi kata-kata suatu kalimat terhadap keseluruhan artikel dan hasilnya akan dibagi dengan jumlah kata dalam suatu artikel. Alasannya adalah semakin tinggi frekuensi kata tersebut di dalam suatu artikel, maka kalimat yang memiliki kata tersebut semakin penting keberadaanya di dalam suatu artikel.
3. Bagian ketiga ini sangat sederhana yaitu hanya melihat posisi kalimat di dalam suatu artikel. Misalkan, untuk 10 (sepuluh) kalimat pertama di dalam suatu artikel akan mendapatkan nilai 2 (dua) sedangkan yang lain akan mendapatkan nilai 1 (satu). Idenya adalah semakin kalimat tersebut berada di awal artikel maka akan memiliki informasi yang penting.

4. Bagian keempat ini sangat berhubungan dengan hasil pemetaan artikel. Karena pada bagian keempat ini akan dihitung jumlah relasi (yang disimbolkan dengan *edge*) suatu kalimat di dalam artikel. Idenya adalah semakin suatu kalimat memiliki relasi yang banyak dengan kalimat lainnya di dalam suatu artikel maka kalimat tersebut kemungkinan mendiskusikan topik utama suatu artikel sehingga kalimat tersebut semakin penting keberadaannya di dalam artikel tersebut.

Kemudian hasil dari 4 (empat) perhitungan tersebut akan dikalikan dan menjadi nilai dari *tf*, seperti terlihat pada persamaan berikut :

$$tf = bobot\,1 * bobot\,2 * bobot\,3 * bobot\,4 \quad (2.1)$$

Dalam sistem ini, akan digunakan perhitungan TF dasar dan akan digunakan proses perkembangan yang ketiga saja. Nilai dari *tf* akan dikalikan dengan nilai *idf* seperti pada persamaan dibawah ini :

$$w = tf \times idf \quad (2.2)$$

$$w = tf \times \log \frac{N}{n} \quad (2.3)$$

Keterangan :

w = bobot kalimat terhadap dokumen

tf = jumlah kemunculan kata/*term* dalam dokumen

N = jumlah semua dokumen yang ada dalam *database*

n = jumlah dokumen yang mengandung kata/*term*

Berdasarkan rumus diatas, berapapun besarnya nilai *tf*, apabila N = n maka akan didapatkan hasil 0 (nol) untuk perhitungan *idf*. Untuk itu dapat ditambahkan nilai 1 (satu) pada sisi *idf*, sehingga perhitungan bobotnya menjadi sebagai berikut (Rolly & Andrew, 2005):

$$w = tf \times \left(\log\left(N/n\right)+1\right) \tag{2.4}$$

2.8. *Jaccard's Coefficient*

Jaccard's Coefficient adalah salah satu metode yang dipakai untuk menghitung *similarity* antara dua objek. Secara umum perhitungan metode ini didasarkan pada *vector space similarity measure. Jaccard similarity* atau *jaccard Coefficient* menghitung *similarity* antara dua objek, X dan Y yang dinyatakan dalam dua buah vektor (Soumen, 2003), sebagai berikut :

$$J(X,Y) = \frac{\sum_{i=1}^{p} x_i y_i}{\sum_{i=1}^{p} x_i^{\,2} + \sum_{i=1}^{p} y_i^{\,2} - \sum_{i=1}^{p} x_i y_i} \tag{2.5}$$

Dimana x,y merupakan hasil dari perhitungan *dot product* dari X dan Y. hal ini dapat dengan lebih mudah dideskripsikan sebagai :

$$\frac{X \cap Y}{X \cup Y}$$

2.9. *Hypertext Preprocessor* (PHP)

PHP singkatan dari PHP *Hypertext Preprocessor*. PHP merupakan bahasa berbentuk skrip yang ditempatkan dalam *server* dan diproses di *server*. Hasilnya akan dikirim ke *client*, tempat pemakai menggunakan *browser*. Secara khusus, PHP dirancang untuk membentuk *web* dinamis. Artinya PHP dapat membentuk suatu tampilan berdasarkan permintaan terkini (Matt, 2000).

2.9.1. Skrip *Hypertext Preprocessor* (PHP)

Skrip PHP berkedudukan sebagai tag dalam bahasa HTML. *HyperText Markup Language* (HTML) adalah bahasa standar untuk membuat halaman-halaman *web*. Kode PHP diawali dengan <? php dan diakhiri dengan ?>. Pasangan kedua kode inilah yang berfungsi sebagai tag kode PHP. Berdasarkan tag inilah, pihak *server* dapat memahami kode PHP dan kemudian memprosesnya. Hasilnya dikirim ke *browser*.

2.9.2. Variabel PHP

Variabel digunakan sebagai tempat penyimpanan data sementara. Data yang disimpan dalam variabel akan hilang setelah program selesai dieksekusi. Untuk dapat menggunakan variabel, ada dua langkah yang harus dilakukan yaitu deklarasi dan inisialisasi.

Dalam PHP, deklarasi variabel seringkali digabung dengan inisialisasi. Ada beberapa aturan yang diikuti berkenaan dengan penggunana nama variabel. Aturan pemberian nama Variabel adalah sebagai berikut :

- Pendeklarasian variabel dalam PHP dimulai dengan tanda $.

  Contoh :

```
$user
$aVar
```

- Karakter pertama harus huruf atau garis bawah (_)
- Karakter berikutnya boleh huruf, angka, atau garis bawah

Inisialisasi variabel adalah mengisi nilai untuk pertama kalinya ke dalam variabel. Contoh inisialisasi :

```
$user = ”Bob”;
$aVar = “42”;
```

2.9.2.1. Fungsi-fungsi pada PHP

PHP menyediakan banyak fungsi yang dapat memanipulasi *string* dan data dalam *array*. Berikut Tabel 2.12 tentang fungsi-fungsi *string* pada PHP.

Tabel 2.12. Fungsi *String*

Nama fungsi	Kegunaan	Contoh	hasil
strlen()	menghitung panjang *string*	$word="test"; $len=strlen($word);	$len=4
strpos()	mencari posisi *string*	$sample="mz00xyz" $x=$strpos($sample,"00");	$x=2
substr()	memotong sebagian *string*	$sample="scallywag"; $a=substr($sample,6); $b=substr($sample,6,2); $c=substr($sample,-3);	$a=wag $b=wa $c=wag
str_replace()	Mengganti semua *string* yang dicari dengan *string* pengganti	$a=str_replace("%body", "black", "<body text ='%body%'>");	<body text ='black'>

Tabel 2.13. Fungsi *Array*

Nama fungsi	Kegunaan	Contoh	hasil
Array_diff	Membandingkan 2 buah array dan mengembalikan perbedaannya	$a=array("red","blue","green"); $b=array("green","yellow","grey"); $c=array_diff($a,$b);	Array([1]=>blue)
Array_count_ values	Menghitung semua jumlah *value* dari array	$a=array("z",1,2,1,"z"); $b=array_count_values($a);	Array ([1] =>2 [2]=>1 [z]=>2)
Array_key_ex sist	*Return true* apabila nilai yang diberikan ada pada *array key*	$a=array('first'=>1,'second'=> 4); If (array_key_exist('first',$a)) Echo "ada";	ada
Array_unique	Menghilangkan *value* yang kembar	$a=array("1"=>"red","grey","blue", "red") $b=array_unique($a);	Array ([0]=>red [1]=>grey [2]=>blue)
Array_push	Menambahkan sebuah elemen pada akhir dari *array*	$a = array("red","blue"); array_push($a, "yellow");	Array ([0]=>red [1]=>blue [2]=>yellow
In_array	*Return true* apabila *nilai* yang diberikan ada pada *array value*	$a=array("Mac","NT","Irix"); if (in_array("Irix",$a)) echo "Got Irix";	Got Irix

2.9.3. PHP dan *Database*

PHP dibentuk agar dapat mudah berintegrasi dengan *database*. Fitur ini adalah salah satu faktor yang membuat bahasa PHP sebagai pilihat yang tepat untuk membuat aplikasi *web*. Banyak *database* yang didukung secara langsung, termasuk InterBase, dBASE, mSQL, MySQL, Oracle, PostgreSQL, dan lainnya.

2.9.4. Koneksi PHP dan MySQL

Langkah pertama yang harus dilakukan untuk dapat menghubungkan program PHP dengan *database* MySQL adalah membuka koneksi. Untuk membuka koneksi membutuhkan alamat *server* serta *username* dan *password* untuk *database*. Perintah PHP untuk membuka koneksi adalah sebagai berikut :

```
$dbServer = “localhost”;
$dbUser= “harry”;
$dbPass = “elbomonkey”;
$dbConn = mysql_connect ($dbServer, $dbUser, $dbPass);
```

Koneksi dengan *database* disimpan dalam variabel $dbConn untuk digunakan pada langkah-langkah memilih dan membuka *database* yang diinginkan. Perintah untuk memilih dan membuka *database* sample adalah sebagai berikut :

```
$dbName = “sample”;
Mysql_select_db ($dbName);
```

Database telah siap digunakan untuk memasukkan data. Perintah memasukkan data dalam *script* PHP adalah sebagai berikut:

```
$query = “INSERT INTO userTable VALUES (1, ‘Endy’, ‘sample’)”;
If (!mysql_query($query)) {
    $dberror=mysql_error();
}
Else {echo “sukses”;}
```

Setelah semua data masuk ke dalam *database*, data tersebut dapat diakses untuk ditampilkan. Langkah-langkah untuk mengakses data dalam *database*:

- Membuat sambungan *database*
- Memilih *database*
- Membuat *query*
- Menjalankan *query*
- Mengambil hasilnya
- Memproses hasilnya

Perintah untuk mengakses *query* dalam *database* dibuat dengan menggunakan aturan SQL. *Script* PHP untuk mengakses *query* adalah sebagai berikut:

```
$query = SELECT * FROM UserTabel WHERE userName='$login'
```

Script PHP untuk mengeksekusi *query* adalah sebagai berikut:

```
$hasil = mysql_query ($query);
```

Script PHP untuk mengetahui jumlah hasil yang didapat dari *query* adalah sebagai berikut:

```
$jumlahHasil = mysql_num_rows ($hasil);
```

Script PHP untuk mengambil data dalam *database* adalah sebagai berikut:

```
$data = mysql_fetch_array ($hasil);
$passdb = $data ["Password"];
```

2.10. MySQL

MySQL (*My Structured Query Language*) adalah salah satu jenis *database server* yang sangat terkenal. Hal tersebut dikarenakan *MySQL* menggunakan *SQL* sebagai bahasa dasar untuk mengakses *database*. *SQL* adalah suatu bahasa permintaan terstruktur yang telah distandarkan untuk semua program pengakses *database*. Pada MySQL, sebuah *database* mengandung satu atau sejumlah tabel. Setiap tabel terdiri atas sejumlah baris dan sejumlah kolom. Perpotongan tiap baris dan kolom merupakan tempat data disimpan (Matt, 2000).