BAB II

KAJIAN TEORITIS DAN KERANGKA BERPIKIR

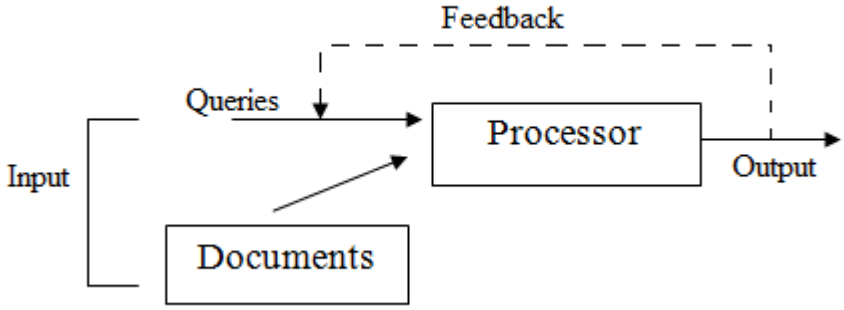
2.1. Kajian Teoritis

2.1.1. *InformationRetrieval*

Sejak tahun 1940-an masalah penyimpanan informasi dan pengambilan informasi telah menarik perhatian. Masalah yang ada, kita memiliki sejumlah besar informasi yang akurat dan cepat akses menjadi semakin sulit. Salah satu efek dari hal ini adalah bahwa informasi yang relevan akan diabaikan karena tidak pernah ditemukan, yang pada gilirannya menyebabkan banyak duplikasi pekerjaan dan usaha. Dengan munculnya komputer, banyak ide telah diberikan untuk menggunakan komputer dalam menyediakan sistem pencarian yang cepat dan cerdas. Diperpustakaan, tentunya banyak memiliki penyimpanan informasi dan pengambilan masalah, beberapa lebih tugas-tugas rutin, seperti katalogisasi dan administrasi umum, telah berhasil diambil alih oleh komputer. Namun, masalah pengambilan yang efektif sebagian besar masih belum terpecahkan. Pada prinsipnya, penyimpanan informasi dan pengambilan sederhana. Seorang pengguna tidak mempunyai waktu atau tidak ingin menghabiskan waktu membaca koleksi seluruh dokumen, terlepas dari kenyataan bahwa mungkin secara fisik tidak mungkin baginya untuk melakukannya. Ketika komputer kecepatan tinggi menjadi tersedia untuk non-numerik kerja, banyak yang mengira bahwa komputer akan mampu membaca koleksi seluruh dokumen untuk mengekstrak dokumen yang relevan.

Menggunakan teks *natural language*dari dokumen tidak hanya disebabkan masukan dan penyimpanan masalah, tetapi juga masalah intelektual dari karakteristik dokumen konten. Bisa dibayangkan bahwa perkembangan *hardware* masa depan dapat membuat *natural language* masukan dan penyimpanan yang lebih layak. Saat perangkat lunak 'membaca', mencoba untuk mengekstrak informasi, baik sintaksis dan semantik dari teks dan menggunakannya untuk memutuskan apakah setiap dokumen relevan atau tidak untuk permintaan tertentu. Kesulitan tidak hanya mengetahui cara mengekstrak informasi tetapi juga bagaimana menggunakannya untuk memutuskan relevansi. Sudah ada gagasan 'relevansi' kedalam diskusi, sekarang gagasan berada di pusat pencarian informasi. Tujuan dari strategi information retrieval adalah untuk mengambil semua dokumen yang relevan pada saat yang sama mungkin dengan mengambil beberapa non-relevan. Ketika karakterisasi dokumen bekerja keluar, sedemikian rupa dokumen itu mewakili relevan dengan query. Index manusia secara tradisional ditandai ketika dokumen menetapkan istilah indeks dokumen. Upaya indexer untuk mengantisipasi jenis istilah indeks pengguna dalam mengambil setiap dokumen yang akan menjelaskan isinya. Secara implisit indeks membangun permintaan untuk dokumen yang relevan. Ketika pengindeksan dilakukan secara otomatis diasumsikan bahwa dengan mendorong teks dokumen atau query melalui analisis otomatis yang sama, output akan menjadi representasi dari konten, jika dokumen relevan dengan query. Intelektual adalah mungkin bagi manusia untuk menetapkan relevansi dokumen ke query. Dalam melakukan

pencarian di komputer, perlu membangun sebuah model dimana keputusan relevansi dapat diukur.



Gambar 2.1 Sistem *information retrieval*

Sebagian besar sistem komputer mengambil penyimpanan hanya representasi dari dokumen (query) yang berarti bahwa teks dari dokumen hilang setelah diproses untuk menghasilkan representasinya. Ketika sistem pengambilan adalah *on-line*, pengguna akan mungkin mengubah query selama satu sesi pencarian. Prosedur seperti itu disebut sebagai umpan balik. Processor, bagian dari sistem pencarian yang bersangkutan dengan proses pengambilan. Proses mungkin melibatkan penataan informasi dalam beberapa cara yang tepat, seperti mengklasifikasi. Dimana akan melaksanakan strategi pencarian dalam menanggapi query. Pada diagram, dokumen telah ditempatkan dalamkotak terpisah untuk menekankan fakta bahwa dokumen tidak hanya input tetapi dapat digunakan selama proses pengambilan sedemikian rupa sehingga struktur dokumen lebih tepat dilihat sebagai bagian dari proses pengambilan.[1]

[1] Van RIJSBERGEN, C.J., *Information Retrieval*.Department of Computing Science.University of Glasgow.hlm.3

2.1.2.*Full-Text Searching*

Dalam pembahasan *full-text searching* akan membahas Fungsi *Full-Text Search, Natural Language Full-Text Searches, Full-Text Stopwords, Full-Text Restrictions, Fine-Tuning MySQL Full-Text Search* dan*Adding a Collation for Full-Text Indexing.*

2.1.2.1.Fungsi *Full-Text Search*

```
search_modifier:
 {
    IN NATURAL LANGUAGE MODE
  | IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION
  | IN BOOLEAN MODE
  | WITH QUERY EXPANSION
 }
```

Gambar 2.2*ScriptFull-Text Search Functions*(Stefan Hinz, et al. 2012)

MySQL mendukung untuk *Full-text* indeks dan pencarian. Indeks *Full-text* dalam MySQL adalah indeks yang berjenis *FULLTEXT*.Indeks *Full-Text* hanya dapat digunakan dengan tabel MyISAM, dan dapat dibuat hanya untuk CHAR, VARCHAR, atau kolom TEXT.Definisi indeks *full-text* dapat diberikan dalam pernyataan `CREATE TABLE` ketika sebuah tabel dibuat, kemudian ditambahkan menggunakan `ALTER TABLE` atau `CREATE INDEX`. Untuk set data yang besar, lebih cepat untuk memuat data ke dalam tabel yang tidak memiliki indeks *full-text*kemudian membuat indeks, selain untuk memuat data ke dalam tabel yang memiliki indeks *FULLTEXT* yang ada.Pencarian *Full-Text* dilakukan dengan menggunakan sintaks `MATCH () … AGAINST. MATCH`

() melakukan pencarian dari kata yang dipisahkan oleh koma dan nama kolom. AGAINST melakukan pencarian dengan mengambil string dan pengubah opsional yang menunjukkan jenis. Pencarian string harus menjadi string yang bernilai konstan selama evaluasi query. String yang bernilai konstan merupakan aturan, misalnya, kolom tabel dapat berbeda untuk setiap baris.

Ada tiga jenis *Full-Text Search* :

1. Sebuah pencarian menafsirkan bahasa bawaan pada string pencarian sebagai frase dalam bahasa bawaan manusia (frase di teks bebas). Tidak ada operator khusus. Daftar *stopword* berlaku. Kata-kata yang hadir dalam 50% atau lebih dari baris yang dianggap umum dan tidak cocok. *Full-Text Search* adalah pencarian bahasa bawaan jikaIN NATURAL LANGUAGE MODE pengubah mode dalam bahasa bawaan diberikan atau jika tidak ada pengubah diberikan. Untuk informasi lebih lanjut, lihat bagian 2.1.2.2.*Natural Language Full-Text Searches*
2. Sebuah pencarian boolean menafsirkan string pencarian menggunakan aturan bahasa query khusus. String berisi kata-kata untuk mencari. Dapat berisi operator yang menentukan persyaratan sehingga kata harus hadir atau tidak ada di baris yang cocok, atau bahwa harus berbobot lebih tinggi atau lebih rendah dari biasanya. Kata-kata umum seperti "*some*" atau "*then*" adalah *stopwords* dan tidak cocok jika hadir dalam string pencarian. IN BOOLEAN MODE menentukan dalam pencarian boolean.
3. Sebuah pencarian ekspansi permintaan merupakan modifikasi dari pencarian bahasa bawaan. String pencarian digunakan untuk melakukan

pencarian bahasa bawaan. Kemudian kata-kata dari baris paling relevan dikembalikan oleh pencarian akan ditambahkan ke string pencarian dan pencarian dilakukan lagi. Permintaan mengembalikan baris dari pencarian kedua. `IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION` atau pengubah `WITH QUERY EXPANSION` menentukan pencarian ekspansi permintaan.

Kendala pencarian teks lengkap tercantum dalam bagian 2.1.2.4.*Full-Text Restrictions*

Utilitas myisam_ftdump dapat digunakan untuk membuang isi dari indeks teks lengkap.Ini membantu untuk *debugging* teks lengkap query.[2]

2.1.2.2.*NaturalLanguage Full-Text Searches*

Secara *default* atau dengan pengubah `IN NATURAL LANGUAGE MODE`, fungsi `MATCH ()` melakukan pencarian bahasa bawaan untuk string terhadap koleksi teks.Koleksi adalah kumpulan dari satu atau lebih kolom termasuk dalam indeks *FULLTEXT*.String pencarian diberikan sebagai argumen untuk `AGAINST()`.Untuk setiap baris dalam tabel, `MATCH ()` mengembalikan nilai relevansi, yaitu, ukuran kesamaan antara string pencarian dan teks dalam baris yang di kolom yang disebutkan dalam daftar `MATCH ()`.

[2]Stefan Hinz, et al.,"12.9. *Full-Text Search Functions*", Aviation Today, diakses darihttp://dev.mysql.com/doc/refman/5.5/en/fulltext-search.html, pada tanggal 3 Desember 2012 pukul 08:17.

```
mysql> CREATE TABLE articles (
    -> id INT UNSIGNED AUTO_INCREMENT NOT NULL PRIMARY KEY,
    -> title VARCHAR(200),
    -> body TEXT,
    -> FULLTEXT (title,body)
    -> ) ENGINE=MyISAM;
Query OK, 0 rows affected (0.00 sec)

mysql> INSERT INTO articles (title,body) VALUES
    -> ('MySQL Tutorial','DBMS stands for DataBase ...'),
    -> ('How To Use MySQL Well','After you went through a ...'),
    -> ('1001 MySQL Tricks','1. Never run mysqld as root. 2. ...'),
    -> ('MySQL vs. YourSQL','In the following database comparison ...'),
    -> ('MySQL Security','When configured properly, MySQL ...');
Query OK, 6 rows affected (0.00 sec)
Records: 6  Duplicates: 0  Warnings: 0

mysql> SELECT * FROM articles
    -> WHERE MATCH (title,body)
    -> AGAINST ('database' IN NATURAL LANGUAGE MODE);
+----+-----------------+---------------------------------------+
| id | title           | body                                  |
+----+-----------------+---------------------------------------+
|  5 | MySQL vs. YourSQL | In the following database comparison ... |
|  1 | MySQL Tutorial    | DBMS stands for DataBase ...          |
+----+-----------------+---------------------------------------+
2 rows in set (0.00 sec)
```

Gambar 2.3*Script Natural Language Full-Text Searches*(Stefan Hinz, et al.2012)

Secara default, pencarian dilakukan dalam *case-insensitive*.Namun, dapat melakukan *case-sensitive full-text search* dengan menggunakan pemeriksaan biner untuk kolom diindeks. Misalnya, kolom yang menggunakan *latin1character* dapat dibuat menjadi latin1_bin untuk membuat *case-sensitive* untuk *full-text searches*.

Ketika `MATCH ()` digunakan dalam klausa `WHERE`, seperti pada contoh yang ditunjukkan sebelumnya, baris kembali secara otomatis diurutkan dengan relevansi tertinggi pertama.Relevansi nilai *nonnegative floating-point*.Relevansi Nol berarti tidak ada kesamaan.Relevansi dihitung berdasarkan jumlah kata dalam baris, jumlah kata unik dalam baris, jumlah kata dalam koleksi, dan jumlah dokumen (baris) yang mengandung kata tertentu.

Untuk *count match*, bisa menggunakan query seperti :

```
mysql> SELECT COUNT(*) FROM articles
    -> WHERE MATCH (title,body)
    -> AGAINST ('database' IN NATURAL LANGUAGE MODE);
+----------+
| COUNT(*) |
+----------+
|        2 |
+----------+
1 row in set (0.00 sec)
```

Gambar 2.4*Script Count Match* menurut relevansi (Stefan Hinz, et al. 2012)

Namun, ada kemungkinan menemukan lebih cepat untuk menulis ulang query sebagai berikut :

```
mysql> SELECT
    -> COUNT(IF(MATCH (title,body) AGAINST ('database' IN NATURAL LANGUAGE
MODE), 1, NULL))
    -> AS count
    -> FROM articles;
+-------+
| count |
+-------+
|     2 |
+-------+
1 row in set (0.03 sec)
```

Gambar 2.5*Script Count Match*tidak menurut relevansi(Stefan Hinz, et al. 2012)

Query pertama menurut relevansi sedangkan yang kedua tidak. Namun, permintaan kedua melakukan scan tabel penuh dan yang pertama tidak. Yang pertama mungkin lebih cepat jika pencarian cocok beberapa baris, jika tidak, kedua mungkin lebih cepat karena akan membaca banyak baris pula.

Untuk natural language *full-text searches*, persyaratan bahwa kolom yang disebutkan dalam fungsi `MATCH ()` menjadi kolom yang sama termasuk dalam beberapa indeks *fulltext* tabel. Untuk catatan, permintaan sebelumnya bahwa kolom yang disebutkan dalam fungsi `MATCH ()` (title dan body) adalah sama seperti yang disebutkan dalam definisi *article* indeks *fulltext*. Jika ingin mencari *title* atau *body* secara terpisah, perlu untuk membuat indeks *fulltext* terpisah untuk setiap kolom.

Hal ini juga memungkinkan untuk melakukan pencarian Boolean atau pencarian dengan ekspansi query.*Full-Text Search* menggunakan indeks nama kolom hanya dari satu tabel dalam klausa `MATCH ()` karena indeks tidak dapat span beberapa tabel. Sebuah pencarian boolean dapat dilakukan tanpa adanya indeks (meskipun lebih lambat), dalam hal ini adalah mungkin untuk nama kolom dari beberapa tabel.

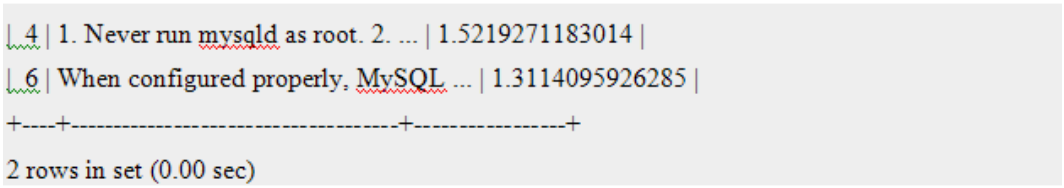
Contoh sebelumnya adalah ilustrasi dasar yang menunjukkan bagaimana menggunakan fungsi `MATCH ()` dimana baris yang dikembalikan dalam rangka penurunan relevansi.Contoh berikut menunjukkan bagaimana untuk mengambil nilai-nilai relevansi eksplisit.Baris yang dikembalikan tidak memerintahkan pernyataan `SELECT` tidak termasuk `WHERE` atau klausa `ORDER BY`.

```
mysql> SELECT id, MATCH (title,body)
    -> AGAINST ('Tutorial' IN NATURAL LANGUAGE MODE) AS score
    -> FROM articles;
+----+-----------------+
| id | score           |
+----+-----------------+
|  1 | 0.65545833110809 |
|  2 |               0 |
|  3 | 0.66266459226608 |
|  4 |               0 |
|  5 |               0 |
|  6 |               0 |
+----+-----------------+
6 rows in set (0.00 sec)
```

Gambar 2.6*Script* nilai-nilai relevansi eksplisit (Stefan Hinz, et al. 2012)

Contoh berikut adalah lebih kompleks.Permintaan mengembalikan nilai relevansi dan juga macam baris dalam urutan penurunan relevansi. Untuk mencapai hasil, harus menentukan `MATCH ()` dua kali: sekali dalam daftar `SELECT` dan sekali dalam klausa `WHERE`. Hal ini tidak menyebabkan overhead tambahan, karena pemberitahuan optimizer MySQL bahwa dua `MATCH ()` panggilan yang identik dan memanggil kode *full-text search* hanya sekali.

```
mysql> SELECT id, body, MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root'
    -> IN NATURAL LANGUAGE MODE) AS score
    -> FROM articles WHERE MATCH (title,body) AGAINST
    -> ('Security implications of running MySQL as root'
    -> IN NATURAL LANGUAGE MODE);
+----+-----------------------------------+----------------+
| id | body                              | score          |
+----+-----------------------------------+----------------+
```



Gambar 2.7*Script* mengembalikan nilai relevansi dan juga macam baris dalam urutan penurunan relevansi (Stefan Hinz, et al. 2012)

Implementasi MySQL *full-text* menganggap setiap urutan karakter kata yang benar (huruf, angka, dan garis bawah) sebagai sebuah kata. Urutan itu juga berisi tanda petik (“ ’ ”), tetapi tidak lebih dari satu berturut-turut. Berarti bahwa aaa’bbb dianggap sebagai satu kata, tetapi aaa” bbb dianggap sebagai dua kata. Tanda petik di awal atau akhir dari sebuah kata yang dilucuti oleh parser *fulltext*; ‘aaa’bbb’ akan diuraikan sebagai aaa’bbb.

Parser *full-text*menentukan mana kata-kata mulai dan berakhir dengan mencari karakter pembatas tertentu, misalnya, “” (spasi), “,” (koma), dan “.” (periode). Jika kata-kata tidak dipisahkan oleh pembatas (seperti dalam, misalnya, *Chinese*), parser *full-text* tidak dapat menentukan mana kata mulai atau berakhir. Untuk dapat menambahkan kata-kata atau istilah lainnya diindeks dalam bahasa tersebut untuk indeks *full-text*, harus diproses sehingga dapat dipisahkan oleh beberapa pembatas seperti “ ” ”.

Beberapa kata diabaikan dalam *full-text searches*:

1. Setiap kata yang terlalu pendek diabaikan. Standar minimum panjang kata yang ditemukan oleh *full-text searches* adalah empat karakter.
2. Kata-kata dalam daftar *stopword* diabaikan. *Stopword* adalah kata seperti “*the*” atau “*some*” yang begitu umum bahwa dianggap memiliki nilai

semantik nol. Ada daftar *built-in stopword*, tetapi dapat ditimpa oleh sebuah daftar yang ditetapkan pengguna.

Daftar *stopword default* diberikan dalam bagian 2.1.2.3.*Full-Text Stopwords*. Standar minimum panjang kata dan daftar *stopword* dapat diubah seperti dijelaskan dalam bagian 2.1.2.5.*Fine-tuning MySQL Full-Text Search*.

Setiap kata yang benar dalam koleksi dan query tertimbang menurut maknanya. Akibatnya, sebuah kata yang hadir dalam banyak dokumen memiliki bobot yang lebih rendah (dan bahkan mungkin memiliki berat nol), karena memiliki nilai yang lebih rendah semantik dalam koleksi tertentu. Sebaliknya, jika kata tersebut jarang terjadi, ia menerima bobot yang lebih tinggi. Bobot dari kata – kata digabungkan untuk menghitung relevansi dari baris.

Untuk tabel yang sangat kecil, distribusi kata tidak cukup mencerminkan nilai semantik dan model ini kadang-kadang dapat menghasilkan hasil yang aneh.Sebagai contoh, meskipun kata "MySQL" hadir dalam setiap baris dari tabel artikel ditunjukkan sebelumya, pencarian untuk kata tidak menghasilkan hasil.

```
mysql> SELECT * FROM articles
    -> WHERE MATCH (title,body)
    -> AGAINST ('MySQL' IN NATURAL LANGUAGE MODE);
Empty set (0.00 sec)
```

Gambar 2.8*Script* kata yang tidak cukup mencerminkan nilai semantic dan diperlakukan sebagai *stopword*(Stefan Hinz, et al. 2012)

Hasil pencarian habis karena kata "MySQL" hadir dalam setidaknya 50% dari baris.Dengan demikian, efektif diperlakukan sebagai suatu *stopword*. Untuk set data yang besar, ini adalah perilaku yang paling diinginkan. `Natural`

`Language` query seharusnya tidak kembali setiap baris kedua dari tabel 1 GB. Untuk set data kecil, mungkin kurang diinginkan.

Sebuah kata yang cocok setengah dari baris dalam sebuah tabel lebih kecil kemungkinannya untuk menemukan dokumen yang relevan.Bahkan, kemungkinan besar menemukan banyak dokumen yang tidak relevan.Terjadi terlalu sering ketika mencoba untuk menemukan sesuatu di *internet* dengan mesin pencari. Dengan alasan bahwa baris yang mengandung kata ditugaskan nilai semantik rendah untuk set data tertentu. Sebuah kata yang diberikan dapat mencapai ambang batas 50% dalam suatu set data tetapi tidak yang lain.

Ambang batas 50% memiliki implikasi signifikan ketika pertama kali mencoba mencari teks lengkap untuk melihat cara kerjanya. Jika membuat tabel dan memasukkan hanya satu atau dua baris teks ke dalamnya, setiap kata dalam teks terjadi pada setidaknya 50% dari baris.Akibatnya, pencarian tidak mengembalikan hasil apapun.Pastikan untuk memasukkan setidaknya tiga baris dan lebih.[3]

2.1.2.3. *Full-Text Stopwords*

Daftar *stopword* dimuat dan mencari query *full-text* menggunakan server set karakter dan *collation* (dimana nilai-nilai pada thecharacter_set_server dan sistem variabel pada collation_server). Salah *hits* atau meleset mungkin terjadi

[3]Stefan Hinz, et al.,"12.9.1 *Natural Language Full-Text Searches*", Aviation Today, diakses darihttp://dev.mysql.com/doc/refman/5.5/en/fulltext-natural-language.htmlpada tanggal 3 Desember 2012 pukul 08:20.

untuk pencarian *stopword* jika file *stopword* atau kolom yang digunakan untuk pengindeksan *full-text* atau *searches* memiliki satu set karakter atau *collation* yang berbeda dari character_set_server atau collation_server.

Case sensitivity stopword tergantung pada pemeriksaan server. Sebagai contoh, pencarian bersifat *case insensitive* jika *collation* adalah latin1_swedish_ci, sedangkan pencarian *case sensitive* jika *collation* adalah latin1_general_cs atau latin1_bin.

Tabel berikut menunjukkan daftar *default* dari *full-text stopwords*. Dalam *MySQL source distribution*, dapat menemukan daftar default di thestorage/myisam/ft_static.c file.[4]

Tabel 2.1 Daftar *default* dari *full-text stopwords*(Stefan Hinz, et al. 2012)

a's	able	About	Above	according
Accordingly	across	Actually	After	afterwards
Again	against	ain't	All	Allow
Allows	almost	Alone	Along	Already
Also	although	Always	Am	Among
Amongst	an	And	Another	Any
Anybody	anyhow	Anyone	Anything	Anyway
Anyways	anywhere	Apart	Appear	Appreciate

[4]Stefan Hinz, et al.,"12.9.4 *Full-Text Stopwords*", Aviation Today, diakses dari http://dev.mysql.com/doc/refman/5.5/en/fulltext-stopwords.html pada tanggal 3 Desember 2012 pukul 08:38.

Appropriate	are	aren't	Around	As
Aside	ask	Asking	associated	At
Available	away	Awfully	Be	Became
Because	become	Becomes	Becoming	Been
Before	beforehand	Behind	Being	Believe
Below	beside	Besides	Best	Better
Between	beyond	Both	Brief	But
By	c'mon	c's	Came	Can
can't	cannot	Cant	Cause	Causes
Certain	certainly	Changes	Clearly	Co
Com	come	Comes	concerning	Consequently
Consider	considering	Contain	containing	Contains
corresponding	could	couldn't	Course	Currently
Definitely	described	Despite	Did	didn't
Different	do	Does	doesn't	Doing
don't	done	Down	downwards	During
Each	edu	Eg	Eight	Either
Else	elsewhere	Enough	Entirely	Especially
Et	etc	Even	Ever	Every
Everybody	everyone	Everything	everywhere	Ex
Exactly	example	Except	Far	Few
Fifth	first	Five	Followed	Following

Follows	for	Former	Formerly	Forth
Four	from	Further	furthermore	Get
Gets	getting	Given	Gives	Go
Goes	going	Gone	Got	Gotten
Greetings	had	hadn't	Happens	Hardly
Has	hasn't	Have	haven't	Having
He	he's	Hello	Help	Hence
Her	here	here's	Hereafter	Hereby
Herein	hereupon	Hers	Herself	Hi
Him	himself	His	Hither	Hopefully
How	howbeit	However	i'd	i'll
i'm	i've	Ie	If	Ignored
Immediate	In	Inasmuch	Inc	Indeed
Indicate	indicated	Indicates	Inner	Insofar
Instead	into	Inward	Is	isn't
It	it'd	it'll	it's	Its
Itself	just	Keep	Keeps	Kept
Know	known	Knows	Last	Lately
Later	latter	Latterly	Least	Less
Lest	let	let's	Like	Liked
Likely	little	Look	Looking	Looks
Ltd	mainly	Many	May	Maybe

Me	mean	Meanwhile	Merely	Might
More	moreover	Most	Mostly	Much
Must	my	Myself	Name	Namely
Nd	near	Nearly	Necessary	Need
Needs	neither	Never	nevertheless	New
Next	nine	No	Nobody	Non
None	noone	Nor	Normally	Not
Nothing	novel	Now	Nowhere	Obviously
Of	off	Often	Oh	Ok
Okay	old	On	Once	One
Ones	only	Onto	Or	Other
Others	otherwise	Ought	Our	Ours
Ourselves	out	Outside	Over	Overall
Own	particular	particularly	Per	Perhaps
Placed	please	Plus	Possible	Presumably
Probably	provides	Que	Quite	Qv
Rather	Rd	Re	Really	Reasonably
Regarding	regardless	Regards	Relatively	Respectively
Right	said	Same	Saw	Say
Saying	says	Second	Secondly	See
Seeing	seem	Seemed	Seeming	Seems
Seen	self	Selves	Sensible	Sent

Serious	seriously	Seven	Several	Shall
She	should	shouldn't	Since	Six
So	some	Somebody	Somehow	Someone
Something	sometime	Sometimes	somewhat	Somewhere
Soon	sorry	Specified	Specify	Specifying
Still	sub	Such	Sup	Sure
t's	take	Taken	Tell	Tends
Th	than	Thank	Thanks	Thanx
That	that's	Thats	The	Their
Theirs	them	themselves	Then	Thence
There	there's	Thereafter	Thereby	Therefore
Therein	theres	Thereupon	These	They
they'd	they'll	they're	they've	Think
Third	this	Thorough	thoroughly	Those
Though	three	Through	throughout	Thru
Thus	To	Together	Too	Took
Toward	towards	Tried	Tries	Truly
Try	trying	Twice	Two	Un
Under	unfortunately	Unless	Unlikely	Until
Unto	up	Upon	Us	Use
Used	useful	Uses	Using	Usually
Value	various	Very	Via	Viz

Vs	Want	Wants	Was	wasn't
Way	We	we'd	we'll	we're
we've	Welcome	Well	Went	Were
weren't	What	what's	Whatever	When
Whence	Whenever	Where	where's	Whereafter
Whereas	Whereby	Wherein	whereupon	Wherever
Whether	Which	While	Whither	Who
who's	Whoever	Whole	Whom	Whose
Why	Will	Willing	Wish	With
Within	Without	won't	Wonder	Would
wouldn't	Yes	Yet	You	you'd
you'll	you're	you've	Your	Yours
Yourself	Yourselves	Zero		

2.1.2.4.*Full-Text Restrictions*

Full-text searches yang didukung untuk tabel MyISAM saja. *Full-text searches* tidak didukung untuk tabel dipartisi. *Full-text searches* dapat digunakan dengan sebagian besar *multi-byte* set karakter. Pengecualian untuk Unicode, karakter set utf8 dapat digunakan, tetapi tidak set karakter UCS2. Namun, meskipun *full-text* indeks pada kolom UCS2 tidak dapat digunakan, dapat melakukan pencarian `IN BOOLEAN MODE` pada kolom UCS2 yang tidak memiliki indeks tersebut. Bahasa ideografik seperti Cina dan Jepang tidak memiliki pembatas kata. Oleh karena itu, parser *full-text* tidak dapat menentukan

mana kata-kata mulai dan berakhir dalam bahasa ideografik.Meskipun penggunaan beberapa set karakter dalam satu tabel didukung, semua kolom dalam indeks *full-text* harus menggunakan set karakter yang sama dan *collation*.`MATCH ()` pada daftar kolom harus sama persis dengan daftar kolom untuk tabel di beberapa definisi indeks *full-text*, kecuali `MATCH ()` di `IN BOOLEAN MODE`. Boolean-mode pencarian dapat dilakukan pada kolom *nonindexed*, meskipun cenderung lambat.`AGAINST ()` harus menjadi nilai string yang konstan selama query evaluation.[5]

2.1.2.5.*Fine-Tuning MySQL Full-Text Search*

Full-textsearch pada MySQL memiliki beberapa *user-tunable parameters*. Dapat melakukan kontrol lebih besar atas *full-text search* jika memiliki distribusi sumber MySQL karena beberapa perubahan membutuhkan modifikasi kode sumber. Perhatikan bahwa *full-text search* digunakan untuk yang paling efektif. Memodifikasi perilaku standar dalam banyak kasus justru dapat menurunkan efektivitas. Jangan mengubah sumber MySQL kecuali tahu apa yang akan dilakukan.

Kebanyakan variabel*full-text* diuraikan dalam bagian yang harus ditetapkan pada waktu server *startup*. Sebuah restart server diperlukan untuk mengubah dan tidak dapat diubah sementara server berjalan.Beberapa perubahan

[5]Stefan Hinz, et al.,"12.9.5 *Full-Text Restrictions*", Aviation Today, diakses dari http://dev.mysql.com/doc/refman/5.5/en/fulltext-restrictions.htmlpada tanggal 3 Desember 2012 pukul 08:47.

variabel mengharuskan membangun kembali indeks *full-text* dalam tabel. Berikut instruksi untuk melakukannya.

1. Panjang minimum dan maksimum dari kata-kata yang akan diindeks didefinisikan oleh ft_min_word_len dan variabel sistem ft_max_word_len. Nilai minimum *default* adalah empat karakter, *default* maksimum adalah tergantung dari versi. Jika mengubah nilai yang lain, harus membangun kembali indeks *FULLTEXT*. Misalnya, jika ingin tiga karakter kata untuk dapat dicari, dapat mengatur variabel ft_min_word_len dengan menempatkan baris berikut dalam file pilihan
2.
```
[mysqld]
ft_min_word_len=3
```

Gambar 2.9*Script*mengatur variable untuk pencarian tiga karakter kata (Stefan Hinz, et al. 2012)

Kemudian *restart* server dan membangun kembali indeks *FULLTEXT*. Perhatikan khususnya komentar mengenai myisamchk dalam daftar petunjuk.

3. Untuk mengganti daftar *stopword default*, atur variabel sistem ft_stopword_file. Nilai variabel harus nama path dari file yang berisi daftar *stopword*, atau string kosong untuk menonaktifkan *stopword*. Server akan mencari file dalam direktori data kecuali nama path absolut yang diberikan untuk menentukan direktori yang berbeda. Setelah mengubah nilai dari variabel atau isi dari file *stopword*, *restart* server dan membangun kembali indeks *FULLTEXT*.

4. Daftar *stopword* adalah bentuk bebas. Artinya, dapat menggunakan karakter apapun *nonalphanumeric* seperti garis baru, spasi, atau komauntuk *stopwords* terpisah. Pengecualian adalah karakter *underscore* ("_") dan tanda kutip tunggal ("' ") sebagai bagian dari sebuah kata. Karakter set daftar *stopword* adalah *default* set karakter server.
5. Ambang batas 50% untuk *natural language searches* ditentukan oleh skema pembobotan tertentu. Untuk menonaktifkannya, cari baris berikut dalam storage/myisam/ftdefs.h.

```
#define GWS_IN_USE GWS_PROB
```

Gambar 2.10*Script*cari baris untukmenonaktifkan skema pembobotan (Stefan Hinz, et al. 2012)

Ubah baris menjadi:

```
#define GWS_IN_USE GWS_FREQ
```

Gambar 2.11*Script* ubah baris untuk menonaktifkan skema pembobotan (Stefan Hinz, et al. 2012)

Kemudian mengkompilasi ulang MySQL.

Catatan : Dengan melakukan perubahan ini, sangat mengurangi kemampuan MySQL untuk memberikan nilai relevansi yang memadai untuk fungsi `MATCH ()`. Jika benar-benar perlu untuk mencari kata umum, akan lebih baik untuk mencari menggunakan `IN BOOLEAN MODE`, yang tidak mengamati ambang batas 50%.

6. Untuk mengubah operator digunakan untuk *boolean full-text searches*, atur variabel sistem ft_boolean_syntax. Variabel ini dapat diubah

sementara server berjalan, tetapi harus memiliki hak istimewa super untuk melakukannya.

7. Jika ingin mengubah set karakter yang dianggap karakter kata, dapat melakukannya dalam beberapa cara, seperti yang dijelaskan dalam daftar berikut. Setelah melakukan modifikasi, harus membangun kembali indeks untuk setiap tabel yang mengandung indeks *FULLTEXT*. Misalkan memperlakukan karakter tanda hubung ('-') sebagai karakter kata. Gunakan salah satu dari metode ini:
    a. Memodifikasi sumber MySQL: Dalam storage/myisam/ftdefs.h, lihat true_word_char () dan misc_word_char () macro. Tambahkan '-' ke salah satu macro dan mengkompilasi ulang MySQL.
    b. Memodifikasi file set karakter: Ini tidak memerlukan kompilasi ulang. The true_word_char () makro menggunakan "tipe karakter" tabel untuk membedakan huruf dan angka dari karakter lain. Dapat mengedit isi dari array <map><ctype> di salah satu karakter mengatur file XML untuk menentukan bahwa '-' adalah “letter”.Kemudian gunakan karakter yang ditetapkan untuk indeks *FULLTEXT*.
    c. Tambahkan *collation*baru untuk set karakter yang digunakan oleh kolom diindeks, dan mengubah kolom untuk menggunakan *collation* itu.

Jika mengubah teks lengkap variabel yang mempengaruhi pengindeksan (ft_min_word_len, ft_max_word_len, atau ft_stopword_file), atau jika mengubah

file *stopword* sendiri, harus membangun kembali indeks fulltext setelah melakukan perubahan dan restart server. Untuk membangun kembali indeks, sudah cukup untuk melakukan operasi perbaikan `QUICK`.

```
mysql> REPAIR TABLE tbl_name QUICK;
```

Gambar 2.12*Script* operasi perbaikan QUICK (Stefan Hinz, et al. 2012)

Atau gunakan `ALTER TABLE` dengan`DROP INDEX` dan `ADD INDEX` pilihan untuk *drop* dan *re-create* setiap indeks *FULLTEXT*. Setiap tabel yang berisi setiap indeks *full-text* harus diperbaiki seperti hanya ditampilkan. Jika tidak, permintaan untuk tabel dapat menghasilkan hasil yang salah, dan modifikasi ke tabel, akan menyebabkan server untuk melihat tabel sebagai korup dan membutuhkan perbaikan.

Catatan bahwa jika menggunakan myisamchk untuk melakukan operasi yang mengubah indeks tabel (seperti perbaikan atau menganalisis), indeks *full-text* dibangun kembali menggunakan *default* full-text dimana nilai parameter untuk panjang kata minimal, panjang kata maksimum, dan file *stopword* kecuali ditentukan sebaliknya. Hal ini dapat mengakibatkan permintaan gagal.Masalah terjadi karena parameter ini hanya diketahui oleh server, tidak disimpan dalam file indeks MyISAM. Untuk menghindari masalah jika telah memodifikasi kata panjang minimum atau maksimum atau nilai *stopword* file yang digunakan oleh server, tentukan ft_min_word_len, ft_max_word_len, dan nilai-nilai ft_stopword_file untuk myisamchk yang digunakan untuk mysqld. Misalnya, jika telah mengatur panjang kata minimal sampai 3, dapat memperbaiki tabel dengan myisamchk seperti ini:

```
shell> myisamchk --recover --ft_min_word_len=3 tbl_name.MYI
```

Gambar 2.13*Script*myisamchk(Stefan Hinz, et al. 2012)

Untuk memastikan bahwa myisamchk dan server menggunakan nilai yang sama untuk *full-text* parameter, tempatkan masing-masing baik di [mysqld] dan [myisamchk] dari file pilihan:

```
[mysqld]
ft_min_word_len=3

[myisamchk]
ft_min_word_len=3
```

Gambar 2.14*Script* penggunaan nilai yang sama untuk full-text parameter (Stefan Hinz, et al. 2012)

Sebuah alternatif untuk menggunakan myisamchk untuk modifikasi indeks adalah dengan menggunakan laporan `REPAIR TABLE`, `ANALYZE TABLE`, `OPTIMIZE TABLE`, atau `ALTER TABLE`. Pernyataan ini dilakukan oleh server, yang tahu nilai parameter yang tepat untuk digunakan.[6]

2.1.2.6.*Adding a Collation for Full-Text Indexing*

Bagian ini menjelaskan cara menambahkan pemeriksaan baru untuk *Full-Text Searches*. Pengumpulan sampel *collation* seperti latin1_swedish_ci tetapi memperlakukan "-" karakter sebagai surat dari pada sebagai karakter tanda baca

[6]Stefan Hinz, et al.,"12.9.6 *Fine-Tuning MySQL Full-Text Search*", Aviation Today, diakses dari http://dev.mysql.com/doc/refman/5.5/en/fulltext-fine-tuning.htmlpada tanggal 3 Desember 2012 pukul 09:10.

sehingga dapat diindeks sebagai karakter kata[7]. Untuk menambahkan pemeriksaan *full-text indexing*, gunakan prosedur ini :

1. Tambahkan*collation*ke file Index.xml . ID *collation* harus terpakai, jadi pilihlah nilai yang berbeda dari 62 jika ID sudah diambil pada sistem.

```
<charset name="latin1">
...
<collation name="latin1_fulltext_ci" id="62"/>
</charset>
```

5. Menyatakan tata urutan *collation* dalam file latin1.xml. Dalam kasus ini, agar dapat disalin dari **latin1_swedish_ci**:

```
<collation name="latin1_fulltext_ci">
<map>
00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F
20 21 22 23 24 25 26 27 28 29 2A 2B 2C 2D 2E 2F
30 31 32 33 34 35 36 37 38 39 3A 3B 3C 3D 3E 3F
40 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 5B 5C 5D 5E 5F
60 41 42 43 44 45 46 47 48 49 4A 4B 4C 4D 4E 4F
50 51 52 53 54 55 56 57 58 59 5A 7B 7C 7D 7E 7F
80 81 82 83 84 85 86 87 88 89 8A 8B 8C 8D 8E 8F
90 91 92 93 94 95 96 97 98 99 9A 9B 9C 9D 9E 9F
A0 A1 A2 A3 A4 A5 A6 A7 A8 A9 AA AB AC AD AE AF
B0 B1 B2 B3 B4 B5 B6 B7 B8 B9 BA BB BC BD BE BF
41 41 41 41 5C 5B 5C 43 45 45 45 45 49 49 49 49
44 4E 4F 4F 4F 4F 5D D7 D8 55 55 55 59 59 DE DF
```

[7]Stefan Hinz, et al.,"12.9.7 *Adding a Collation for Full-Text Indexing*", Aviation Today, diakses dari http://dev.mysql.com/doc/refman/5.5/en/full-text-adding-collation.htmlpada tanggal 3 Desember 2012 pukul 09:17.

22. 41 41 41 41 5C 5B 5C 43 45 45 45 45 49 49 49 49
23. 44 4E 4F 4F 4F 4F 5D F7 D8 55 55 55 59 59 DE FF
24. </map>
    </collation>

25. Memodifikasi array ctype di latin1.xml. Mengubah nilai sesuai dengan 0x2D (yang merupakan kode untuk ‘-‘ karakter) dari 10 (tanda baca) ke 01 (huruf kecil). Dalam array berikut, elemen di baris keempat bawah, nilai ketiga dari akhir.

26. <ctype>
27. <map>
28. 00
29. 20 20 20 20 20 20 20 20 20 28 28 28 28 28 20 20
30. 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20
31. 48 10 10 10 10 10 10 10 10 10 10 10 10 ***01*** 10 10
32. 84 84 84 84 84 84 84 84 84 84 10 10 10 10 10 10
33. 10 81 81 81 81 81 81 01 01 01 01 01 01 01 01 01
34. 01 01 01 01 01 01 01 01 01 01 01 01 10 10 10 10 10
35. 10 82 82 82 82 82 82 02 02 02 02 02 02 02 02 02
36. 02 02 02 02 02 02 02 02 02 02 02 10 10 10 10 20
37. 10 00 10 02 10 10 10 10 10 10 01 10 01 00 01 00
38. 00 10 10 10 10 10 10 10 10 10 02 10 02 00 02 01
39. 48 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
40. 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
41. 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01 01
42. 01 01 01 01 01 01 01 10 01 01 01 01 01 01 01 02
43. 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02 02
44. 02 02 02 02 02 02 02 10 02 02 02 02 02 02 02 02
45. </map>
46. </ctype>

47. Restart server.

48. Untuk menggunakan *collation* baru, termasuk dalam definisi kolom yang menggunakannya :

```
49. mysql>DROP TABLE IF EXISTS t1;
50. Query OK, 0 rows affected (0.13 sec)
51.
52. mysql>CREATE TABLE t1 (
53.     ->a TEXT CHARACTER SET latin1 COLLATE latin1_fulltext_ci,
54.     ->FULLTEXT INDEX(a)
55.     ->) ENGINE=MyISAM;
56. Query OK, 0 rows affected (0.47 sec)
```

57. Uji pemeriksaan untuk memverifikasi tanda hubung yang dianggap sebagai kata karakter:

```
58. mysql>INSERT INTO t1 VALUEs ('----'),('....'),('abcd');
59. Query OK, 3 rows affected (0.22 sec)
60. Records: 3  Duplicates: 0  Warnings: 0
61.
62. mysql>SELECT * FROM t1 WHERE MATCH a AGAINST ('----' IN
    BOOLEAN MODE);
63. +------+
64. | a    |
65. +------+
66. | ---- |
67. +------+
    1    row in set (0.00 sec)
```

2.1.3. Karya Akhir

Karya Akhir adalah suatu karya tulis ilmiah, berupapaparan tulisan hasil observasi, praktik kerja, atau penelitian terapan sederhana yang:

1. Mendeskripsikan atau membahas suatu proses dandinamika kegiatan kerja
2. Membahas suatu masalah dalam bidang terapan ilmu tertentu dengan menggunakan kaidah-kaidah yang berlaku dalam bidang terapan ilmu tersebut.[8]

2.1.4. Abstrak

Abstrak adalah bagian ringkas suatu uraian yang merupakangagasan utama dari suatu pembahasan yang akan diuraikan.[9]Abstrak karya akhir terdiri dari seratus lima puluh sampai dengan dua ratus kata, dan disajikan pada halaman tersendiri sebelum bab I. Abstak penelitian sekurang-kurangnya berisi hal-hal berikut.

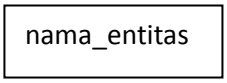
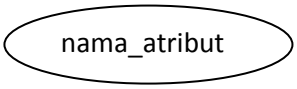
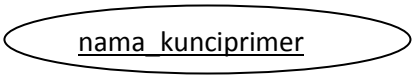
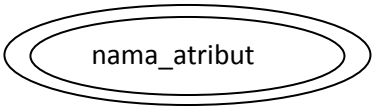
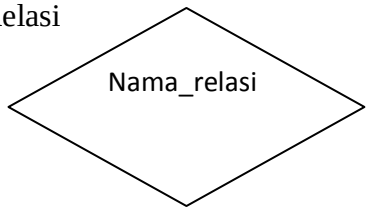
1. Tujuan penelitian
2. Metode penelitian
3. Pembahasan
4. Hasil penelitian atau temuan
5. Kontibusi penelitian
6. Kata kunci (Keywords)sedikitnya tiga buah.

[8]Buku Pedoman Karya Akhir Tahun Akademik 2011/2012, hlm.1.
[9]Hariyanto,"Pengertian abstrak dan cara membuat abstrak",Aviation Today, diakses dari http://belajarpsikologi.com/abstrak-contoh-abstrak-penelitian/ pada tanggal 25 Januari 2013 pukul 10:51

2.1.5. *Entity Relational Diagram* (ERD)

ERD dikembangkan berdasarkan teori himpunan dalam bidang matematika. ERD digunakan untuk pemodelan basis data relational.[10] Sehingga jika penyimpanan basis data menggunakan OODBMS maka perancangan basis data tidak perlu menggunakan ERD. Tabel 2.2 menunjukkan simbol-simbol yang digunakan pada ERD.

Tabel 2.2 Simbol-simbol ERD.

Simbol	Deskripsi
Entitas /*entity* 	Entitas merupakan data inti yang akan disimpan, bakal tabel pada basis data
Atribut 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas
Atribut kunci primer 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas dan digunakan sebagai kunci akses record yang diinginkan, biasanya berupa id
Atribut multinilai/ *multivalue* 	*Field* atau kolom data yang butuh disimpan dalam suatu entitas yang dapat memiliki nilai lebih dari satu
Relasi 	Relasi yang menghubungkan antar entitas, biasanya diawali dengan kata kerja
Asosiasi / *association*	Penghubung antara relasi dan entitas di mana di kedua ujungnya memiliki

[10]Rosa A.S-M. *Shalahuddin, Modul Pembelajaran Rekayasa perangkat Lunak*, Modula, Bandung, 2011, hlm.49.

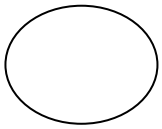
1 0..*	*multiplicity* kemungkinan jumlah pemakaian

2.1.6. *Data Flow Diagram* (DFD)

Data Flow Diagram adalah representasi grafik yang menggambarkan aliran informasi dan transformasi informasi yang diaplikasikan sebagai data yang mengalir dari masukan (*input*) dan keluaran (*output*).[11] DFD dapat digunakan untuk mempresentasikan sebuah sistem atau perangkat lunak pada beberapa level abstraksi. DFD dapat dibagi menjadi beberapa level yang lebih detail untuk mempresentasikan aliran informasi atau fungsi yang lebih detail. DFD menyediakan mekanisme untuk pemodelan fungsional ataupun pemodelan aliran informasi. Oleh karena itu, DFD lebih sesuai digunakan untuk memodelkan fungsi-fungsi perangkat lunak yang akan diimplementasikan menggunakan pemrograman terstruktur karena pemrograman terstruktur membagi-bagi bagiannya dengan fungsi-fungsi dan prosedur-prosedur.

Notasi – notasi pada DFD (Edward Yourdon dan Tom DeMarco) adalah sebagai berikut :

Tabel 2.3 Notasi-notasi DFD.

Notasi	Keterangan
	Proses atau fungsi atau prosedur, pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur, maka

[11]Rosa A.S-M. *Shalahuddin, Modul Pembelajaran Rekayasa perangkat Lunak*, Modula, Bandung, 2011, hlm.63.

	pemodelan notasi inilah yang harusnya menjadi fungsi atau prosedur di dalam kode program Catatan : nama yang diberikan pada sebuah proses biasanya berupa kata kerja
_____ _____	*File* atau basis data atau penyimpanan (*storage*), pada pemodelan perangkat lunak yang akan diimplementasikan dengan pemrograman terstruktur, maka pemodelan notasi inilah yang harusnya dibuat menjadi tabel-tabel basis data yang dibutuhkan, tabel-tabel ini juga harus sesuai dengan perancangan tabel-tabel pada basis data (*Entity Relationship Diagram* (ERD), (*Conceptual Data Model* (CDM), (*Physical Data Model* (PDM) Catatan : nama yang diberikan pada sebuah penyimpanan biasanya kata benda
	Entitas luar (*external entity*) atau masukan (*input*) atau keluaran (*output*) atau orang yang memakai / berinteraksi dengan perangkat lunak yang dimodelkan atau sistem lain yang terkait dengan aliran data dari sistem yang dimodelkan Catatan: nama yang digunakan pada masukan (*input*) dan keluaran (*output*)

	biasanya berupa kata benda
	Aliran data, merupakan data yang dikirim antar proses, dari penyimpanan ke proses atau dari proses ke masukan (input) atau keluaran (output) Catatan : nama yang digunakan pada aliran data biasanya berupa kata benda, dapat diawali dengan kata data misalnya "data siswa" atau tanpa kata data misalnya "siswa"

Berikut ini adalah tahapan-tahapan perancangan dengan menggunakan DFD:

1. Level 0 (Diagram Konteks)

DFD level 0 digunakan untuk menggambarkan interaksi antara sistem yang akan dikembangkan dengan entitas luar.

2. Level 1 (Diagram Nol)

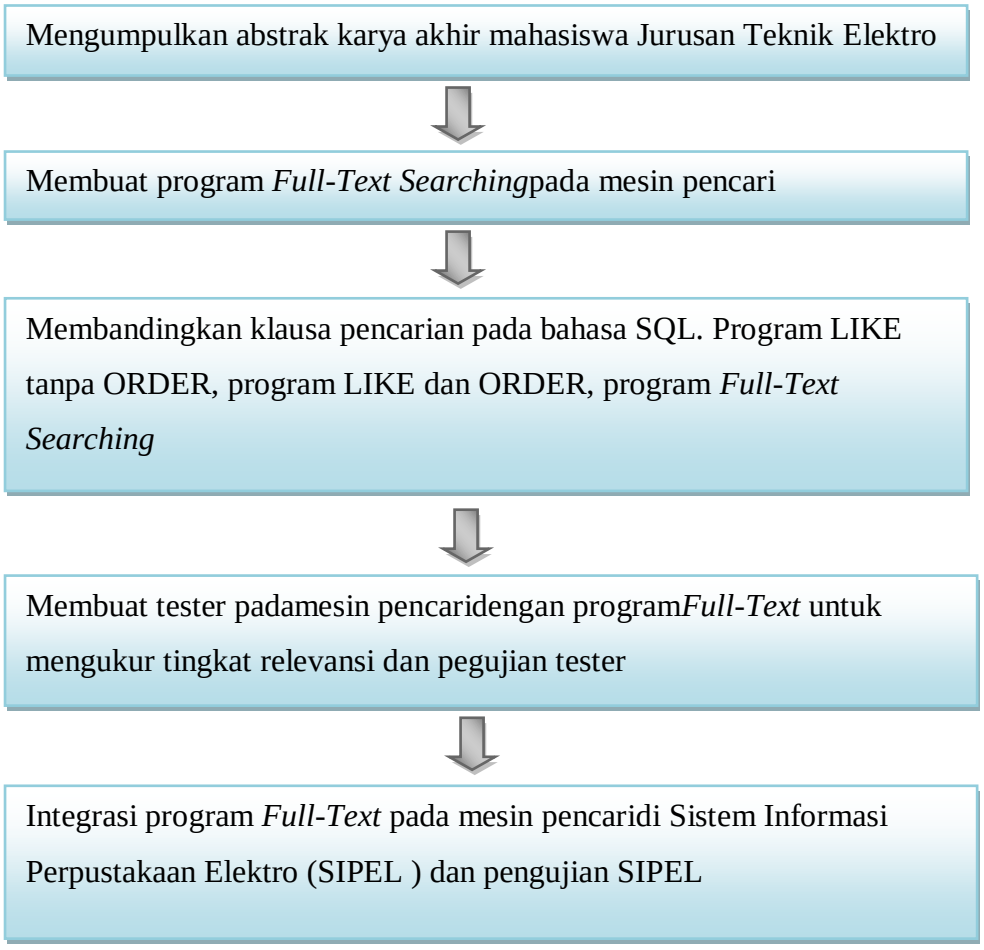
DFD level 1 digunakan untuk menggambarkan modul-modul yang ada dalam sistem yang akan dikembangkan.

3. Level 2 (Diagram Rinci)

DFD level 2 merincikan DFD level 1. Untuk sebuah sistem, jumlah DFD level 2 sama dengan jumlah modul pada DFD level 1 yang di *breakdown*.

2.2. Kerangka Berfikir

Kerangka berpikir dalam pengembangan sistem pencarian karya akhir berdasarkan abstrak menggunakan *full-text searching*di sistem informasi perpustakaan Jurusan Teknik Elektro Universitas Negeri Jakarta di tunjukkan pada gambar 2.19.



Gambar 2.19 Kerangka Berfikir