Full text search with Sphinx & MySQL

Shlomi Noach
openark.org

MySQL Users Group Meeting
Israel, July 7th. 2009

FULLTEXT search in MySQL

- Built in full text search supported in MyISAM tables.
- Search provided by the special FULLTEXT index.

MyISAM FULLTEXT Example

```
CREATE TABLE `City` (
  `ID` int(11) NOT NULL AUTO_INCREMENT,
  `Name` char(35) NOT NULL DEFAULT '',
  `CountryCode` char(3) NOT NULL DEFAULT '',
  `District` char(20) NOT NULL DEFAULT '',
  `Population` int(11) NOT NULL DEFAULT '0',
  PRIMARY KEY (`ID`),
  FULLTEXT KEY `Name` (`Name`,`District`)
) ENGINE=MyISAM;
```

MyISAM FULLTEXT Example

```
mysql> SELECT * FROM City WHERE MATCH(Name, District) AGAINST ('aviv');
+------+----------------+-------------+----------+------------+
| ID   | Name           | CountryCode | District | Population |
+------+----------------+-------------+----------+------------+
| 1451 | Tel Aviv-Jaffa | ISR         | Tel Aviv |     348100 |
| 1459 | Bat Yam        | ISR         | Tel Aviv |     137000 |
| 1455 | Holon          | ISR         | Tel Aviv |     163100 |
| 1461 | Ramat Gan      | ISR         | Tel Aviv |     126900 |
| 1460 | Bene Beraq     | ISR         | Tel Aviv |     133900 |
+------+----------------+-------------+----------+------------+
5 rows in set (0.00 sec)
```

MyISAM FULLTEXT pros

- Supports BOOLEAN MODE
- Supports stop words
- Supports plugin API (5.1)
- Text indexed on line

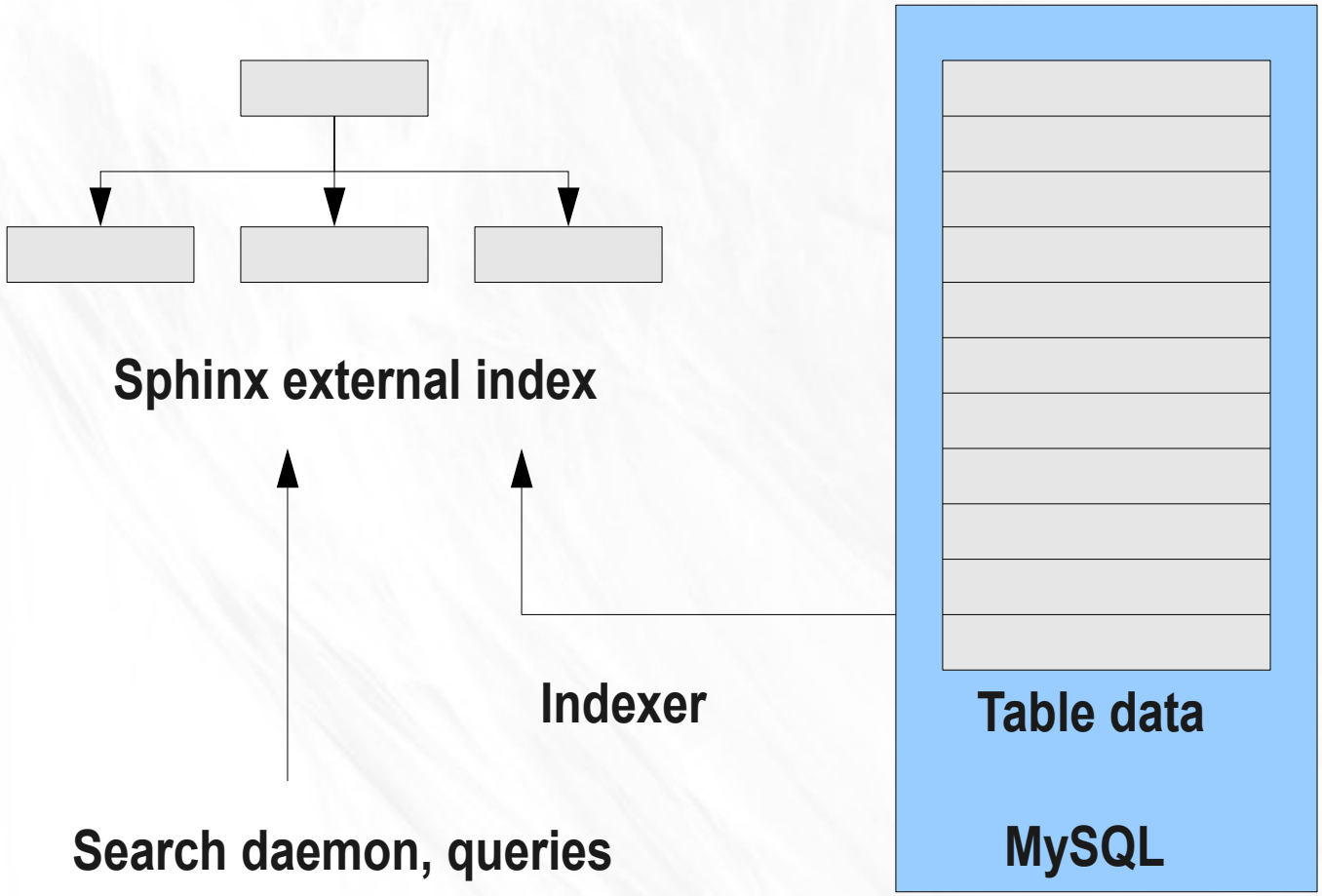
MyISAM FULLTEXT cons

- Does not scale well
- Does not shard
- Restricted to MyISAM

Sphinx Search Engine

- Developed since 2001 by Andrew Aksyonoff.
- Has gained recognition in the past few years.
- Today, competes with Apache's Lucene.
- Is in use by heavyweight MySQL based applications.

Sphinx & MySQL layout



Sphinx layout

- Indexer: creates full text index(es).
- Keeps a static index (can be re-indexed, fully, incrementally, merged)
- Searchd: a standalone server, provides resultf for queries.

Sphinx facts

- Not a MySQL extension.
- Designed to work with RDBMS, can read input with SQL queries.
- Can read custom XML format.
- Can index any storage engine (not InnoDB specific)

Sphinx facts

- Does not keep actual text.
- Can keep numeric data.
- Provides SQL-like GROUP-BY.
- Provides SQL-like ORDER-BY.
- Comes with many APIs for common programming languages.
- Comes with MySQL API in the form of SphinxSE.

Example

```
CREATE TABLE `film` (
  `film_id` smallint(5) unsigned NOT NULL
auto_increment,
  `title` varchar(255) NOT NULL,
  `description` text,
  `last_update` timestamp NOT NULL default
CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP,
  ...
  PRIMARY KEY  (`film_id`),
  ...
) ENGINE=InnoDB ;
```

Example: sphinx.conf

```
source film
{
  type         = mysql

  sql_host       = localhost
  sql_user       = sakila_ro
  sql_pass       = 123456
  sql_db         = sakila
  sql_port       = 3306 # optional, default is 3306

  sql_query        = \
    SELECT film_id, title, UNIX_TIMESTAMP(last_update) AS
last_update_timestamp FROM film

  sql_attr_int     = film_id
  sql_attr_timestamp   = last_update_timestamp

  sql_query_info     = SELECT * FROM film WHERE film_id=$id
}
```

Example: sphinx.conf

```
index film
{
  source          = film
  path            = /usr/bin/sphinx/data/film
}
```

Example: Java API

```
SphinxClient cl = new SphinxClient();
cl.SetServer( host, port );
cl.SetMatchMode( SphinxClient.SPH_MATCH_ALL );
cl.SetLimits ( offset, limit );
cl.SetSortMode ( SphinxClient.SPH_SORT_RELEVANCE,
sortClause );
cl.SetGroupBy ( groupBy, SphinxClient.SPH_GROUPBY_ATTR,
groupSort );

SphinxResult res = cl.Query(q.toString(), index);
```

Example: SphinxSE

```
CREATE TABLE sphinx_film
(
    film_id INT NOT NULL,
    weight  INT NOT NULL,
    query VARCHAR(3072) NOT NULL,
    last_update INT,
    INDEX(query)
) ENGINE=SPHINX
CONNECTION="sphinx://localhost:12321/film";
```

Example: SphinxSE queries

```
SELECT * FROM sphinx_film WHERE query='drama';

SELECT * FROM sphinx_film INNER JOIN file USING
(film_id) WHERE query='drama';

SELECT * FROM sphinx_film INNER JOIN file USING
(film_id) WHERE query='drama;limit=50';

SELECT * FROM sphinx_film INNER JOIN file USING
(film_id) WHERE
query='drama;limit=50;sort=attr_asc:last_update';

SELECT * FROM sphinx_film INNER JOIN file USING
(film_id) WHERE
query='drama;limit=50;groupby=day:last_update';
```

SphinxSE

- With SphinxSE, Sphinx can be queried via MySQL.
- Usually Sphinx results need to be matched with table data, at any case
- Can be found in OurDelta builds, or can be built manually.

Sphinx queries

- Sphinx provides ORDER-BY, GROUP-BY and filtering functionality, built in.
- Based on, and limited to, common functionality.
- This means no need for filesort once search results retrieved.
- Sphinx can actually outperform MySQL on these operations.

Excerpts

- Sphinx does not store actual text data.
- But can be accessed with data and search phrase, and asked for excerpts

```
docs = ['this is my test text to be highlighted','this is
another test text to be highlighted']
words = 'test text'
index = 'test1'
opts = {'before_match':'<b>', 'after_match':'</b>',
'chunk_separator':' ... ', 'limit':400, 'around':15}

cl = SphinxClient()
res = cl.BuildExcerpts(docs, index, words, opts)
```

Sphinx proxy

- Sphinx supports aggregation of multiple indexes, as well as aggregation of multiple server results.
- One server can act as proxy to other servers.
- It automatically calls upon, and awaits results from remote servers; merging results.
- Good solution for sharded databases.

Many more features

- Incremental index builds.
- Index merges.
- Prefix, suffix, infix.
- Morphology support.

Thank you!

Hope to see you in the next MySQL Users Group meeting!