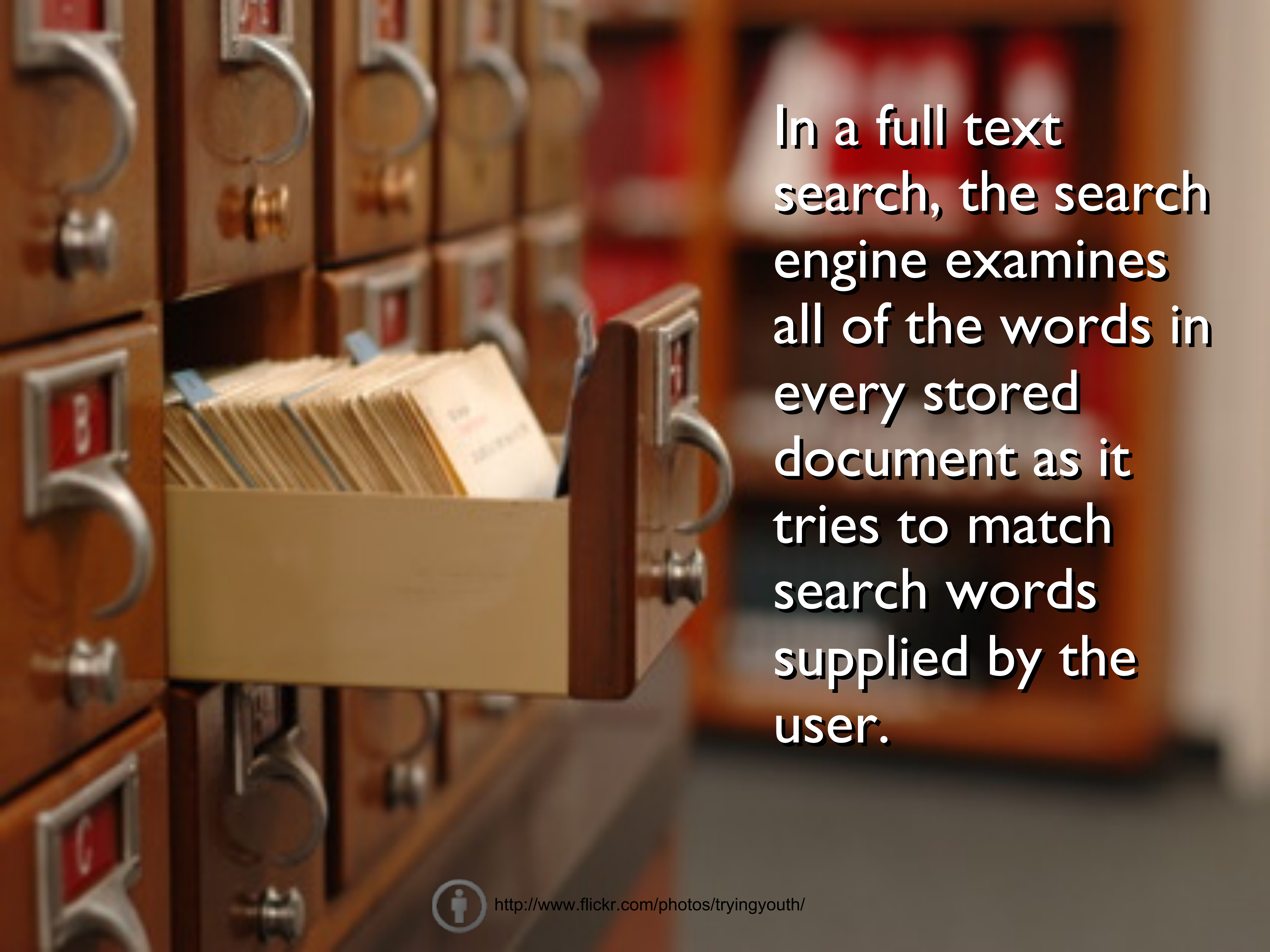
Full Text Search Throwdown

Bill Karwin, Percona Inc.

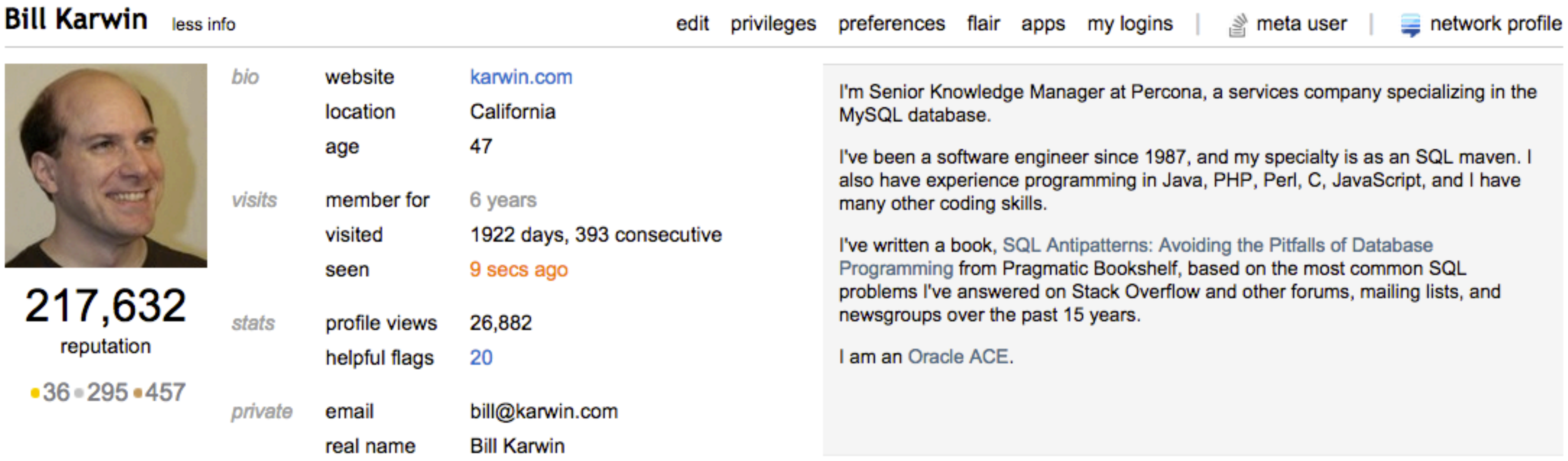


In a full text search, the search engine examines all of the words in every stored document as it tries to match search words supplied by the user.

http://www.flickr.com/photos/tryingyouth/

StackOverflow Test Data

- Latest data dump, exported 2014
- 8 million Posts = 8.9 GB



Bill Karwin less info

edit privileges preferences flair apps my logins | meta user | network profile

217,632
reputation
●36 ●295 ●457

bio	website	karwin.com
	location	California
	age	47
visits	member for	6 years
	visited	1922 days, 393 consecutive
	seen	9 secs ago
stats	profile views	26,882
	helpful flags	20
private	email	bill@karwin.com
	real name	Bill Karwin

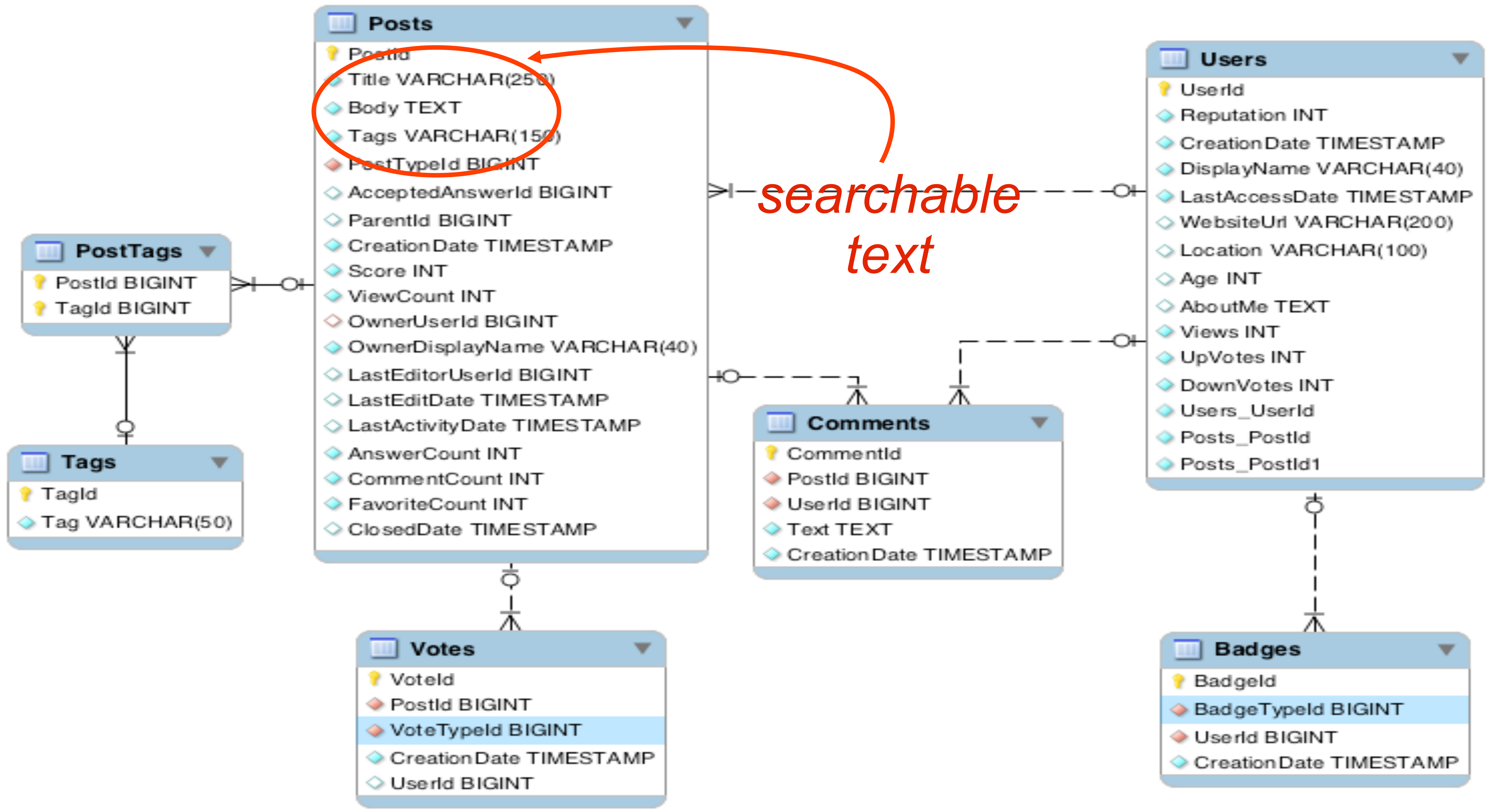
I'm Senior Knowledge Manager at Percona, a services company specializing in the MySQL database.

I've been a software engineer since 1987, and my specialty is as an SQL maven. I also have experience programming in Java, PHP, Perl, C, JavaScript, and I have many other coding skills.

I've written a book, SQL Antipatterns: Avoiding the Pitfalls of Database Programming from Pragmatic Bookshelf, based on the most common SQL problems I've answered on Stack Overflow and other forums, mailing lists, and newsgroups over the past 15 years.

I am an Oracle ACE.

StackOverflow ER diagram



The Baseline:
Naive Search Predicates

Some people, when confronted with a problem, think

“I know, I’ll use regular expressions.”

Now they have two problems.

— Jamie Zawinsky

Accuracy issue

- Irrelevant or false matching words ‘one’, ‘money’, ‘prone’, etc.:

```
 SELECT * FROM Posts
WHERE Body LIKE '%one%'
```

- Regular expressions in MySQL support escapes for word boundaries:

```
 SELECT * FROM Posts
WHERE Body RLIKE '[[:<:]]one[[:>:]]'
```

Performance issue

- LIKE with wildcards:

```
SELECT * FROM Posts
WHERE title LIKE '%performance%'
 OR body LIKE '%performance%'
 OR tags LIKE '%performance%';
```

97 sec

- POSIX regular expressions:

```
SELECT * FROM Posts
WHERE title RLIKE '[[:<:]]performance[[:>:]]'
 OR body RLIKE '[[:<:]]performance[[:>:]]'
 OR tags RLIKE '[[:<:]]performance[[:>:]]';
```

655 sec

Why so slow?

```
CREATE TABLE TelephoneBook (
   FullName VARCHAR(50));

CREATE INDEX name_idx ON TelephoneBook
   (FullName);

INSERT INTO TelephoneBook VALUES
   ('Riddle, Thomas'),
   ('Thomas, Dean');
```

Why so slow?

- Search for all with last name "Thomas"

```
SELECT * FROM telephone_book
WHERE full_name LIKE 'Thomas%'
```

uses index

- Search for all with first name "Thomas"

```
SELECT * FROM telephone_book
WHERE full_name LIKE '%Thomas'
```

can't use index

Because:

☞ *B-Tree indexes can’t search for substrings*

FULLTEXT in MyISAM
FULLTEXT in InnoDB
Apache Solr
Sphinx Search
Trigraphs

FULLTEXT in MyISAM

FULLTEXT Index with MyISAM

- Special index type for MyISAM
- Integrated with SQL queries
- Indexes always in sync with data
- Balances features vs. speed vs. space
- Testing: MySQL Community Edition 5.7.5

Build Index on Data (MyISAM)

```
mysql> CREATE FULLTEXT INDEX PostText
   ON Posts(title, body, tags);
```

time: 30 min, 10 sec

Size of Index (MyISAM)

```
mysql>SHOW TABLE STATUS LIKE 'Posts'\G
Name: posts
Engine: MyISAM
Rows: 8000000
Avg_row_length: 927
Data_length: 7417899480 (6.91GB)
Index_length: 2803019776 (Δ = 2.50GB)
```

Querying

```
SELECT * FROM Posts
  WHERE MATCH(column(s))
  AGAINST('query pattern');
```

must include all columns of your index, in the order you defined

Natural Language Mode (MyISAM)

- Searches concepts with free text queries:

```
SELECT * FROM Posts
WHERE MATCH(title, body, tags )
AGAINST('mysql performance'
IN NATURAL LANGUAGE MODE)
LIMIT 100;
```

time with index:
183 milliseconds

Query Profile: Natural Language Mode (MyISAM)

```
+--------------------------+-----------+
| Status                   | Duration  |
+--------------------------+-----------+
| starting                 | 0.000112  |
| checking permissions     | 0.000012  |
| Opening tables           | 0.000155  |
| checking permissions     | 0.000004  |
| checking permissions     | 0.000032  |
| init                     | 0.000007  |
| checking permissions     | 0.000070  |
| System lock              | 0.000012  |
| optimizing               | 0.000009  |
| statistics               | 0.000028  |
| preparing                | 0.000009  |
| FULLTEXT initialization  | 0.180492  |
| executing                | 0.000012  |
| Sending data             | 0.002302  |
| end                      | 0.000004  |
| query end                | 0.000005  |
| closing tables           | 0.000017  |
| freeing items            | 0.000432  |
| cleaning up              | 0.000090  |
+--------------------------+-----------+
```

Boolean Mode (MyISAM)

- Searches words using mini-language:

```
SELECT * FROM Posts
WHERE MATCH(title, body, tags)
AGAINST('+mysql +performance'
IN BOOLEAN MODE)
LIMIT 100;
```

time with index:
10 milliseconds

Query Profile: Boolean Mode (MyISAM)

```
+--------------------------+----------+
| Status                   | Duration |
+--------------------------+----------+
| starting                 | 0.000123 |
| checking permissions     | 0.000009 |
| Opening tables           | 0.000153 |
| checking permissions     | 0.000005 |
| checking permissions     | 0.000033 |
| init                     | 0.000007 |
| checking permissions     | 0.000070 |
| System lock              | 0.000012 |
| optimizing               | 0.000009 |
| statistics               | 0.000027 |
| preparing                | 0.000009 |
| FULLTEXT initialization  | 0.000025 |
| executing                | 0.000008 |
| Sending data             | 0.009507 |
| end                      | 0.000009 |
| query end                | 0.000008 |
| closing tables           | 0.000017 |
| freeing items            | 0.000107 |
| cleaning up              | 0.000024 |
+--------------------------+----------+
```

FULLTEXT in InnoDB

FULLTEXT Index with InnoDB

- Usage very similar to FULLTEXT in MyISAM
- Integrated with SQL queries
- Indexes always* in sync with data
- Read the blogs for more details:
  - http://blogs.innodb.com/wp/2011/07/overview-and-getting-started-with-innodb-fts/
  - http://blogs.innodb.com/wp/2011/07/innodb-full-text-search-tutorial/
  - http://blogs.innodb.com/wp/2011/07/innodb-fts-performance/
  - http://blogs.innodb.com/wp/2011/07/difference-between-innodb-fts-and-myisam-fts/
- Testing: MySQL Community Edition 5.7.5

Build Index on Data (InnoDB)

- Relatively new code; you might see problems:

```
mysql> CREATE FULLTEXT INDEX PostText
   ON Posts(title, body, tags);
ERROR 2013 (HY000): Lost connection to
  MySQL server during query
```

Build Index on Data (InnoDB)

- Solution: define a primary key column called `` `FTS_DOC_ID` `` explicitly:

```
mysql> ALTER TABLE Posts
  CHANGE COLUMN PostId
  `FTS_DOC_ID` BIGINT UNSIGNED;

mysql> CREATE FULLTEXT INDEX PostText
   ON Posts(title, body, tags);
```

time: 30 min, 19 sec

Size of Index (InnoDB)

```
mysql>SHOW TABLE STATUS LIKE 'Posts'\G
Name: posts
Engine: InnoDB
Rows: 6877702
Avg_row_length: 1427
Data_length: 9817817088 (9.14GB)
Index_length: 0 *
```

2.62 GB on disk

Natural Language Mode (InnoDB)

- Searches concepts with free text queries:

```
SELECT * FROM Posts
WHERE MATCH(title, body, tags)
AGAINST('mysql performance'
IN NATURAL LANGUAGE MODE)
LIMIT 100;
```

time with index:
610 milliseconds

Query Profile: Natural Language Mode (InnoDB)

```
+--------------------------+----------+
| Status                   | Duration |
+--------------------------+----------+
| starting                 | 0.000185 |
| checking permissions     | 0.000009 |
| Opening tables           | 0.000154 |
| checking permissions     | 0.000004 |
| checking permissions     | 0.000033 |
| init                     | 0.000010 |
| checking permissions     | 0.000072 |
| System lock              | 0.000010 |
| optimizing               | 0.000011 |
| statistics               | 0.000027 |
| preparing                | 0.000010 |
| FULLTEXT initialization  | 0.404054 |
| executing                | 0.000013 |
| Sending data             | 0.143711 |
| end                      | 0.000014 |
| query end                | 0.000011 |
| closing tables           | 0.000011 |
| freeing items            | 0.063529 |
| cleaning up              | 0.000059 |
+--------------------------+----------+
```

Boolean Mode (InnoDB)

- Searches words using mini-language:

```
SELECT * FROM Posts
WHERE MATCH(title, body, tags)
AGAINST('+mysql +performance'
IN BOOLEAN MODE)
LIMIT 100;
```

time with index:
323 milliseconds

Query Profile: Boolean Mode (InnoDB)

```
+--------------------------+----------+
| Status                   | Duration |
+--------------------------+----------+
| starting                 | 0.000128 |
| checking permissions     | 0.000008 |
| Opening tables           | 0.000155 |
| checking permissions     | 0.000004 |
| checking permissions     | 0.000033 |
| init                     | 0.000007 |
| checking permissions     | 0.000072 |
| System lock              | 0.000010 |
| optimizing               | 0.000009 |
| statistics               | 0.000027 |
| preparing                | 0.000010 |
| FULLTEXT initialization  | 0.313276 |
| executing                | 0.000010 |
| Sending data             | 0.008098 |
| end                      | 0.000008 |
| query end                | 0.000011 |
| closing tables           | 0.000012 |
| freeing items            | 0.001478 |
| cleaning up              | 0.000024 |
+--------------------------+----------+
```

Apache Solr

Apache Solr

- http://lucene.apache.org/solr/
- Formerly known as Lucene, started 2001
- Apache License
- Java implementation
- Web service architecture
- Many sophisticated search feature
- Testing: Apache Solr 4.10.1, Java 8

DataImportHandler

- *conf/solrconfig.xml:*

```
. . .
<requestHandler name="/dataimport"
   class="solr.DataImportHandler">
  <lst name="defaults">
    <str name="config">solr-data-config.xml</str>
  </lst>
</requestHandler>
. . .
```

DataImportHandler

- *conf/data-config.xml:*

```
<dataConfig>
  <dataSource type="JdbcDataSource"
              driver="com.mysql.jdbc.Driver"
              url="jdbc:mysql://localhost/testpattern?useUnicode=true"
              batchSize="-1"
              user="xxxx"
              password="xxxx"/>
  <document>
    <entity name="id"
            query="SELECT PostId, ParentId, Title, Body, Tags FROM Posts">
    </entity>
  </document>
</dataConfig>
```

extremely important to avoid buffering the whole query result!

DataImportHandler

- *conf/schema.xml:*

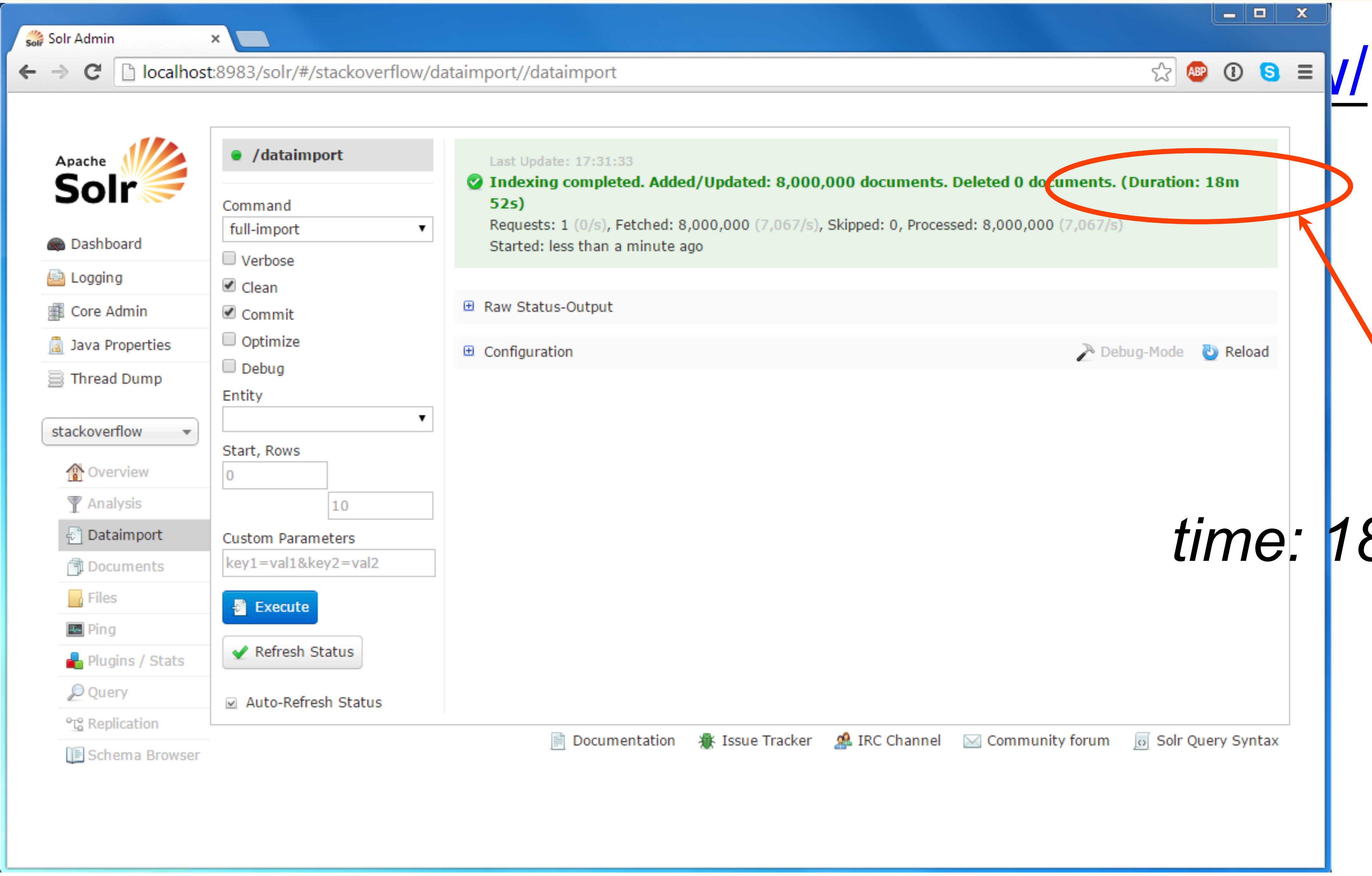
```
. . .
<fields>
   <field name="Id" type="string" indexed="true" stored="true" required="true" />
   <field name="ParentId" type="string" indexed="true" stored="true" required="false" />
   <field name="Title" type="text_general" indexed="false" stored="false"
    required="false" />
   <field name="Body" type="text_general" indexed="false" stored="false" required="false"
    >
   <field name="Tags" type="text_general" indexed="false" stored="false" required="false"
    >

   <field name="text" type="text_general" indexed="true" stored="false" multiValued="true"
    >
<fields>

<uniqueKey>PostId</uniqueKey>
<defaultSearchField>text</defaultSearchField>

<copyField source="Title" dest="text"/>
<copyField source="Body" dest="text"/>
<copyField source="Tags" dest="text"/>
. . .
```

Insert Data into Index (Solr)



```
localhost:8983/solr/#/stackoverflow/dataimport//dataimport
```

Last Update: 17:31:33

Indexing completed. Added/Updated: 8,000,000 documents. Deleted 0 documents. (Duration: 18m 52s)

Requests: 1 (0/s), Fetched: 8,000,000 (7,067/s), Skipped: 0, Processed: 8,000,000 (7,067/s)

Started: less than a minute ago

Command: full-import

- [ ] Verbose
- [x] Clean
- [x] Commit
- [ ] Optimize
- [ ] Debug

Entity:

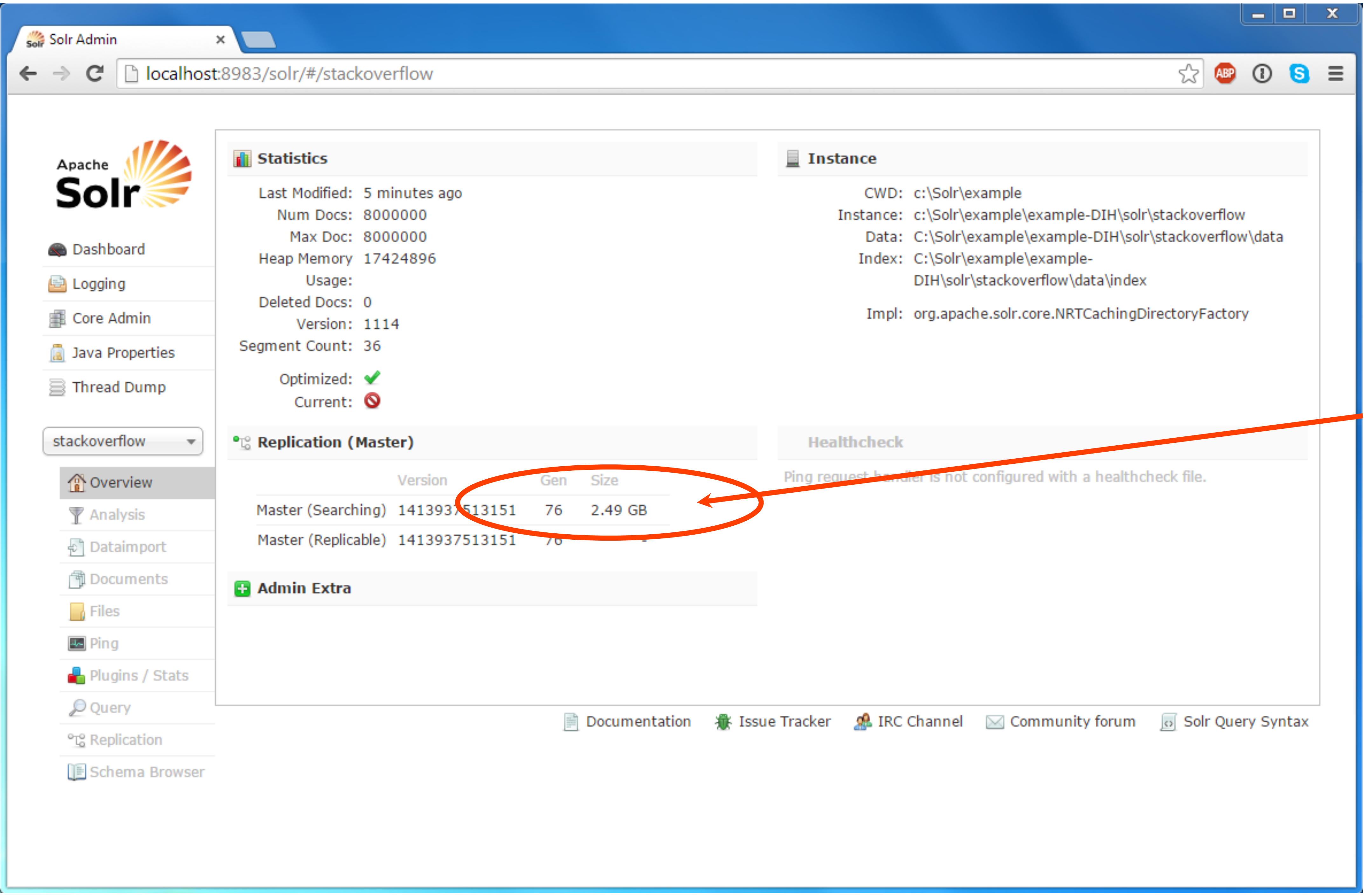
Start, Rows: 0, 10

Custom Parameters: key1=val1&key2=val2

- [x] Auto-Refresh Status

time: 18 min 52 sec

Size of Index (Solr)



Statistics

Last Modified:	5 minutes ago
Num Docs:	8000000
Max Doc:	8000000
Heap Memory Usage:	17424896
Deleted Docs:	0
Version:	1114
Segment Count:	36
Optimized:	✓
Current:	⊘

Instance

CWD:	c:\Solr\example
Instance:	c:\Solr\example\example-DIH\solr\stackoverflow
Data:	C:\Solr\example\example-DIH\solr\stackoverflow\data
Index:	C:\Solr\example\example-DIH\solr\stackoverflow\data\index
Impl:	org.apache.solr.core.NRTCachingDirectoryFactory

Replication (Master)

	Version	Gen	Size
Master (Searching)	1413937513151	76	2.49 GB
Master (Replicable)	1413937513151	76	-

Healthcheck

Ping request handler is not configured with a healthcheck file.

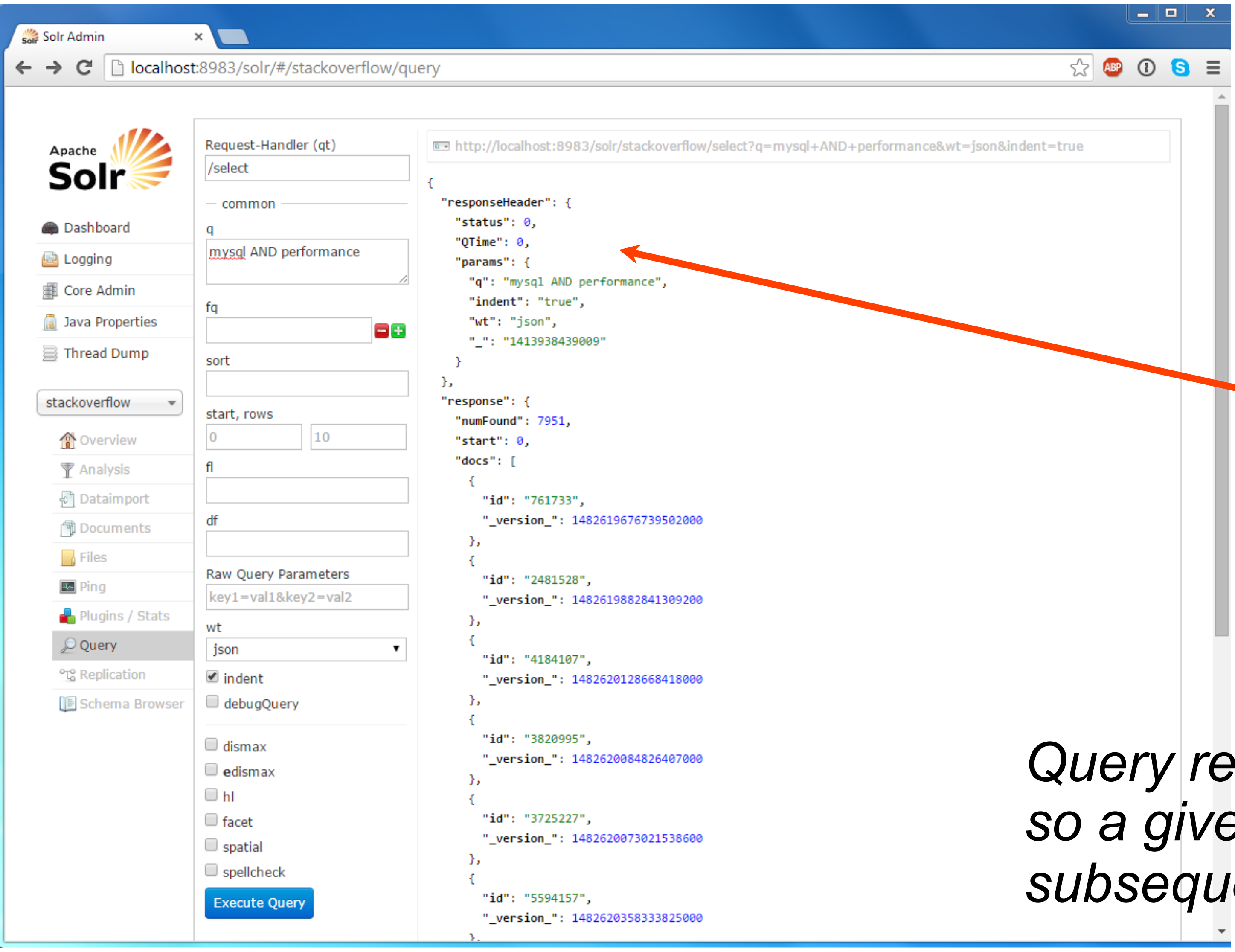
Admin Extra

Documentation Issue Tracker IRC Channel Community forum Solr Query Syntax

size: 2.49 G

Searching Solr

- http://localhost:8983/solr/stackoverflow/query?q=mysql+AND+performance



Request-Handler (qt): /select
common
q: mysql AND performance
fq:
sort:
start, rows: 0, 10
fl:
df:
Raw Query Parameters: key1=val1&key2=val2
wt: json
[✓] indent
[ ] debugQuery
[ ] dismax
[ ] edismax
[ ] hl
[ ] facet
[ ] spatial
[ ] spellcheck

http://localhost:8983/solr/stackoverflow/select?q=mysql+AND+performance&wt=json&indent=true

```
{
  "responseHeader": {
    "status": 0,
    "QTime": 0,
    "params": {
      "q": "mysql AND performance",
      "indent": "true",
      "wt": "json",
      "_": "1413938439009"
    }
  },
  "response": {
    "numFound": 7951,
    "start": 0,
    "docs": [
      {
        "id": "761733",
        "_version_": 1482619676739502000
      },
      {
        "id": "2481528",
        "_version_": 1482619882841309200
      },
      {
        "id": "4184107",
        "_version_": 1482620128668418000
      },
      {
        "id": "3820995",
        "_version_": 1482620084826407000
      },
      {
        "id": "3725227",
        "_version_": 1482620073021538600
      },
      {
        "id": "5594157",
        "_version_": 1482620358333825000
      },
```

time: 60-1700ms

Query results are cached (like MySQL Query Cache), so a given search returns much faster (< 1ms) on subsequent execution

Sphinx Search

Sphinx Search

- http://sphinxsearch.com/
- Started in 2001
- GPLv2 license
- C++ implementation
- SphinxSE storage engine for MySQL
- Supports MySQL protocol, SQL-like queries
- Many sophisticated search features
- Testing: Sphinx Search 2.2.5

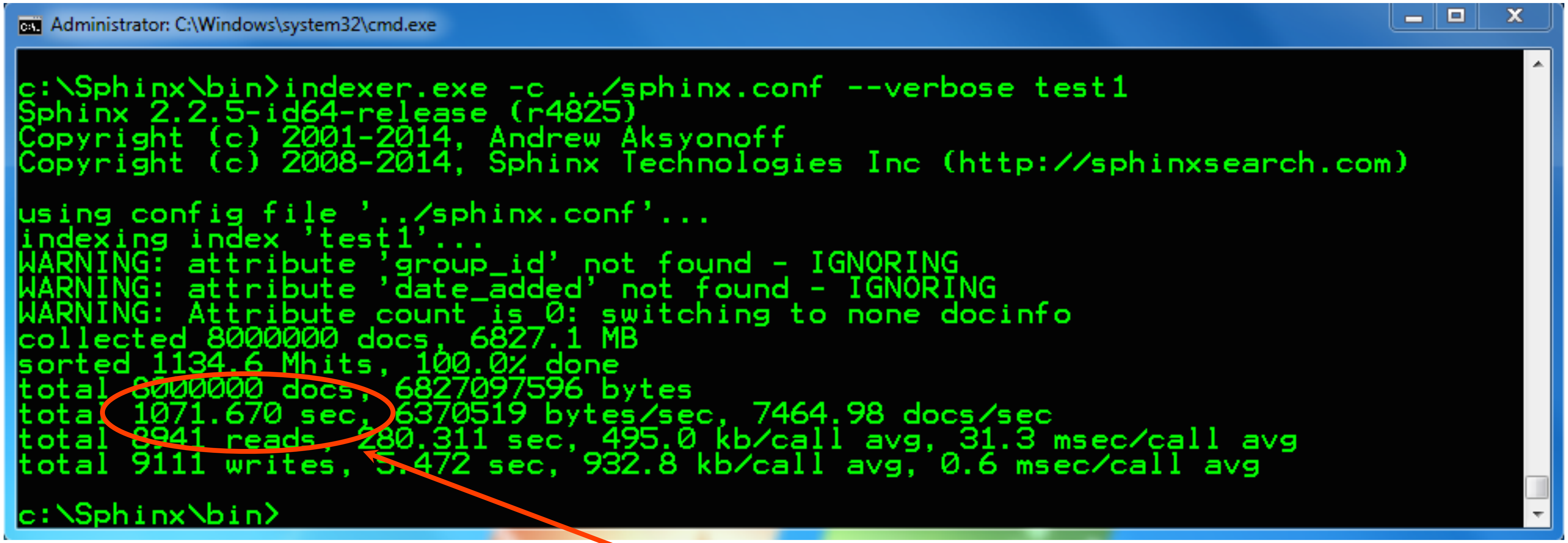
sphinx.conf

```
source src1
{
 type = mysql
 sql_host = localhost
 sql_user = xxxx
 sql_pass = xxxx
 sql_db = testpattern
 sql_query = SELECT PostId, ParentId, Title,
    Body, Tags FROM Posts
 sql_query_info = SELECT * FROM Posts \
    WHERE PostId=$id
}
```

sphinx.conf

```
index test1
{
 source = src1
 path = C:\Sphinx\data
}
```

Insert Data into Index (Sphinx)



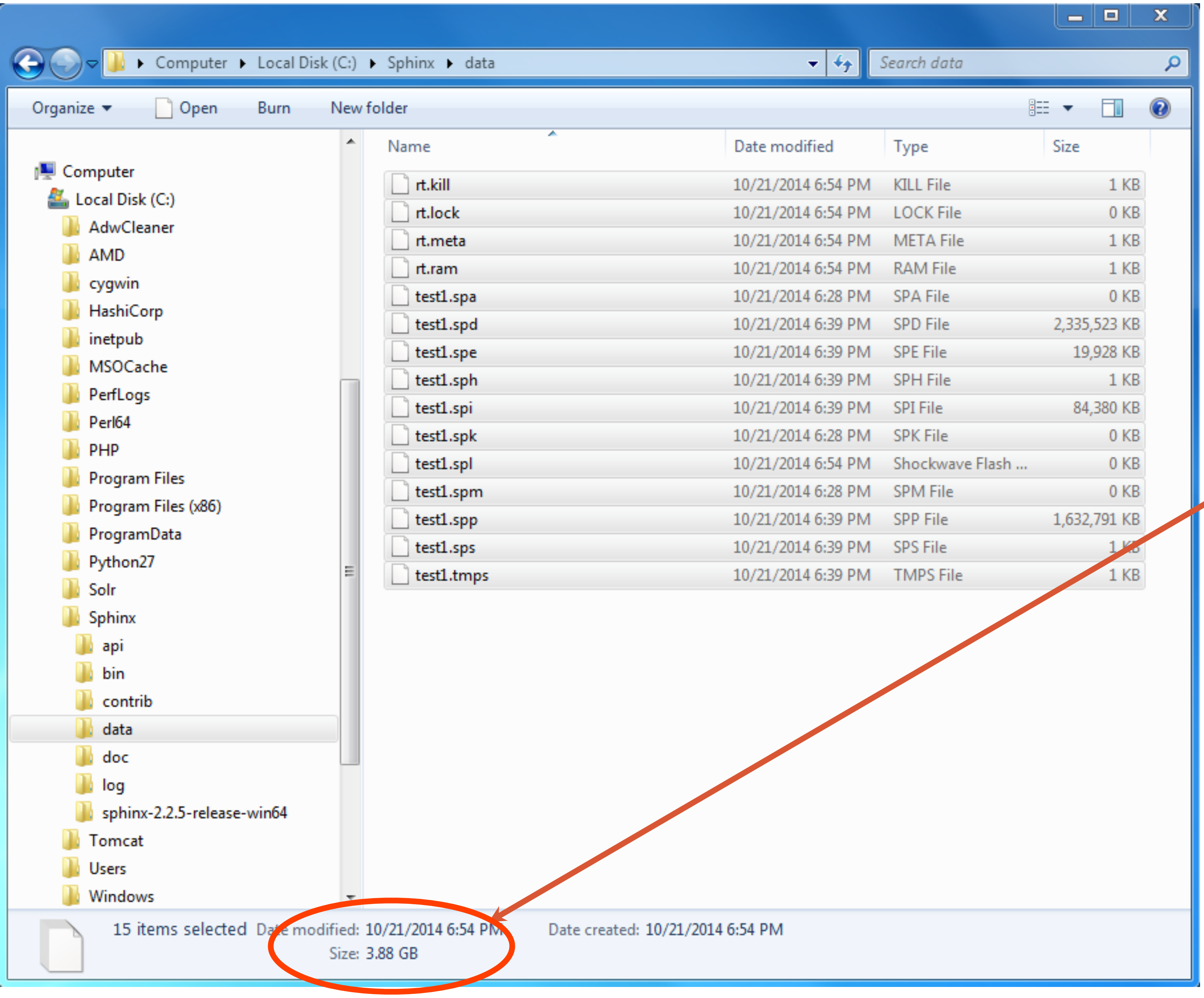
```
c:\Sphinx\bin>indexer.exe -c ../sphinx.conf --verbose test1
Sphinx 2.2.5-id64-release (r4825)
Copyright (c) 2001-2014, Andrew Aksyonoff
Copyright (c) 2008-2014, Sphinx Technologies Inc (http://sphinxsearch.com)

using config file '../sphinx.conf'...
indexing index 'test1'...
WARNING: attribute 'group_id' not found - IGNORING
WARNING: attribute 'date_added' not found - IGNORING
WARNING: Attribute count is 0: switching to none docinfo
collected 8000000 docs, 6827.1 MB
sorted 1134.6 Mhits, 100.0% done
total 8000000 docs, 6827097596 bytes
total 1071.670 sec, 6370519 bytes/sec, 7464.98 docs/sec
total 8941 reads, 280.311 sec, 495.0 kb/call avg, 31.3 msec/call avg
total 9111 writes, 5.472 sec, 932.8 kb/call avg, 0.6 msec/call avg

c:\Sphinx\bin>
```

time: 17 min 52 sec

Index Size (Sphinx)



Name	Date modified	Type	Size
rt.kill	10/21/2014 6:54 PM	KILL File	1 KB
rt.lock	10/21/2014 6:54 PM	LOCK File	0 KB
rt.meta	10/21/2014 6:54 PM	META File	1 KB
rt.ram	10/21/2014 6:54 PM	RAM File	1 KB
test1.spa	10/21/2014 6:28 PM	SPA File	0 KB
test1.spd	10/21/2014 6:39 PM	SPD File	2,335,523 KB
test1.spe	10/21/2014 6:39 PM	SPE File	19,928 KB
test1.sph	10/21/2014 6:39 PM	SPH File	1 KB
test1.spi	10/21/2014 6:39 PM	SPI File	84,380 KB
test1.spk	10/21/2014 6:28 PM	SPK File	0 KB
test1.spl	10/21/2014 6:54 PM	Shockwave Flash ...	0 KB
test1.spm	10/21/2014 6:28 PM	SPM File	0 KB
test1.spp	10/21/2014 6:39 PM	SPP File	1,632,791 KB
test1.sps	10/21/2014 6:39 PM	SPS File	1 KB
test1.tmps	10/21/2014 6:39 PM	TMPS File	1 KB

15 items selected Date modified: 10/21/2014 6:54 PM Date created: 10/21/2014 6:54 PM
Size: 3.88 GB

size: 3.88 GB

Querying index

```
$ mysql --port 9306

Server version: 2.2.5-id64-release (r4825)

mysql> SELECT * FROM test1 WHERE MATCH('mysql performance');
+---------+
| id      |
+---------+
| 6016856 |
| 4207641 |
| 2656325 |
| 7192928 |
| 8118235 |
. . .
20 rows in set (0.04 sec)
```

Querying index

```
mysql> SHOW META;

+---------------+-------------+
| Variable_name | Value       |
+---------------+-------------+
| total         | 1000        |
| total_found   | 8340        |
| time          | 0.037       |
| keyword[0]    | mysql       |
| docs[0]       | 179579      |
| hits[0]       | 404247      |
| keyword[1]    | performance |
| docs[1]       | 158433      |
| hits[1]       | 227427      |
+---------------+-------------+
```

time: 37ms

Trigraphs

Trigraphs Overview

- Not very fast, but still better than LIKE / RLIKE
- Generic, portable SQL solution
- No dependency on version, storage engine, third-party technology

Three-Letter Sequences

```
CREATE TABLE AtoZ (
 c       CHAR(1),
 PRIMARY KEY (c));

INSERT INTO AtoZ (c)
VALUES ('a'), ('b'), ('c'), ...

CREATE TABLE Trigraphs (
 Tri      CHAR(3),
 PRIMARY KEY (Tri));

INSERT INTO Trigraphs (Tri)
SELECT CONCAT(t1.c, t2.c, t3.c)
FROM AtoZ t1 JOIN AtoZ t2 JOIN AtoZ t3;
```

Insert Data Into Index

```
my $sth = $dbh1->prepare("SELECT * FROM Posts") or die $dbh1->errstr;
$sth->execute() or die $dbh1->errstr;
$dbh2->begin_work;
my $i = 0;
while (my $row = $sth->fetchrow_hashref ) {
  my $text = lc(join('|', ($row->{title}, $row->{body}, $row->{tags})));
  my %tri;
  map($tri{$_}=1, ( $text =~ m/[[:alpha:]]{3}/g ));
  next unless %tri;
  my $tuple_list = join(",", map("('$_',$row->{postid})", keys %tri));
  my $sql = "INSERT IGNORE INTO PostsTrigraph (tri, PostId) VALUES
 $tuple_list";
  $dbh2->do($sql) or die "SQL = $sql, ".$dbh2->errstr;
  if (++$i % 1000 == 0) {
    print ".";
    $dbh2->commit;
    $dbh2->begin_work;
  }
}
print ".\n";
$dbh2->commit;
```

takes hours, and creates a very large number of rows

Indexed Lookups

```
SELECT p.*
FROM Posts p
JOIN PostsTrigraph t1 ON
 t1.PostId = p.PostId AND t1.Tri = 'mys'
```

Search Among Fewer Matches

```
SELECT p.*
FROM Posts p
JOIN PostsTrigraph t1 ON
 t1.PostId = p.PostId AND t1.Tri = 'mys'
JOIN PostsTrigraph t2 ON
 t2.PostId = p.PostId AND t2.Tri = 'per'
```

Search Among Fewer Matches

```
SELECT p.*
FROM Posts p
JOIN PostsTrigraph t1 ON
 t1.PostId = p.PostId AND t1.Tri = 'mys'
JOIN PostsTrigraph t2 ON
 t2.PostId = p.PostId AND t2.Tri = 'per'
JOIN PostsTrigraph t3 ON
 t3.PostId = p.PostId AND t3.Tri = 'for'
```

Search Among Fewer Matches

```
SELECT p.*
FROM Posts p
JOIN PostsTrigraph t1 ON
 t1.PostId = p.PostId AND t1.Tri = 'mys'
JOIN PostsTrigraph t2 ON
 t2.PostId = p.PostId AND t2.Tri = 'per'
JOIN PostsTrigraph t3 ON
 t3.PostId = p.PostId AND t3.Tri = 'for'
JOIN PostsTrigraph t4 ON
 t4.PostId = p.PostId AND t4.Tri = 'man'
```

Narrow Down Further

```
SELECT p.*
FROM Posts p
JOIN PostsTrigraph t1 ON
 t1.PostId = p.PostId AND t1.Tri = 'mys'
JOIN PostsTrigraph t2 ON
 t2.PostId = p.PostId AND t2.Tri = 'per'
JOIN PostsTrigraph t3 ON
 t3.PostId = p.PostId AND t3.Tri = 'for'
JOIN PostsTrigraph t4 ON
 t4.PostId = p.PostId AND t4.Tri = 'man'
WHERE CONCAT(p.title,p.body,p.tags) LIKE '%mysql%'
 AND CONCAT(p.title,p.body,p.tags) LIKE '%performance%';
```

Not Recommended

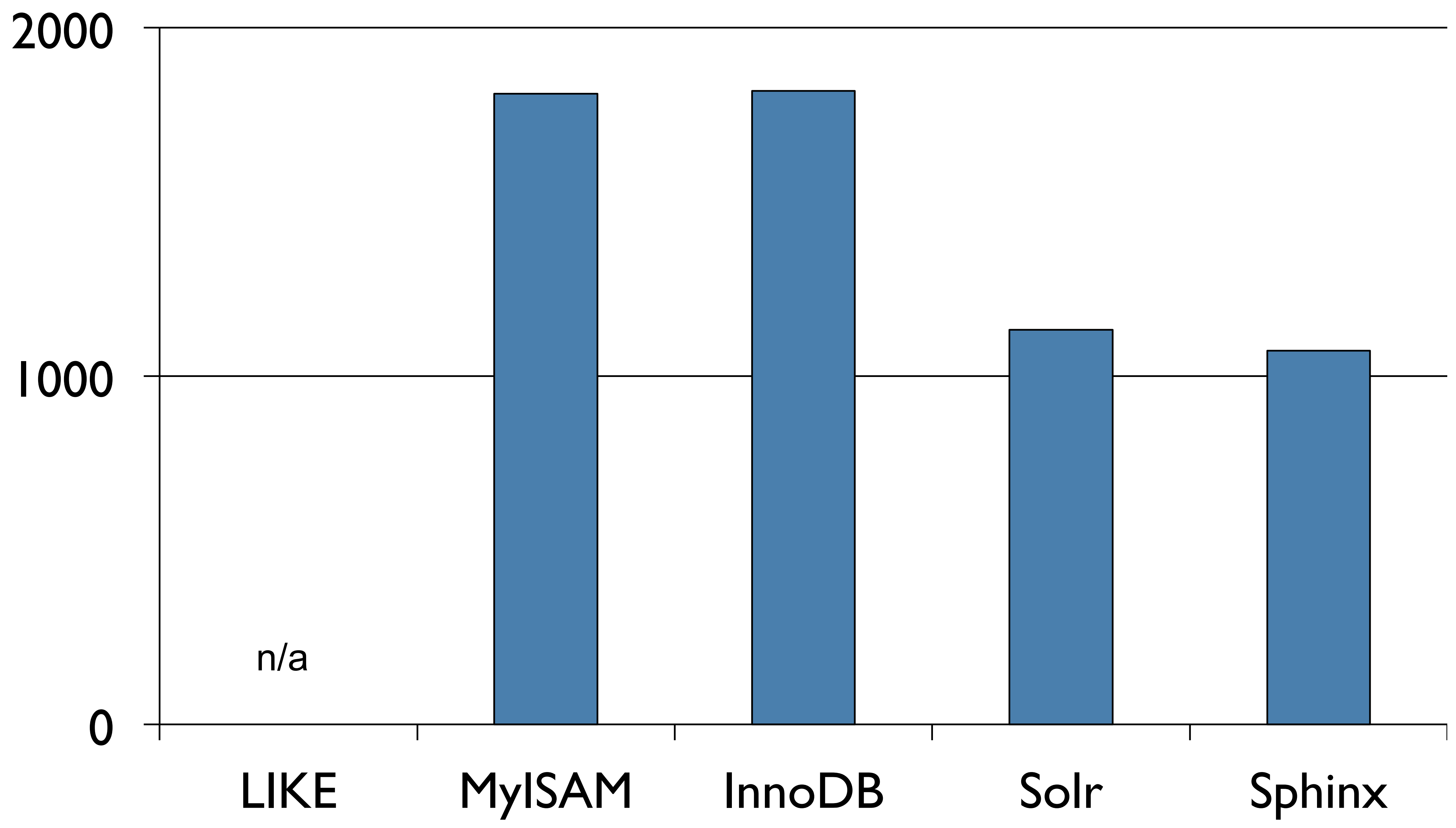
- Best query performance was still > 15 sec.
- Specialized fulltext search technology is much better, so trigraphs are useful only when portable, standard SQL is the only solution allowed.

And the winner is...

Time to Build Index on Data

LIKE expression	n/a
FULLTEXT MyISAM	30 min, 10 sec
FULLTEXT InnoDB	30 min, 19 sec
Apache Solr	18 min, 52 sec
Sphinx Search	17 min, 52 sec

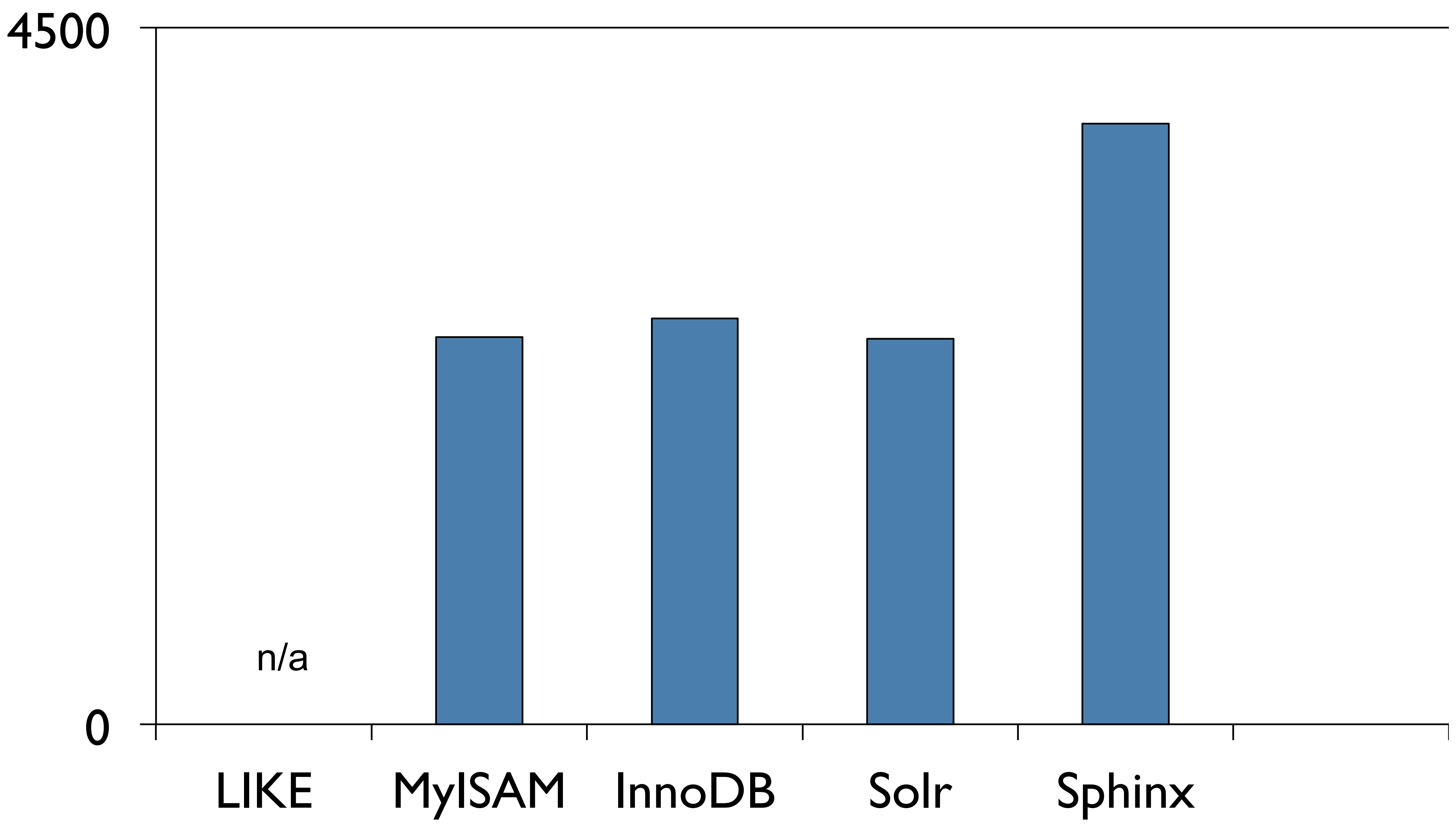
Build Index on Data (sec)



Index Storage

LIKE expression	n/a
FULLTEXT MyISAM	2.50 GB
FULLTEXT InnoDB	2.62 GB
Apache Solr	2.49 GB
Sphinx Search	3.88 GB

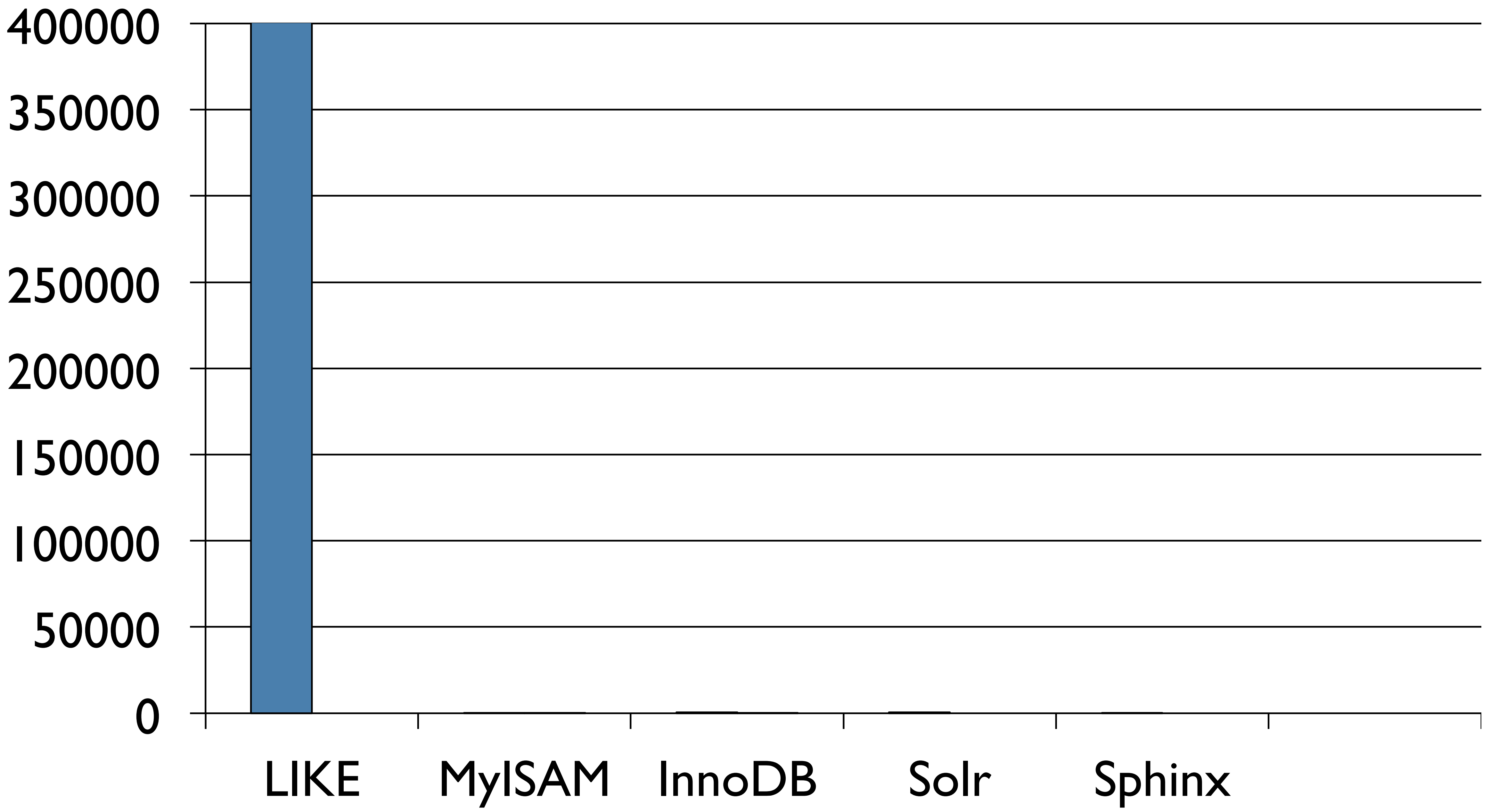
Index Storage (MiB)



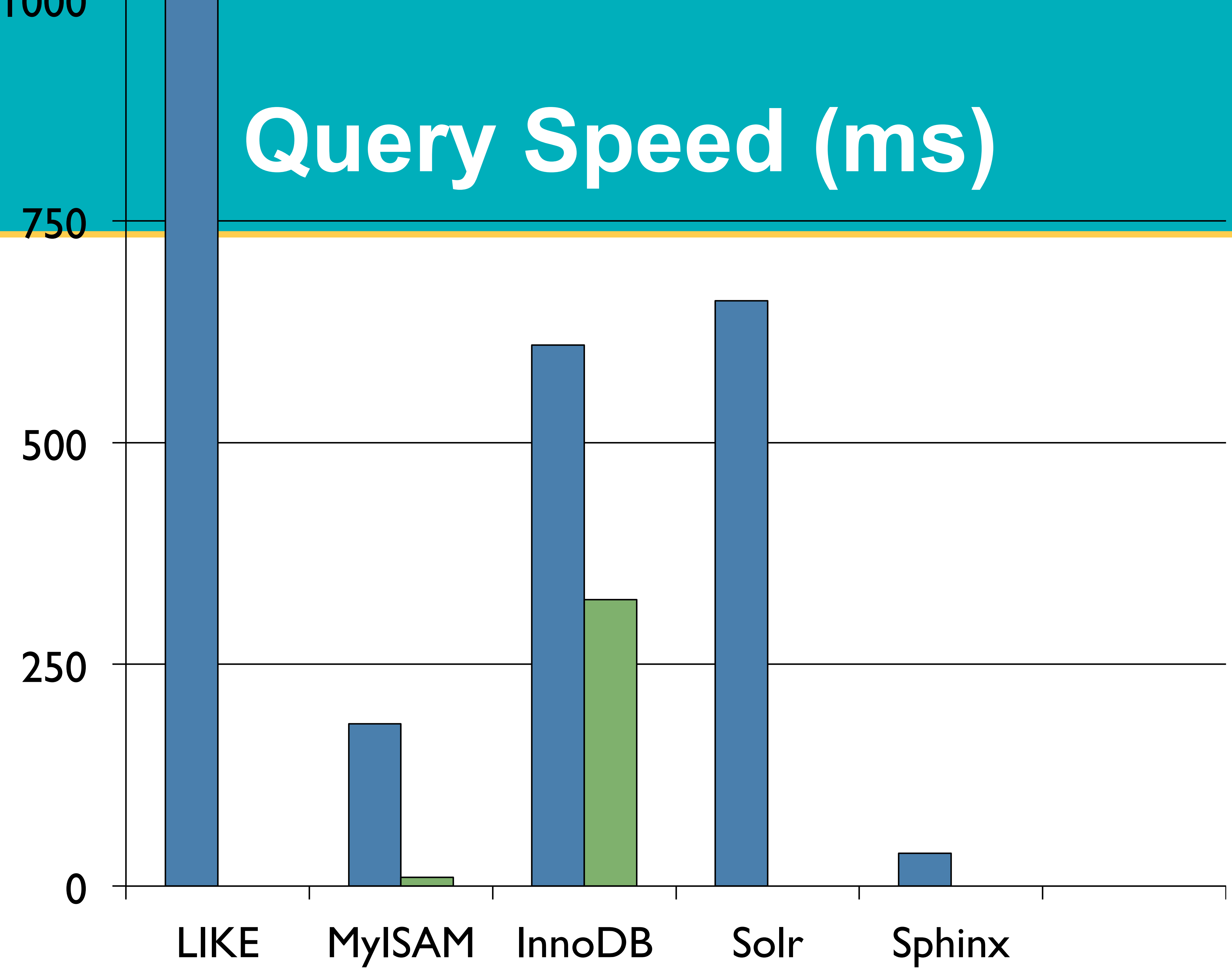
Query Speed

LIKE / RLIKE	97000-655000ms
FULLTEXT MyISAM	10-183ms
FULLTEXT InnoDB	323-610ms
Apache Solr	60-1700ms
Sphinx Search	37ms

Query Speed (ms)



Query Speed (ms)



Bottom Line

	build	*storage*	*query*	*solution*
LIKE expression	0	0	97,000ms	SQL
RLIKE expression	0	0	655,000ms	SQL
FULLTEXT MyISAM	30:10	2.50GB	183ms	MySQL
FULLTEXT InnoDB	30:19	2.62GB	323ms	MySQL 5.6
Apache Solr	18:52	2.49GB	660ms	Java
Sphinx Search	17:52	3.88GB	37ms	C++

Final Thoughts

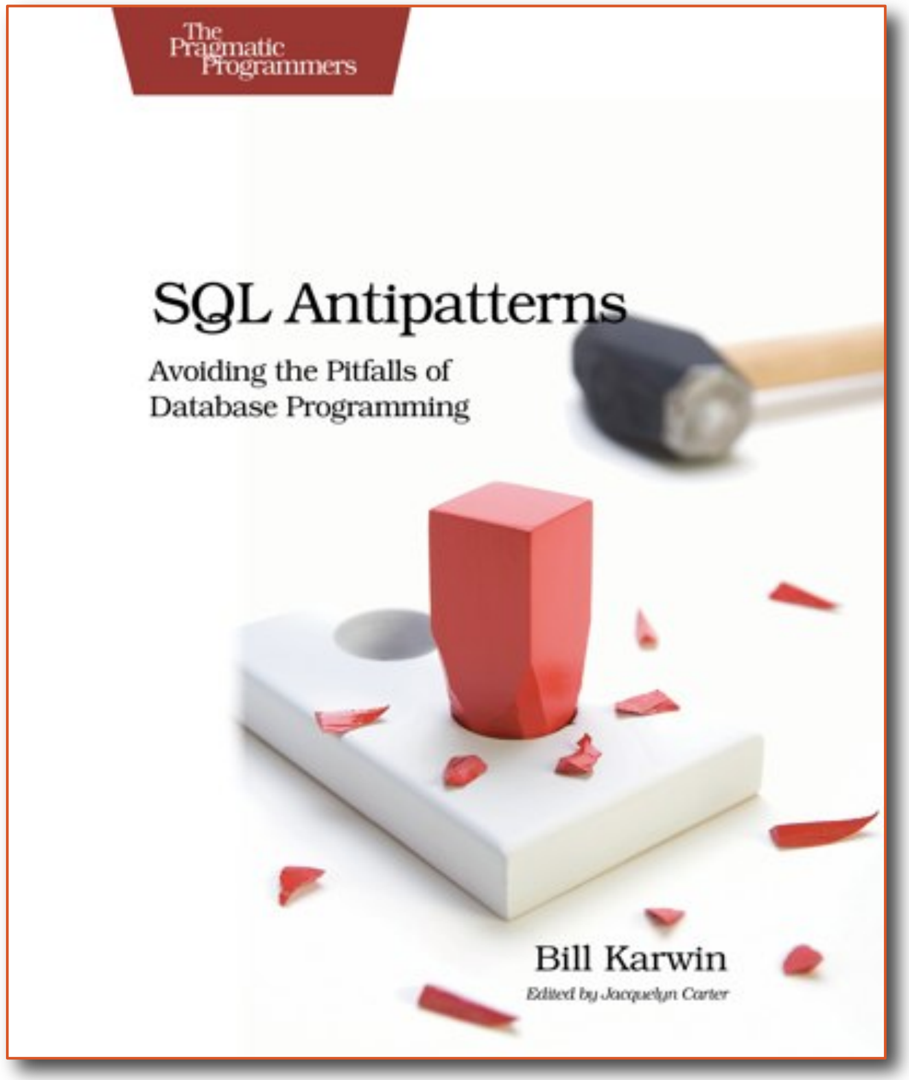
- Third-party search engines are complex to keep in sync with data, and adding another type of server adds more operations work for you.
- Built-in FULLTEXT indexes are therefore useful even if they are not absolutely the fastest.

Final Thoughts

- Different search implementations may return different results, so you should evaluate what works best for your project.

Final Thoughts

- *Any* indexed search solution is orders of magnitude better than `LIKE`!


The Pragmatic Programmers
SQL Antipatterns
Avoiding the Pitfalls of Database Programming
Bill Karwin
Edited by Jacquelyn Carter

http://www.pragprog.com/titles/bksqla/

London, November 3-4, 2014
Santa Clara, April 13-16, 2015

www.percona.com/live

PERCONA LIVE

Super-saver registration for Santa Clara until Nov. 16

Promo code: COUNTDOWN for £30 discount
for new registrations at PLUK!