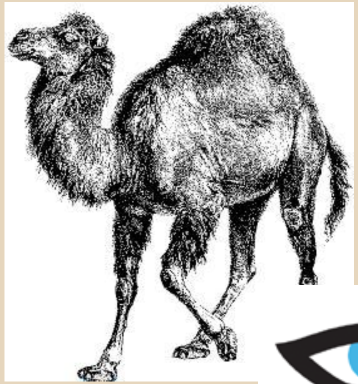
Indexing Stuff && Things with Sphinx and Perl



Houston Perl Mongers

May 8th, 2014
Hosted by cPanel, Inc.

Brett Estrade <estrabd@gmail.com>

Sphinx

- full text search indexer and daemon
- `indexer` - builds indexes
- `searchd` - services search requests
- very easy to install and configure

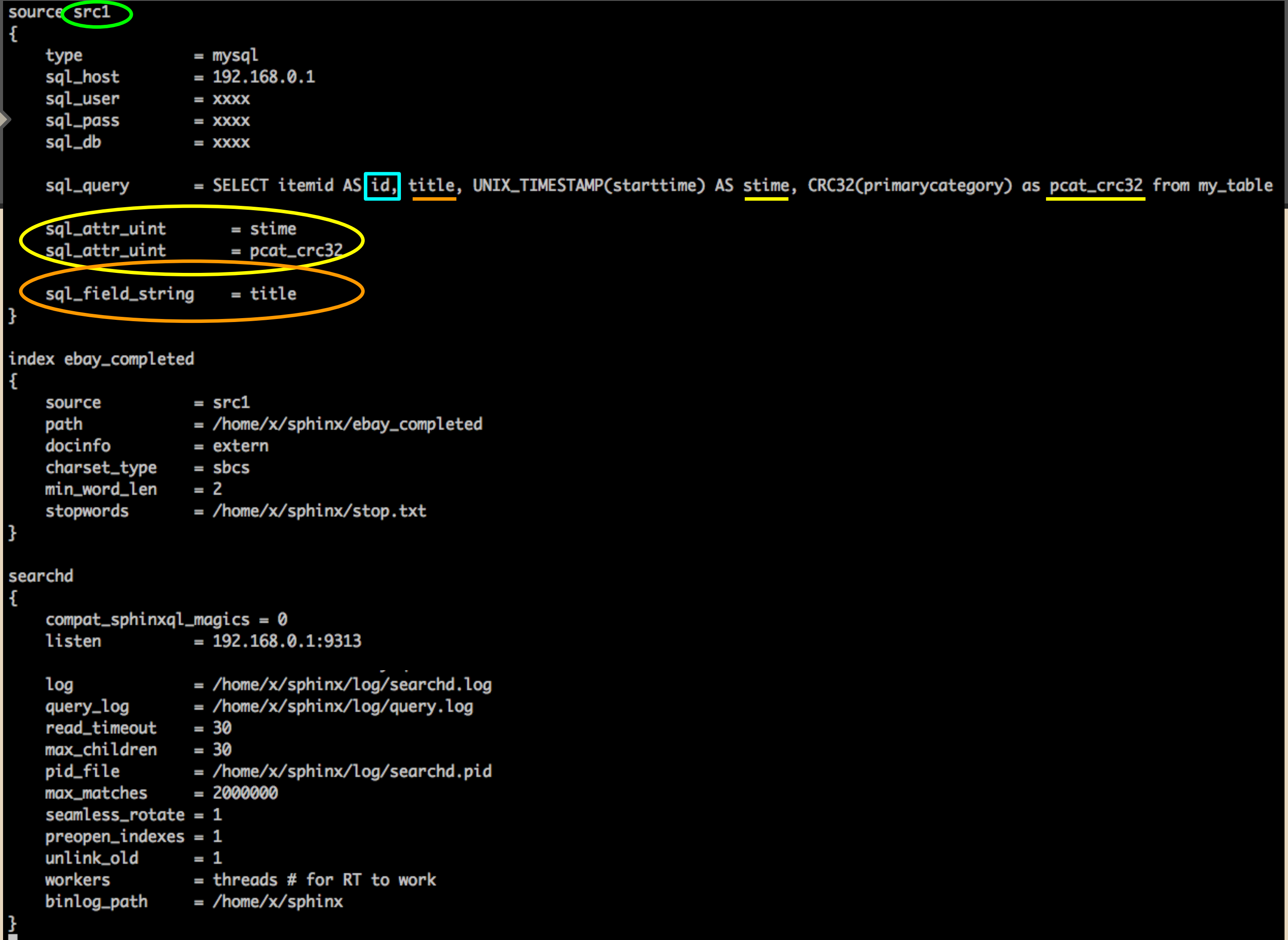
Sphinx Data Sources

- Directly from MySQL (MariaDB), PostgreSQL
  - Indexing data from arbitrary SQL
  - Excellent for fast reading of expensive JOINs
- XMLPipe2
  - General intermediate data understood by Sphinx

Search Interface

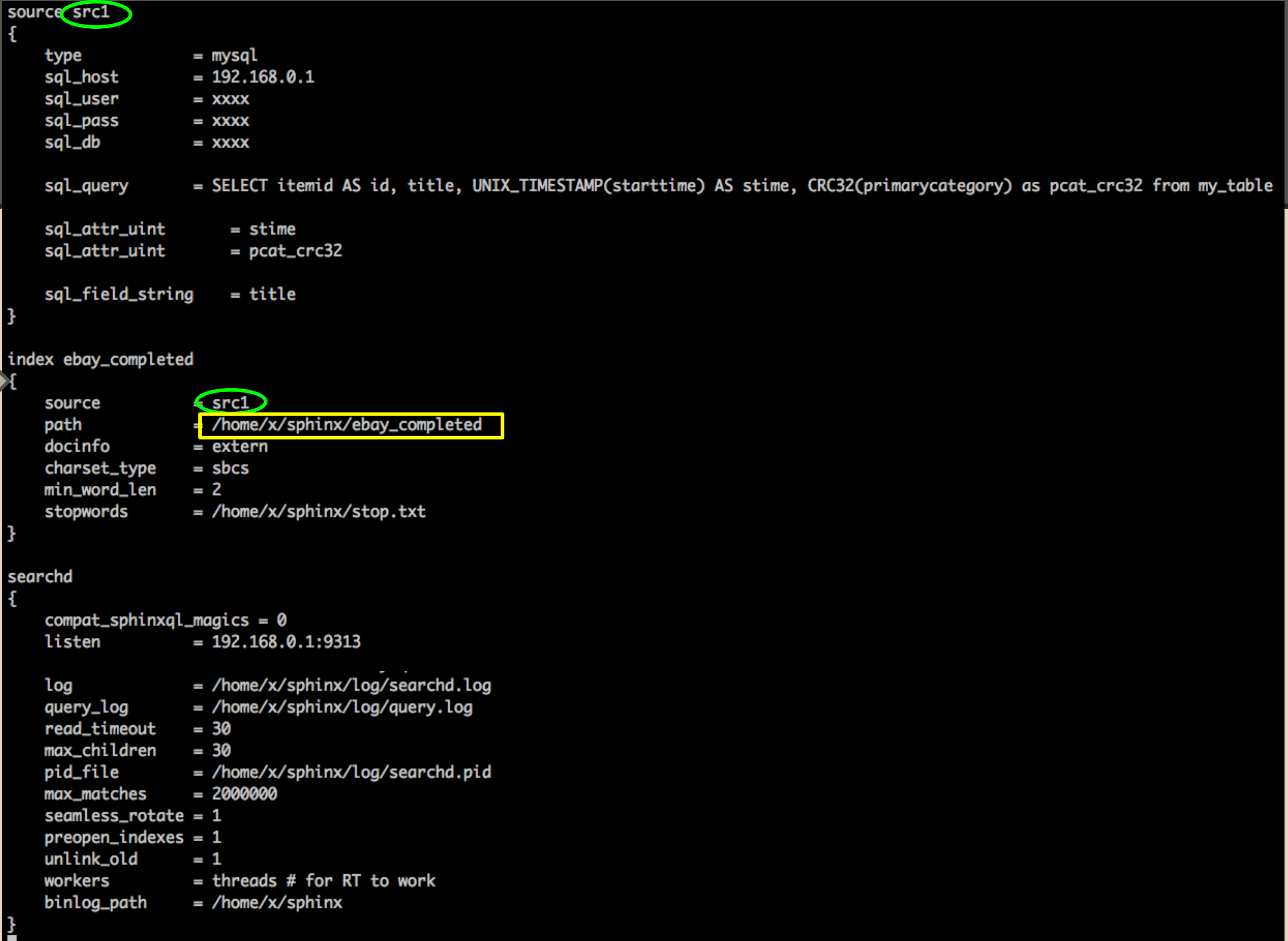
- Native protocol (e.g., Sphinx::Search)
- Supports MySQL protocol (4.1)
  - Subset of SQL supported is called *SphinxQL*

`indexer` data

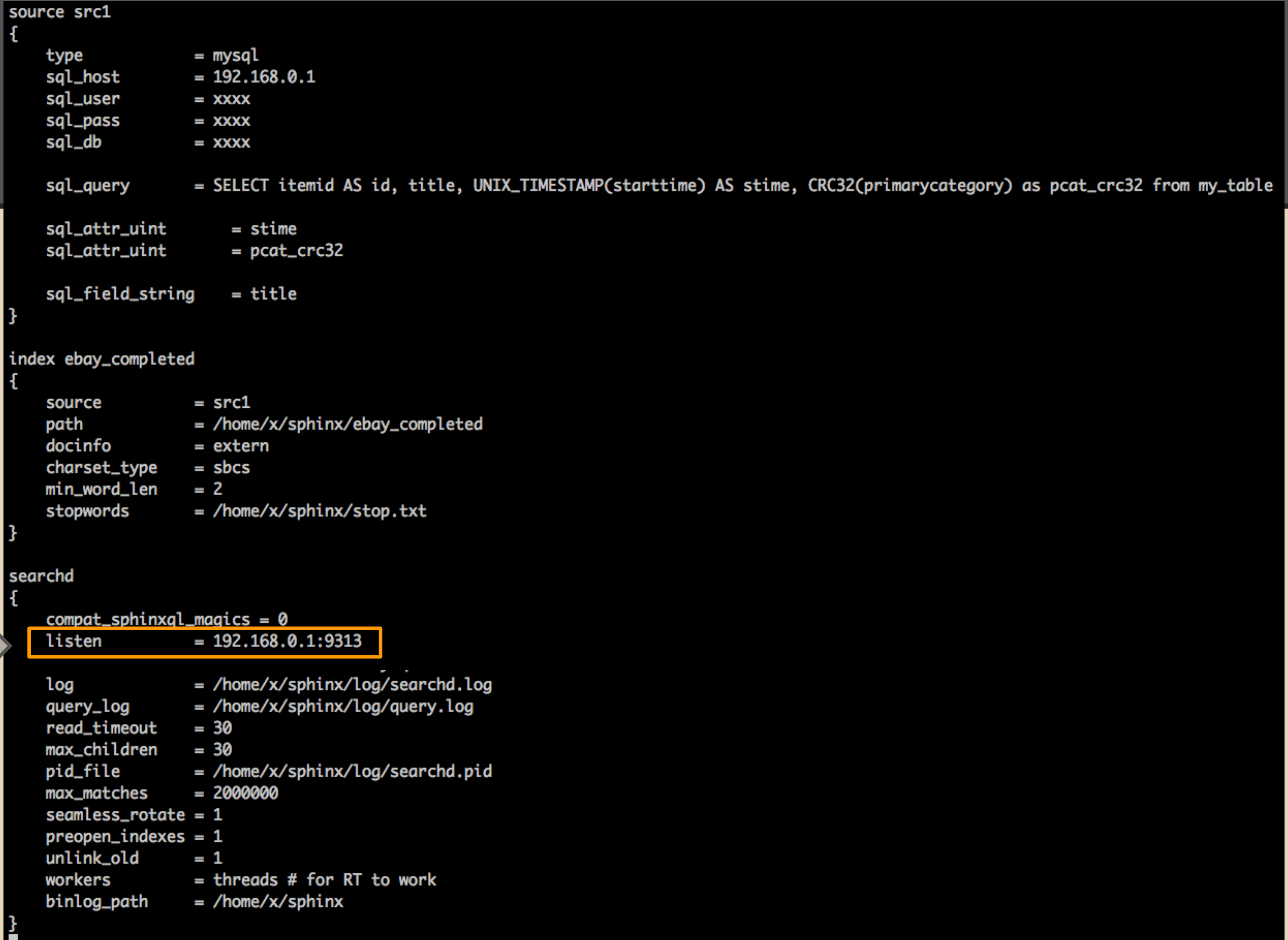


```
source src1
{
    type             = mysql
    sql_host         = 192.168.0.1
    sql_user         = xxxx
    sql_pass         = xxxx
    sql_db           = xxxx

    sql_query        = SELECT itemid AS id, title, UNIX_TIMESTAMP(starttime) AS stime, CRC32(primarycategory) as pcat_crc32 from my_table

    sql_attr_uint        = stime
    sql_attr_uint        = pcat_crc32

    sql_field_string     = title
}

index ebay_completed
{
    source           = src1
    path             = /home/x/sphinx/ebay_completed
    docinfo          = extern
    charset_type     = sbcs
    min_word_len     = 2
    stopwords        = /home/x/sphinx/stop.txt
}

searchd
{
    compat_sphinxql_magics = 0
    listen           = 192.168.0.1:9313

    log              = /home/x/sphinx/log/searchd.log
    query_log        = /home/x/sphinx/log/query.log
    read_timeout     = 30
    max_children     = 30
    pid_file         = /home/x/sphinx/log/searchd.pid
    max_matches      = 2000000
    seamless_rotate  = 1
    preopen_indexes  = 1
    unlink_old       = 1
    workers          = threads # for RT to work
    binlog_path      = /home/x/sphinx
}
```

named index for
`searchd`



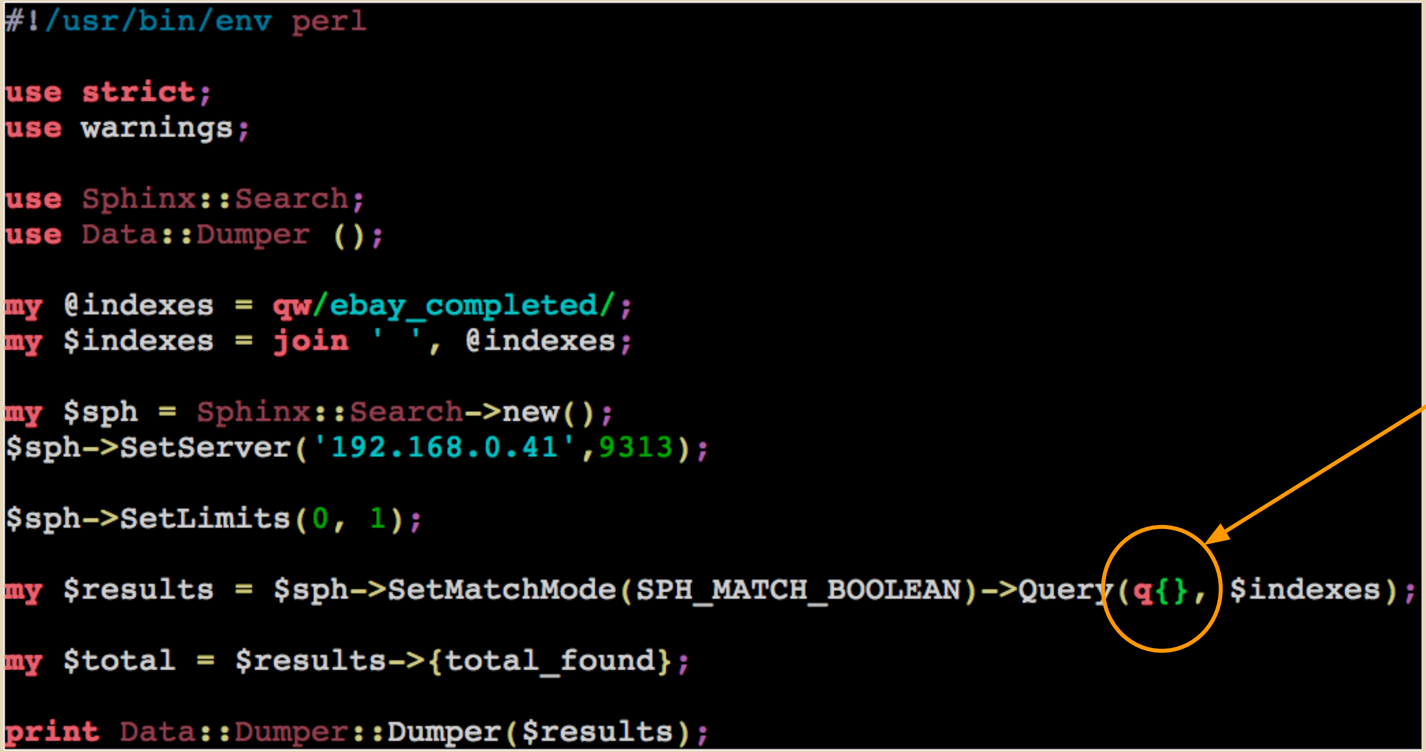
```
source src1
{
    type              = mysql
    sql_host          = 192.168.0.1
    sql_user          = xxxx
    sql_pass          = xxxx
    sql_db            = xxxx

    sql_query         = SELECT itemid AS id, title, UNIX_TIMESTAMP(starttime) AS stime, CRC32(primarycategory) as pcat_crc32 from my_table

    sql_attr_uint         = stime
    sql_attr_uint         = pcat_crc32

    sql_field_string      = title
}

index ebay_completed
{
    source            = src1
    path              = /home/x/sphinx/ebay_completed
    docinfo           = extern
    charset_type      = sbcs
    min_word_len      = 2
    stopwords         = /home/x/sphinx/stop.txt
}

searchd
{
    compat_sphinxql_magics = 0
    listen            = 192.168.0.1:9313

    log               = /home/x/sphinx/log/searchd.log
    query_log         = /home/x/sphinx/log/query.log
    read_timeout      = 30
    max_children      = 30
    pid_file          = /home/x/sphinx/log/searchd.pid
    max_matches       = 2000000
    seamless_rotate = 1
    preopen_indexes = 1
    unlink_old        = 1
    workers           = threads # for RT to work
    binlog_path       = /home/x/sphinx
}
```



```
source src1
{
    type            = mysql
    sql_host        = 192.168.0.1
    sql_user        = xxxx
    sql_pass        = xxxx
    sql_db          = xxxx

    sql_query       = SELECT itemid AS id, title, UNIX_TIMESTAMP(starttime) AS stime, CRC32(primarycategory) as pcat_crc32 from my_table

    sql_attr_uint       = stime
    sql_attr_uint       = pcat_crc32

    sql_field_string    = title
}

index ebay_completed
{
    source          = src1
    path            = /home/x/sphinx/ebay_completed
    docinfo         = extern
    charset_type    = sbcs
    min_word_len    = 2
    stopwords       = /home/x/sphinx/stop.txt
}

searchd
{
    compat_sphinxql_magics = 0
    listen          = 192.168.0.1:9313

    log             = /home/x/sphinx/log/searchd.log
    query_log       = /home/x/sphinx/log/query.log
    read_timeout    = 30
    max_children    = 30
    pid_file        = /home/x/sphinx/log/searchd.pid
    max_matches     = 2000000
    seamless_rotate = 1
    preopen_indexes = 1
    unlink_old      = 1
    workers         = threads # for RT to work
    binlog_path     = /home/x/sphinx
}
```

searchd config

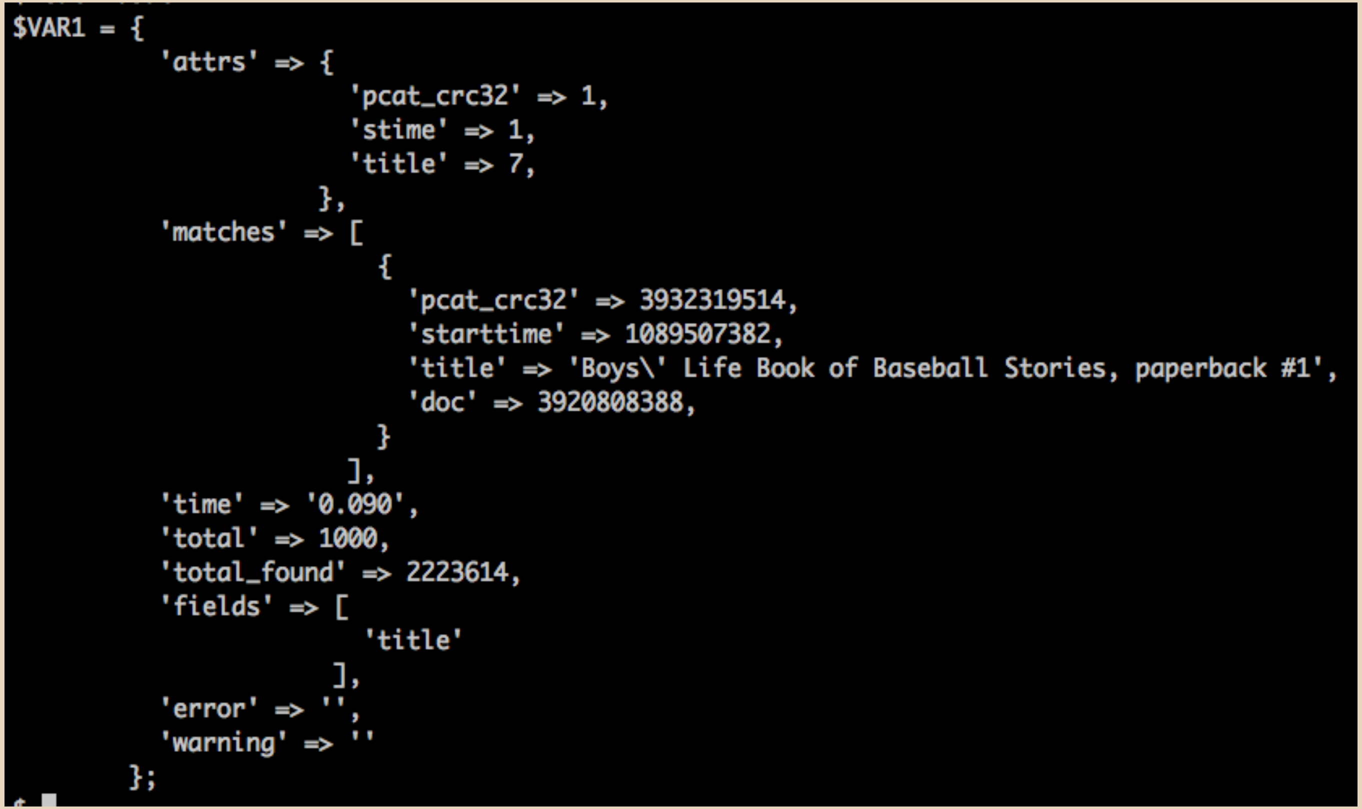
Client Example - Sphinx::Search



```
#!/usr/bin/env perl

use strict;
use warnings;

use Sphinx::Search;
use Data::Dumper ();

my @indexes = qw/ebay_completed/;
my $indexes = join ' ', @indexes;

my $sph = Sphinx::Search->new();
$sph->SetServer('192.168.0.41',9313);

$sph->SetLimits(0, 1);

my $results = $sph->SetMatchMode(SPH_MATCH_BOOLEAN)->Query(q{}, $indexes);

my $total = $results->{total_found};

print Data::Dumper::Dumper($results);
```

search term - empty string returns “all”

Search Results



```
$VAR1 = {
          'attrs' => {
                       'pcat_crc32' => 1,
                       'stime' => 1,
                       'title' => 7,
                     },
          'matches' => [
                         {
                           'pcat_crc32' => 3932319514,
                           'starttime' => 1089507382,
                           'title' => 'Boys\' Life Book of Baseball Stories, paperback #1',
                           'doc' => 3920808388,
                         }
                       ],
          'time' => '0.090',
          'total' => 1000,
          'total_found' => 2223614,
          'fields' => [
                        'title'
                      ],
          'error' => '',
          'warning' => ''
        };
```

Some Common Use Cases

- ★ Rebuild index from database regularly
- Incrementally add to existing index

- Query Sphinx for DB primary keys, make DB call for related rows
- ★ Query Sphinx for wanted data (no DB at all)

★ == my use case

Real Life Examples

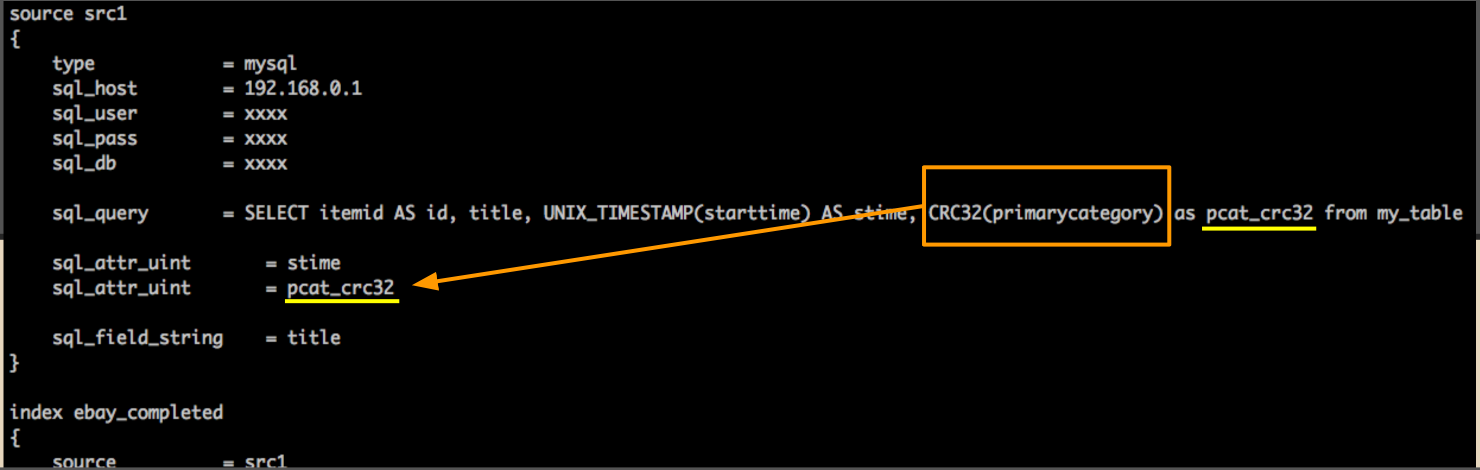
1. Indexing MariaDB
2. Filtering on string using CRC32
3. Creating sources w/Sphinx::XML::Pipe2
4. Dynamic config w/Sphinx::Config::Builder

Indexing MariaBD ~2.25 Million Rows

- Use case - saving eBay auction data in DB
- Providing search interface to it
- Demo run of indexer

How to Filter on Strings

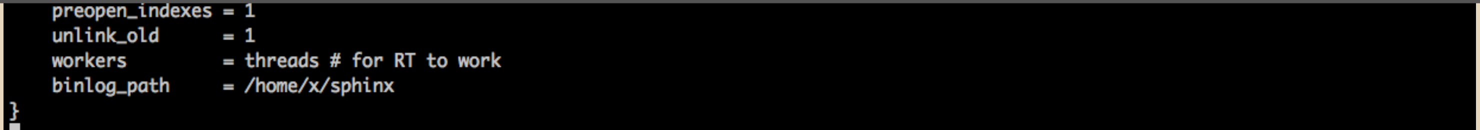
- Requires CRC32 hashing (strings to ints)
- When indexing, use MySQL’s CRC32 function
- Use Perl’s String::CRC32 to encode string,
    - then set filter



```
source src1
{
    type            = mysql
    sql_host        = 192.168.0.1
    sql_user        = xxxx
    sql_pass        = xxxx
    sql_db          = xxxx

    sql_query       = SELECT itemid AS id, title, UNIX_TIMESTAMP(starttime) AS stime, CRC32(primarycategory) as pcat_crc32 from my_table

    sql_attr_uint       = stime
    sql_attr_uint       = pcat_crc32

    sql_field_string    = title
}

index ebay_completed
{
    source          = src1
```

And inside of client, use Perl’s String::CRC32 to encode to the same integer

```
my $crc32 = undef;
my $cat   = $cgi->param('primaryCategory');
if ( $cat and grep { /$cat/ } @categories ) {
    $crc32 = String::CRC32::crc32($cat);
    eval { $sph->SetFilter( 'primarycategory_crc32', [$crc32] ); };
}
```

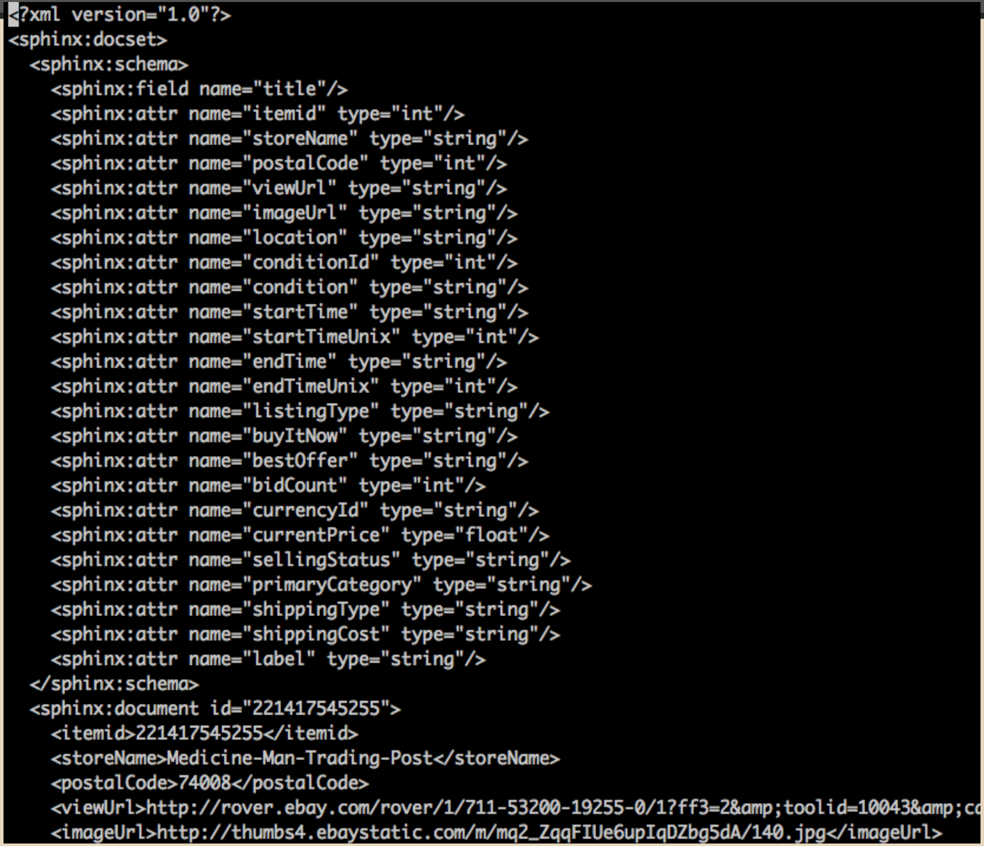


```
    preopen_indexes = 1
    unlink_old      = 1
    workers         = threads # for RT to work
    binlog_path     = /home/x/sphinx
}
```

Transforming Things to XMLPipe2

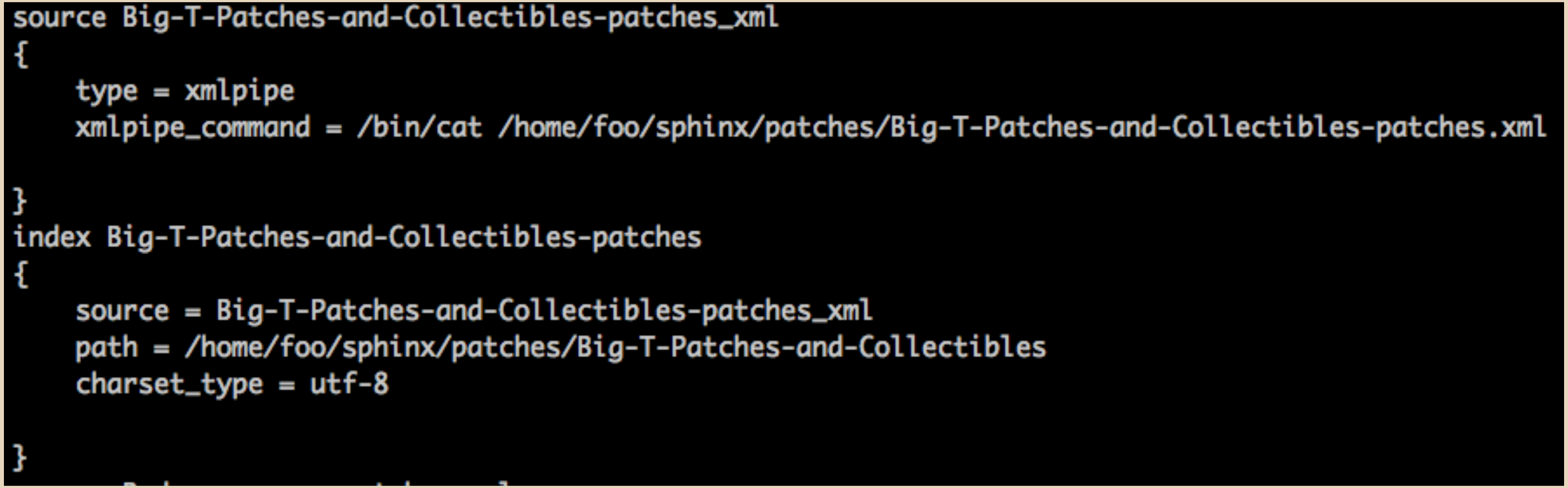
- XMLPipe2 is Sphinx’s generic data format
- Extract/Transform scripts -> XMLPipe2
- `use Sphinx::XML::Pipe2;` #’nuff said

Sample XMLPipe2 File



```
<?xml version="1.0"?>
<sphinx:docset>
  <sphinx:schema>
    <sphinx:field name="title"/>
    <sphinx:attr name="itemid" type="int"/>
    <sphinx:attr name="storeName" type="string"/>
    <sphinx:attr name="postalCode" type="int"/>
    <sphinx:attr name="viewUrl" type="string"/>
    <sphinx:attr name="imageUrl" type="string"/>
    <sphinx:attr name="location" type="string"/>
    <sphinx:attr name="conditionId" type="int"/>
    <sphinx:attr name="condition" type="string"/>
    <sphinx:attr name="startTime" type="string"/>
    <sphinx:attr name="startTimeUnix" type="int"/>
    <sphinx:attr name="endTime" type="string"/>
    <sphinx:attr name="endTimeUnix" type="int"/>
    <sphinx:attr name="listingType" type="string"/>
    <sphinx:attr name="buyItNow" type="string"/>
    <sphinx:attr name="bestOffer" type="string"/>
    <sphinx:attr name="bidCount" type="int"/>
    <sphinx:attr name="currencyId" type="string"/>
    <sphinx:attr name="currentPrice" type="float"/>
    <sphinx:attr name="sellingStatus" type="string"/>
    <sphinx:attr name="primaryCategory" type="string"/>
    <sphinx:attr name="shippingType" type="string"/>
    <sphinx:attr name="shippingCost" type="string"/>
    <sphinx:attr name="label" type="string"/>
  </sphinx:schema>
  <sphinx:document id="221417545255">
    <itemid>221417545255</itemid>
    <storeName>Medicine-Man-Trading-Post</storeName>
    <postalCode>74008</postalCode>
    <viewUrl>http://rover.ebay.com/rover/1/711-53200-19255-0/1?ff3=2&amp;toolid=10043&amp;cc
    <imageUrl>http://thumbs4.ebaystatic.com/m/mq2_ZqqFIUe6upIqDZbg5dA/140.jpg</imageUrl>
```

Sample XMLPipe2 Source Conf Entry



```
source Big-T-Patches-and-Collectibles-patches_xml
{
    type = xmlpipe
    xmlpipe_command = /bin/cat /home/foo/sphinx/patches/Big-T-Patches-and-Collectibles-patches.xml

}
index Big-T-Patches-and-Collectibles-patches
{
    source = Big-T-Patches-and-Collectibles-patches_xml
    path = /home/foo/sphinx/patches/Big-T-Patches-and-Collectibles
    charset_type = utf-8

}
```

Example XMLPipe2 Use Case

- Monitor ephemera,e.g. active eBay listings
- Don’t want to use a database
- Many data partitions (i.e., indexes)
    - e.g., by store, by category, etc
    - > 250 (yikes!)
- Data partitions change over time (slowly)

Dynamic Indexing of XMLPipe2 Stuff

- Fact - Sphinx partitions data by indexes
- Problem - each index uses its own data file
    - data as XMLPipe2
- Challenge - how to manage a changing set of indexes?

Sphinx’s --config to the Rescue!

- Config files are typically static, right?
- Sphinx can handle executables via --config
- `indexer --config ./generate-config.pl --all`

Sphinx::Config::Builder

- Module I created specifically for this case
  - uploaded to CPAN
- Why? No Sphinx config builders were a fit
- Module is low level and does what I need
  - i.e., dynamically builds a XMLPipe2 specific config
- A+ 100 Passing
  - http://cpantesters.org/distro/S/Sphinx-Config-Builder.html

Solution

- Expects XML2Pipe data files to already exist
- Iterate over array of indexes to build
- Creates “source” entries for XMLPipe2 data
- Creates “index” entries for each “source”

Demo

Tip of the Iceberg

- Sphinx has TONs of options and modes
- Tons of areas of application
- Many clients, Simple interface
- Super easy to install and maintain

Thank You!

- http://sphinxsearch.com/
- cpan://Sphinx::Search
- cpan://Sphinx::Config::Builder
- http://houston.pm.org