Impact Factor ISRA (India) = **1.344**
Impact Factor ISI (Dubai, UAE) = **0.829**
based on International Citation Report (ICR)

Impact Factor JIF = **1.500**
Impact Factor GIF (Australia) = **0.356**
Impact Factor SIS (USA) = **0.438**

Doi: 10.15863/TAS
International Scientific Journal
Theoretical & Applied Science

p-ISSN: 2308-4944 (print) **e-ISSN:** 2409-0085 (online)

Year: 2015 **Issue:** 01 **Volume:** 21

Published: 30.01.2015 **http://www.T-Science.org**

SECTION 4. Computer science, computer engineering and automation.

Nikita Sergeevich Drozdovsky
master of
North-Caucasus Federal University, Stavropol, Russia
nikita_drozdovsk@mail.ru

Pavel Vladimirovich Dvoryaninov
master of
North-Caucasus Federal University, Stavropol, Russia
paveldv92@mail.ru

Yaroslav Yurievich Lenskiy
master of
North-Caucasus Federal University, Stavropol, Russia
yalenskiy@yandex.ru

MYSQL FULL-TEXT SEARCH OPTIMIZATION BY INTEGRATION WITH SPHINX

Abstract: *Now people, using the Internet, want to receive necessary information instantly, with hardly any trouble at all, the full and exact. In this regard, even want from the small web sites functionality of a wide range and it is promoted: RSS, blogs, forums, full-fledged search engines. The PHP developers can find a set of the software for RSS, blogs and forums to add the site. Also there is a set of search engines which I know practically everything, but it doesn't mean that they well approach all sites. If information of the site is too specific or users need the search adapted for processes of work of this site, full work it will be better to add global search engines local, specially created for the concrete site. In this article development of small part of the web site will be considered and the main attention is paid to components necessary for effective search: to a search engine, PHP interface and database.*

Key words: *Web-sites, PHP, data bases, Sphinx, full-fledged search.*

Language: *Russian*

Citation: Drozdovsky NS, Dvoryaninov PV, Lenskiy YY (2015) MYSQL FULL-TEXT SEARCH OPTIMIZATION BY INTEGRATION WITH SPHINX. ISJ Theoretical & Applied Science 01 (21): 116-120. doi: http://dx.doi.org/10.15863/TAS.2015.01.21.19

ОПТИМИЗАЦИЯ ПОЛНОТЕКСТОВОГО ПОИСКА В MYSQL ПУТЕМ ИНТЕГРАЦИИ С SPHINX

Аннотация: *В настоящее время люди, пользуясь интернетом, хотят получать необходимую информацию моментально, без каких-либо проблем, наиболее полную и точную. В связи с этим, даже от небольших web-сайтов хотят функционал широкого спектра и этому способствуют: RSS, блоги, форумы, полноценные поисковые системы. Разработчики PHP могут найти множество программного обеспечения для RSS, блогов и форумов, чтобы дополнить свой сайт. Также существует множество поисковых систем, которые знаю практически все, но это не значит, что они хорошо подходят ко всем сайтам. Если информация сайта слишком специфична или пользователям нужен поиск, адаптированный к процессам работы этого сайта, то полноценной работы лучше будет дополнить глобальные поисковые системы локальной, специально созданной для конкретного сайта. В данной статье будет рассмотрена разработка малой части web-сайта и основное внимание уделяется компонентам, необходимым для эффективного поиска: поисковой системе, интерфейсу PHP и базе данных.*

Ключевые слова: *Web-сайты, PHP, база данных, Sphinx, полнотекстовый поиск.*

Для того чтобы реализовать на сайте собственную систему поиска, нужны данные и возможность их сортировки. В качестве данных на web-ресурсах выступают реляционные базы данных. Они обычно уже имеют встроенные возможности поиска. Но некоторые виды поиска могут быть слишком специализированными и, тем самым, слишком сложными. Это замедлит скорость обработки данных. И, чтобы не изменять структуру всех таблиц базы данных, можно использовать специализированную поисковую систему. Можно рассматривать много различных коммерческих поисковых систем, но это не всегда оправдано. Поэтому в качестве альтернативы можно рассматривать Sphinx. Sphinx – бесплатная поисковая система с открытым исходным кодом, предназначенная для быстрого поиска текста. Написан на C++, скомпилирован с помощью GNU, работает на 64-разрядных платформах под управлением всех

операционных систем. В нем реализовано множество функций, основные возможности:

- Индекс Sphinx можно распределить на несколько машин, обеспечив отказоустойчивость работы.
- Высокая скорость интеграции.
- Высокая скорость поиска: до 150-250 запросов в секунду на каждое процессорное ядро с 1000000 документов.
- Высокая масштабируемость.
- Поддержка распределенного поиска.
- Поддержка нескольких полей полнотекстового поиска в документе.
- Поддержка стоп-слов.
- Поддержка морфологического поиска – имеются встроенные модули для английского, русского и чешского языков; доступны модули для французского, испанского, португальского, итальянского, румынского, немецкого, голландского, шведского, норвежского, датского, финского, венгерского языков.
- Нативная поддержка MySQL.

Поддержка ODBC совместимых баз данных.

Sphinx состоит из двух компонентов:

1. Генератор индекса – indexer, который выполняет запросы к базам данных, индексирует каждый столбец в каждой строке результата и привязывает запись индекса к первичному ключу строки.

2. Поисковая система – searchd. Она представляет из себя демона, который получает критерии поиска и прочие параметры, проходит по нескольким индексам и возвращает результат. При нахождении соответствия поисковая система возвращает массив первичных ключей.

Рассмотрим использование Sphinx при работе сайта по продаже компьютерных комплектующих. Для реализации поиска будет использоваться источник данных MySQL и поисковая система Sphinx. База данных MySQL обладает широкими возможностями, но функция полнотекстового поиска – не самая сильная черта. Формат таблиц базы данных не поддерживает внешние ключи, и поэтому имеет ограниченную применимость.

На сайте по продаже комплектующий будет происходить поиск по серийному номеру, фирме-производителю, номеру изделия, году выпуска, состоянию, по описанию и по выборке нескольких характеристик. Для этого была создана база данных, имеющая таблицы: PC, Assembly, Inventory, Compatibility.

Таблица PC содержит описание компьютера. Листинг таблицы:

```
CREATE TABLE PC (
id int(10) unsigned NOT NULL auto_increment,
label varhar(7) NOT NULL, / наименование компьютера
description varchar(256) NOT NULL, / описание компьютера в произвольной форме
year_production int(4) NOT NULL, / год выпуска
PRIMARY KEY (id)
) ENGINE=InnoDB;
```

Данные для таблицы PC:

```
INSERT INTO PC
('id', 'label', 'description', 'year_production')
VALUES
(1, 'PC 1', 'Core 2 DUO', 2014),
(2, 'PC 2', 'AMD 10', 2012),
(3, 'PC 3', 'ADM 8', 2014),
(4, 'PC 4', 'Core 2 DUO', 2013);
```

Таблица Assembly содержит полный набор комплектующих компьютера. Она сопоставляет названию и описанию узла уникальный идентификатор. Листинг таблицы:

```
CREATE TABLE PC (
id int(10) unsigned NOT NULL auto_increment,
label varhar(7) NOT NULL,
description varchar(150) NOT NULL,
PRIMARY KEY (id)
) ENGINE=InnoDB;
```

Данные для таблицы Assembly:

```
INSERT INTO Assembly
(1, '5-00', 'CPU'),
(2, '4-00', 'GPU'),
(3, '3-00', 'RAM'),
(4, ''6-00, 'HD'),
(5, '11-00', 'Motherboard'),
(6, '100-00', 'Accessories');
```

Таблица Inventory содержит список деталей. Листинг таблицы:

```
CREATE TABLE Inventory (
id int(10) unsigned NOT NULL auto_increment,
partno varchar(32) NOT NULL,
description varchar(256) NOT NULL,
price float unsigned NOT NULL default '0',
PRIMARY KEY (id),
UNIQUE KEY partno USING BTREE (partno)
) ENGINE=INNDB;
```

Таблица Compatibility связывает комплектующие с другими комплектующими которые стоят на компьютере. В каждой строке содержится уникальный id, внешний ключ к строке таблицы Inventory, внешние ключи, указывающие на определенный компьютер и модификацию из таблицы PC Листинг таблицы:

```
CREATE TABLE Compatibility (
id int(10) unsigned NOT NULL auto_increment,
partno_id int(10) unsigned NOT NULL,
assembly_id int(10) unsigned NOT NULL,
pc_id int(10) unsigned NOT NULL,
PRIMARY KEY (id),
KEY partno_index USING BTREE (partno_id),
```

```
KEY assembly_index USING BTREE (assembly_id),
KEY pc_index USING BTREE (pc_id),
FOREIGN KEY (partno_id) REFERENCES Inventory(id),
FOREIGN KEY (assembly_id) REFERENCES Assembly(id),
FOREIGN KEY (pc_id) REFERENCES PC(id)
) ENGINE-InnoDB;
Данные для таблицы Compatibility:
INSERT INTO 'Compatibility'
('id', 'partno_id', 'assembly_id', 'pc_id')
VALUES
(1,6,5,1),
(2,8,5,1),
(3,1,3,1),
(4,5,3,1),
(5,8,5,7),
(6,6,5,7);
```

Имея такую структуру таблиц, пользуясь встроенной функцией поиска MySQL, можно выполнить множество поисковых запросов:

- Вывести все модификации компьютера.
- Показать все компоненты для сборки определенного компьютера.
- Показать все компоненты, которые имеют необходимый id.
- Показать все компьютеры определенного года.

Для поиска в больших объемах текстовых данных, необходимо создать несколько индексов с помощью Sphinx.

Источник (source) определяет базу данных, которую нужно индексировать, представляет информацию для аутентификации и указывает запросы, необходимые для формирования каждой строки. При желании можно использовать один или несколько столбцов как фильтры. В Sphinx эта функция называется группы – используются для фильтрации результатов.

Для индекса необходимо чтобы был определен источник и способ классификации данных из этого источника. Источники определяются в файле sphinx.conf. В данном случае источником является база данных MySQL. В листинге показана настройка доступа к базе данных.

```
source catalog
{ type        = mysql
  sql_host    = localhost
  sql_user    = reaper
  sql_pass    = 123321
  sql_db      = pc_shop
  sql_sock    = /var/run/mysqlb/mysqlb.sock
  sql_port    = 3306
}
```

Теперь необходимо создать запрос, возвращающий строки подлежащие индексации. Для поиска компьютера и года сборки используется таблица Assembly, а id детали и ее описание содержится в таблице Inventory. Для этого у Sphinx есть возможность связать результаты с 32-разрядным целочисленным первичным ключом. Для получения данных в нужной форме необходимо собрать все в виртуальную таблицу. Создание таблицы в листинге:

```
CREATE OR REPLACE VIEW Catalog AS
SELECT
Inventory.id,
Inventory.partno,
Inventory.description.
Assembly.id AS assembly,
PC.id AS pc
FROM
Assembly, Inventory, PC, Compatibility
WHERE
Compatibility.partno_id+Inventory.id
AND Compatibility.pc_id=pc.id
      AND
Compatibility.assembly_id=Assembly.id;
```

Пока id представления указывает на запись детали в таблице Inventory. Столбцы partno и description содержат текст для поиска, а столбцы assembly и pc – группы для фильтрации результатов. При таком представлении запросы создаются моментально. Листинг создания строк для индексации:

```
# indexer query
# document_id MUST be the very first field
# document_id MUST be positive (non-zero, non-negative)
# document_id MUST fit into 32 bits
# document_id MUST be unique
  sql_query                    =
        SELECT
              id, partno, description,
              assembly, pc
        FROM Catalog;
sql_group_column             = assembly
sql_group_column             = pc
# document info query
# ONLY used by search utility to display document information
# MUST be able to fetch document info by its id, therefore
# MUST contain '$id' macro
# sql_query_info             = SELECT * FROM Inventory WHERE id=$id
}
```

В запрос sql_query должен входить первичный ключ, который надо использовать в последующем для поиска, а также все поля, которые можно индексировать и использовать в качестве групп. Две записи sql_group_column объявляют, что для фильтрации результатов могут использоваться поля Assembly и PC. Для поиска нужных записей в поисковой утилите

используется sql_query_info. В запросе $id заменяется каждым первичным ключом, возвращенным searchd. Листинг описания одного из возможных индексов для источника catalog:

```
index catalog
{
source                  = catalog
path                    = /var/data/sphinx/catalog
morphology              = stem_en
min_word_len            = 3
min_prefix_len          = 0
min_infix_len           = 3
}
```

После этого можно приступать к созданию индекса для web-сайта. Листинг создания индекса:

```
$ sudo /usr/local/bin/indexer --config /usr/local/etc/sphinx.conf –all /аргумент all
```

перестраивает все индексы, перечисленные в sphinx.conf

```
Sphinx 0.9.7
Copyright (c) 2001-2007, Andrew Aksyonoff
using config file '/usr/local/etc/sphinx.conf'...
indexing index 'catalog'...
collected 8 docs, 0.0 MB
sorted 0.0 Mhits, 82.8% done
total 8 docs, 149 bytes
total 0.010 sec, 14900.00 bytes/sec, 800.00 docs/sec
```

Проверка индекса с помощью search:

```
$/usr/local/bin/search --config /usr/local/etc/sphinx.conf ENG
Sphinx 0.9.7
Copyright (c) 2001-2007, Andrew Aksyonoff
index 'catalog': query 'ENG ': returned 2 matches of 2 total in 0.000 sec
displaying matches:
1. document=8, weight=1, assembly=5, pc=7
    id=8
    partno=ENG088
    description=CPU
    price=55
2. document=9, weight=1, assembly=5, pc=3
    id=9
    partno=F23
    description= RAM
    price=65
words:
1. 'eng': 2 documents, 2 hits
$ /usr/local/bin/search --config /usr/local/etc/sphinx.conf wind
Sphinx 0.9.7
Copyright (c) 2001-2007, Andrew Aksyonoff
index 'catalog': query 'wind ': returned 2 matches of 2 total in 0.000 sec
displaying matches:
1. document=1, weight=1, assembly=3, pc=1
    id=1
    partno=S408
    description= GPU
    price=423
2. document=5, weight=1, assembly=3, pc=1
    id=5
    partno=WIN958
    description= Motherboard
    price=500
words:
1. 'wind': 2 documents, 2 hits
$ /usr/local/bin/search \
--config /usr/local/etc/sphinx.conf --filter model 3 ENG
Sphinx 0.9.7
Copyright (c) 2001-2007, Andrew Aksyonoff
index 'catalog': query 'ENG ': returned 1 matches of 1 total in 0.000 sec
displaying matches:
1. document=9, weight=1, assembly=5, pc=3
    id=9
    partno=ENG976
    description=Large cylinder head
    price=65
words:
1. 'eng': 2 documents, 2 hits
```

После всего этого можно приступать к написанию кода PHP для вызова поисковой системы Sphinx. API-интерфейс Sphinx для PHP небольшой и очень простой. Приложение PHP будет вызывать searchd и извлекать те же результаты, что и последняя команда показанная выше. Листинг вызова поисковой системы из PHP:

```
<?php
include('sphinx-0.9.7/api/sphinxapi.php');
$cl = new SphinxClient();
$cl->SetServer( "localhost", 3312 );
$cl->SetMatchMode( SPH_MATCH_ANY );
$cl->SetFilter( pc, array( 3 ) );
$result = $cl->Query( 'cpu', 'catalog' );
if ( $result === false ) {
    echo "Query failed: " . $cl->GetLastError() . ".\n"; }
else {
    if ( $cl->GetLastWarning() ) {
        echo "WARNING: " . $cl->GetLastWarning() . ""; }
    if ( ! empty($result["matches"]) ) {
        foreach ( $result["matches"] as $doc => $docinfo ) {
            echo "$doc\n"; }
        print_r( $result ); } }
exit;?>
```

В Sphinx реализовано множество полезных функций. В этой статье были описаны только общие моменты и возможности, и создан реальный работающий пример.

References:

1. (2012) Steve Francia MongoDB and PHP – O'Reilly Media.
2. (2012) Vaswani Vikram Zend Framework: A Beginner's Guide – Piter.
3. Koterov Dmitri (2013) PHP 5 – BHV-Peterburg.
4. Baron Schwartz, Peter Zaitsev, Vadim Tkachenko (2012) High Performance MySQL: Optimization, Backups, Replication, and More.
5. (2011) Brett McLaughlin PHP & MySQL: The Missing Manual.
6. Jonathan Gennick (2010) SQL Pocket Guide – O'Reilly Media.
7. Charles Bell, Mats Kindahl, Lars Thalmann (2010) - MySQL High Availability: Tools for Robust Data Centers – O'Reilly Media.
8. Andrew Curioso, Ronald Bradford, Patrick Galbraith (2010) Expert PHP and MySQL – O'Reilly Media.
9. Larry Ullman (2015) Effortless E-Commerce with PHP and MySQL
10. (2015) Brett McLaughlin MySQL Manual – O'Reilly Media.