**Вступление**

Данная библиотека предназначена для работы с RT-индексами поискового движка Sphinx посредством SphinxQL.

Поддерживаемые операции:

* Вставка
* Замена
* Обновление
* Удаление
* Очистка
* Оптимизация
* Транзакции (в ручном и автоматическом режимах)

*\* Для поиска используйте* [*Sphinx API*](http://sphinxsearch.com/wiki/doku.php?id=php_api_docs)

*\* Работа библиотеке проверялась на Sphinx 2.1.1. Могут возникнуть проблемы при работе с более старыми версиями поискового движка. Метод truncate будет работать только на версии 2.1.1 и выше. В таком случае, используйте метод deleteAll.*

**Установка и использование**

* Скопируйте файл Sphinxrt.class.php в директорию проекта;
* Подключите файл: (require\_ince(‘путь\_до\_класса/Sphinxrt.class.php’));
* Инициализируйте класс ($sphinxrt = new Sphinxrt()).

**Основные методы**

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| insert($index, $data) | Вставляет данные в указанный индекс. |
| replace($index, $data) | Заменят данные в указанном индексе. |
| update($index, $data, $where) | Обновляет данные в указанном индексе. Метод update позволяет обновлять только атрибуты типов: int, bigint, float, MVA. То есть атрибуты, которые объявлены в конфигурационном файле как: rt\_attr\_uint, rt\_attr\_bigint, rt\_attr\_float, rt\_attr\_multi, rt\_attr\_multi\_64, rt\_attr\_timestamp. Это специфика Sphinx’а. Так же имеются проблемы с точностью при обновлении полей типа float. Например: число 56.8 преобразовывается в 56.799999. При передаче целого числа или числа с нулями после запятой в float поле, в индексе будет 0.000000. Проблем в последнем случае можно избежать, если передавать число в виде строки с «000001» после запятой. Убедитесь, что этот метод Вам действительно нужен перед тем, как его использовать. |
| delete($index, $ids) | Удаляет запись/записи из указанного индекса. |
| truncate($index) | Полностью очищает данные в указанном индексе. |
| deleteAll($index) | Удаляет все данные из указанного индекса |
| optimize($index) | Производит оптимизацию указанного индекса. |
| errorMessage() | Возвращает сообщение об ошибке, возникшей во время выполнения последнего запроса. Если ошибки нет, метод вернёт пустую строку. |
| errorNumber() | Возвращает номер ошибки, возникшей во время выполнения последнего запроса. Если ошибки нет, метод вернёт 0. |

**Описание аргументов**

|  |  |  |
| --- | --- | --- |
| **Аргумент** | **Тип** | **Описание** |
| $index | string | Имя индекса, указанное в конфигурационном файле Sphinx’а.  Например: news\_rt. |
| $data | array | Массив данных, представляющий собой пары ключ – значение. Где ключ – это поле в таблице. Например:  array(  'author\_id' => 1,  'title' => 'New title'  );  Можно осуществлять множественные вставки и замены данных. Для этого в методы «insert» и «replace» необходимо передавать массив таких параметров.  Например:  array(  array(  'author\_id' => 1,  'title' => 'New title'  ),  array(  'author\_id' => 2,  'title' => 'New title 2'  )  ); |
| $where | array | Массив условий выборки. Посмотреть подробное описание параметра. |
| $ids | integer или array | Условия выборки для команд, поддерживающих работу только с полем id (оператор DELETE). Параметр может быть идентификатором записи или массивом идентификаторов.  Например:  8  array(8, 9, 10); |

**Параметр указания условий (where)**

Массив условий представляет собой пары ключ-значение, например:

// Выбрать записи, у которых category\_id = 1

array(

‘category\_id’ => 1

);

Ключ данного параметра может содержать операторы: =, !=, >, <, <>, >=, <=, IN, NOT IN, MATCH.

При использовании оператора IN или NOT IN в качестве значения необходимо указывать массив, например:

array(

‘category\_id IN => array(2, 3, 8)

);

При использовании оператора MATCH в качестве значения необходимо указывать массив, содержащий необходимые ключевые слова, например:

array(

// Все ключевые слова в одном элементе массива

'MATCH' => array('office ipad')

);

**Методы работы с транзакциями**

|  |  |
| --- | --- |
| **Метод** | **Описание** |
| transBegin() | Начинает транзакцию |
| transStatus() | Позволяет узнать, есть ли ошибки в текущей транзакции. Возвращает true, если ошибок нет, и false в обратном случае. |
| transCommit() | Автоматическое завершение транзакции. Данный метод зафиксирует изменения, если транзакция не содержит ни одной ошибки. В противном случае, изменения сохранены не будут. Метод так же возвращает true или false в случае успеха или неудачи соответственно. |
| transRollback() | Откатывает все изменения, совершённые в текущей транзакции. |
| transComplete() | Применяет все изменения, совершённые в текущей транзакции. |

*\* Транзакции не работают с методами update и truncate (специфика Sphinx’а).*

**Примеры использования**

Рассмотрим работу библиотеки на примере работы с некими новостями.

Структура таблицы:

CREATE TABLE IF NOT EXISTS `news` (

`id` bigint(20) unsigned NOT NULL AUTO\_INCREMENT,

`author\_id` bigint(20) unsigned NOT NULL,

`category\_id` tinyint(3) unsigned NOT NULL,

`title` varchar(100) NOT NULL,

`junk\_field` tinyint(1) unsigned DEFAULT '1',

`text` text NOT NULL,

`created\_at` int(11) unsigned NOT NULL,

`views` bigint(20) NOT NULL DEFAULT '0',

`rating` float unsigned NOT NULL DEFAULT '0',

PRIMARY KEY (`id`)

) ENGINE=MyISAM DEFAULT CHARSET=utf8;

Пример конфигурации индекса для Sphinx’а:

index news\_rt

{

type = rt

path = C:/WebServer/Sphinx/data/news/news/news

rt\_field = title

rt\_field = text

rt\_attr\_uint = author\_id

rt\_attr\_uint = category\_id

rt\_attr\_uint = views

rt\_attr\_timestamp = created\_at

rt\_attr\_string = title

rt\_attr\_float = rating

docinfo = extern

morphology = stem\_enru

min\_word\_len = 1

html\_strip = 1

charset\_type = utf-8

enable\_star = 1

rt\_mem\_limit = 64M

}

*Структура таблицы отличается от структуры индекса. Например, индекс не содержит поля «junk\_field». Это сделано для того, чтобы продемонстрировать небольшую особенность библиотеки. При передаче данных в методы, пользователю нет необходимости их отфильтровывать, можно передавать полный набор данных. Библиотека сама определит необходимые поля. Это касается и метода «update» - библиотека выберет из набора данных атрибуты нужных типов и обновит только их. (Для обновления не только атрибутов, но и полей используйте «replace».)*

*Следующие примеры могут не иметь практической ценности. Они приведены лишь для демонстрации работы библиотеки.*

*\* $news – массив новостей, полученных из базы данных.*

Добавим все новости:

$sphinxrt->insert('news\_rt', $news);

Добавим в индекс 4-ю новость:

$sphinxrt->insert('news\_rt', $news[3]);

Установим 5-ой новости 1000 просмотров:

$sphinxrt->update('news\_rt', array('views' => 1000), array('id' => 5));

Удалим 8-ю новость:

$sphinxrt->delete('news\_rt', 8);

Удалим 2-ю, 5-ю и 7-ю новости:

$sphinxrt->delete('news\_rt', array(2, 5, 7));

Рассмотрим пример успешной транзакции (изменения будут сохранены):

$sphinxrt->transBegin();

$sphinxrt->insert('news\_rt', $news[5]);

$sphinxrt->delete('news\_rt', 8);

$sphinxrt->transComplete();

Рассмотрим пример ошибочной транзакции (изменения не будут сохранены):

// Сначала добавим отдельную запись, например 5

// Затем попытаемся вставить все записи

// Первые четыре записи вставятся, но при вставке пятой записи возникнет ошибка: duplicate id '5'

// Проверив статус транзакции, можно откатить изменения и в индексе останется лишь 5-я запись

$sphinxrt->insert('news\_rt', $news[4]);

$sphinxrt->transBegin();

$sphinxrt->insert('news\_rt', $news);

if ($sphinxrt->transStatus())

{

$sphinxrt->transCommit();

}

else

{

$sphinxrt->transRollback();

}