基于MySQL全文索引的简单搜索引擎实现

**摘要：**

**目录：**

**1 简介**

**2 全文索引的类型**

**2.1 自然语言搜索**

**2.2 布尔搜索**

**2.3 查询扩展搜索**

**3 配置**

**4 测试使用innoDB创建全文索引**

**4.1 创建全文索引**

**4.1.1 创建表的同时创建全文索引**

**4.1.2 通过ALTER TABLE的方式来添加**

**4.1.3 通过CREATE INDEX的方式来添加**

**4.2 删除全文索引**

**4.2.1 使用DROP INDEX的方式**

**4.2.2 使用ALTER TABLE的方式**

**4.3 使用全文索引**

**4.3.1 全文索引的基本单位是词**

**4.3.2 MySQL有关全文索引的几个变量**

**5 简单搜索引擎代码实现**

**5.1 完整实现代码**

**5.2 代码分析**

**6 问题与结论**

**正文**

**1 简介**

本文侧重描述结合php、html并基于MySQL全文索引的简单搜索引擎实现，使用php、html语言完成页面代码与结果输出代码框架的搭建，使用MySQL完成基于数据库的搜索功能的实现。常用的搜索引擎往往实现模糊查询，最符合受众的搜索结果范围。一般情况下，使用where…like%...语句进行搜索，like关键字也常常能够符合需求。但是，当列的内容非常大的时候，like的性能不太乐观，因为它并不能保证每次查询都能用上索引。此时就可以使用全文索引。全文索引的对象是一个全文的集合，MySQL能够将全文索引的多列拼接成一个字符串，然后再进行索引。

**2 全文索引的类型**

MySQL索引可以分为：主键索引、普通索引、唯一索引、全文索引。其中，全文索引是比较特殊的，它只有少数的几个储存引擎支持，且只有类型为char、vchar、text的列能建立全文索引。全文索引是特殊的B+ Tree结构，第一层是所有的关键字，第二层则是每个关键字的一组指针。通俗来讲，全文索引结构是以关键字去找文档（行）的索引结构，而非像其他一些索引以主键来找其他列的内容。

全文索引有三种类型：自然语言搜索、布尔全文搜索和查询扩展搜索。

这里简要举例，具体介绍于后文详述。首先进行全文索引，语句如下：

Creat fulltext index 索引名 on 表名（列）

使用语句：

Where match(列) against (‘A B’);匹配A B

如果要使用布尔查询或短语查询，由于全文索引无法判断是否精确匹配了短语，需要回表过滤：

Where match(列) against (‘“A”’);//在单引号里用双引号包裹一个短语，让返回结果精确匹配指定的短语。

Where match(列) against (‘+A -B’IN BOOLEAN MODE);//返回结果必须含有A，但不能含有B

Where match(列) against (‘>A <B’ IN BOOLEAN MODE);//含有A优先级升高，含有B优先级降低。

也可以把全文匹配的结果返回，返回的浮点数表示这一行关于这些词语的匹配度：

Select id, match(列) against(‘A’) as factor from 表名 where…// 返回每行内容对A的匹配度。

**2.1 自然语言搜索（natural language search）**

自然语言搜索是一种使用会话式语言引导搜索过程的搜索方法。通过MATCH AGAINST传递某个特定的字符串进行检索，此为默认方式，更加简单方便。使用自然语言搜索的结果会比用户预期的结果更多一些。

**2.2 布尔全文搜索(Boolean search)**

布尔查询是数据库检索最基本的方法，是用逻辑“或”、“与”“非”等算符在数据库中对相关文献的定性选择的方法。为检索的字符串增加操作符，如“+”（或）表示必须包含，“-”（非）表示不包含，“\*”（与）表示通配符，即使传递的字符串较小或出现在词语中，也不会被过滤掉。

**2.3 查询扩展搜索(query expansion search)**

为了改善资讯检索召回率，而将原来查询句增加新的关键词来重新查询，此查询方式称为扩展查询。搜索引擎会将使用者输入的查询句先做一次检索，根据检索出来的文件，选取出适合的关键字，加到查询句重新检索，借此来找出更多相关文件。搜索字符串用于执行自然语言搜索，然后，搜索返回的最相关行的单词被添加到搜索字符串，并且再次进行搜索，查询将返回来自第二个搜索的行。

**3 配置**

XAMPP（Apache+MySQL+PHP+PERL）作为一个功能强大的建站集成软件包、服务器系统开发套件，十分方便安装和使用。同时它还包含了管理MySQL的工具phpMyAdmin，即可对MySQL进行可视化操作。采用这种紧密的集成，XAMPP可以运行任何程序：从个人主页到功能全面的产品站点。配置过程中可以对虚拟主机配置文件基于ip等对其进行修改。

**4 测试使用innoDB创建全文索引**

**4.1 创建全文索引**

在创建表的时候，可以在CREATE TABLE语句中给出FULLTEXT索引定义，或者稍后使用ALTER TABLE或CREATE INDEX添加该定义。

**4.1.1 创建表的同时创建全文索引**

CREATE TABLE IF NOT EXISTS `termbase` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`term` varchar(45) NOT NULL,

`contents` TEXT FULLTEXT(term,content),

`User` varchar(45) NOT NULL,

`upTime` varchar(45) NOT NULL,

`category` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=8 TYPE=MYISAM;

**4.1.2 通过ALTER TABLE的方式来添加**

ALTER TABLE table\_name ADD FULLTEXT INDEX index\_name(column1,column2,…)

或者：

ALTER TAVLE table\_name ADD FULLTEXT index\_name(column1,column2,…)

**4.1.3 通过CREATE INDEX的方式来添加**

CREATE FULLTEXT INDEX index\_name ON table\_name (column1, column2, column3,…);

也可以在创建索引的时候指定索引的长度：

CREATE FULLTEXT INDEX index\_name ON table\_name (column1(20));

**4.2 删除全文索引**

**4.2.1 使用DROP INDEX的方式(没有drop fulltext index 这种用法)**

DROP INDEX index\_name ON table\_name;

**4.2.2 使用ALTER TABLE的方式**

ALTER TABLE table\_name DROP INDEX index\_name;

**4.3 使用全文索引**

全文索引的格式为：MATCH (columnName) AGAINST(‘string’)

如：SELECT \* FROM ‘student’ WHERE MATCH (‘name’) AGAINST (‘聪’)

当查询多列数据时：

建议在此多列数据上建立一个联合的全文索引，否则使用不了索引的。

如：SELECT \* FROM ‘student’ WHERE MATCH (‘name’,’address’) AGAINST(‘聪 广东’)

**4.3.1 全文索引的基本单位是词**

全文索引是以词为基础的，MySQL默认的分词是所有非字母和数字特殊符号都是分词符。

**4.3.2 MySQL有关全文索引的几个变量**

控制全文索引的参数都是以ft（FullText）开头的：show variables like “ft%”；结果如下：

ft\_boolean\_syntax:表示布尔查询的时候可以用的符号。改变IN BOOLEAN MODE的查询字符，不用重新启动MySQL也不用重建索引。

ft\_max\_word\_len:最长的索引字符串，默认值84，修改后要重建索引。

ft\_min\_word\_len:最长的索引字符串，默认值4，修改后要重建索引。

ft\_query\_expansion\_limit:查询扩展时取最相关的几个值用作二次查询。

ft\_stopword\_file(built-in):停词文件，这个文件里的词查询时会忽略掉。

**5 简单搜索引擎代码实现**

**5.1完整实现代码**

**创建表**

CREATE TABLE IF NOT EXISTS `termbase` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`term` varchar(45) NOT NULL,

`contents` TEXT FULLTEXT(term,content),

`User` varchar(45) NOT NULL,

`upTime` varchar(45) NOT NULL,

`category` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=8 TYPE=MYISAM;

**conn.php**

<?php

$dbhost = '127.0.0.1:3307';

$dbuser = 'root';

$dbpass = 'root';

$conn = mysqli\_connect($dbhost, $dbuser, $dbpass);

if(! $conn )

{

die('Could not connect: ' . mysqli\_error());

}

mysqli\_select\_db($conn,"database");

header("Content-Type: text/html;charset=utf-8");

mysqli\_set\_charset($conn,'utf8');

?>

**Index.php**

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>搜索引擎</title>

</head>

<body>

<form action="result.php" method="post">

<table border=1 title="SEARCH" style='margin:0 auto;'>

<tr>

<td>

<button type="submit">搜索</button>

</td>

<td><input type="text" name="searchterm" placeholder="请输入你想查找的内容" />

</tr>

</table>

</form>

**result.php**

<html>

<head>

<meta charset="utf-8">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>查询结果显示</title>

</head>

<body>

<?php

include('conn.php');

mysqli\_query($conn,"set names utf8");

$searchterm = $\_POST["searchterm"];

$sql = "SELECT \* FROM `termbase` WHERE MATCH `contents` AGAINST '%{$searchterm}%'";

mysqli\_query($conn,"set names utf8");

$result = mysqli\_query($conn,$sql);

echo "<table border='1'>

<tr>

<td>条目</td>

<td>内涵</td>

<td>上传用户</td>

<td>上传时间</td>

<td>分类</td>

</tr>";

while($row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC))

{

echo "<tr><td>".$row["item"]."</td>";

echo "<td>".$row["contents"]."</td>";

echo "<td>".$row["User"]."</td>";

echo "<td>".$row["upTime"]."</td>";

echo "<td>".$row["category"]."</td>";

echo mysqli\_error($conn);

}

echo "</table>";

?>

</body>

</html>

**5.2 代码分析**

基于MySQL全文索引的简单搜索引擎的整体实现代码包括三个代码文件，其中包括页面文件（表单），功能实现文件（查询与结果显示）与数据库连接文件。

**创建表**

CREATE TABLE IF NOT EXISTS `termbase` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`term` varchar(45) NOT NULL,

`contents` TEXT FULLTEXT(term,content),

`User` varchar(45) NOT NULL,

`upTime` varchar(45) NOT NULL,

`category` varchar(45) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO\_INCREMENT=8 TYPE=MYISAM;

//创建表的同时创建全文索引

**conn.php：单独数据库连接文件，实现代码复用**

<?php

$dbhost = '127.0.0.1:3307'; //mysql服务器主机地址

$dbuser = 'root'; //mysql用户名

$dbpass = '';//mysql用户名密码

$conn = mysqli\_connect($dbhost, $dbuser, $dbpass);//连接数据库

if(! $conn )

{

die('Could not connect: ' . mysqli\_error());

}

mysqli\_select\_db($conn,"database");//修改默认连接数据库为“database”

header("Content-Type: text/html;charset=utf-8");

mysqli\_set\_charset($conn,'utf8');

?>

**Index.php**

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>搜索引擎</title>

</head>

<body>

<form action="result.php" method="post">

<table border=1 title="SEARCH" style='margin:0 auto;'>

<tr>

<td>

<button type="submit">搜索</button>

</td>

<td><input type="text" name="searchterm" placeholder="请输入你想查找的内容" />

</tr>

</table>

</form>

//页面文件，通过表单获取查询的内容并传到result.php中进行功能实现

**result.php**

<html>

<head>

<meta charset="utf-8">

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>查询结果显示</title>

</head>

<body>

<?php

include('conn.php');//使用数据库连接文件进行数据库连接

mysqli\_query($conn,"set names utf8");

$searchterm = $\_POST["searchterm"];

$sql = "SELECT \* FROM `termbase` WHERE MATCH `contents` AGAINST '%{$searchterm}%'"; mysqli\_query($conn,"set names utf8");

//使用全文索引获得所需内容行

$result = mysqli\_query($conn,$sql);

echo "<table border='1'>

<tr>

<td>条目</td>

<td>内涵</td>

<td>上传用户</td>

<td>上传时间</td>

<td>分类</td>

</tr>";

while($row = mysqli\_fetch\_array($result, MYSQLI\_ASSOC))//获取行内容

{

echo "<tr><td>".$row["item"]."</td>";

echo "<td>".$row["contents"]."</td>";

echo "<td>".$row["User"]."</td>";

echo "<td>".$row["upTime"]."</td>";

echo "<td>".$row["category"]."</td>";

echo mysqli\_error($conn);

}

echo "</table>";

//对结果行指定列的内容进行输出

?>

</body>

</html>

**6 问题与结论**

全文索引依旧存在问题，性能在某些方面不如普通索引。全文索引占有的储存空间更大，如果内存一次装不下全部索引，性能会非常差。增删改的代价更大，修改文本中几个单词就要操作维护索引几次，在这一点上相比之下，普通索引更便捷一些。如果一个列上有全文索引则一定会用上，即使有其他索引能够性能与效果更好也不会使用。但全文索引因为用了索引，性能更高并且灵活性强，由词库支撑的话可以进行分词，提供一些语义查询的功能，由词语停用表则可以忽略某些词语，词语的最大、最小值的设置可以是结果更加贴近受众期望。