# 1.创建全文索引(FullText index)

       旧版的MySQL的全文索引只能用在MyISAM表格的char、varchar和text的字段上。

       不过新版的MySQL5.6.24上InnoDB引擎也加入了全文索引，所以具体信息要随时关注官网，

## 1.1. 创建表的同时创建全文索引

             CREATE TABLE article (   
                  id INT AUTO\_INCREMENT NOT NULL PRIMARY KEY,   
                  title VARCHAR(200),   
                  body TEXT,   
                  **FULLTEXT(title, body)**  
              ) TYPE=MYISAM; 

## 1.2.通过 alter table 的方式来添加

                ALTER TABLE `student` ADD FULLTEXT INDEX ft\_stu\_name  (`name`) #ft\_stu\_name是索引名，可以随便起

       或者：ALTER TABLE `student` ADD FULLTEXT ft\_stu\_name  (`name`)

## 1.3. 直接通过create index的方式

                CREATE **FULLTEXT INDEX** ft\_email\_name ON `student` (`name`)

           也可以在创建索引的时候指定索引的长度：

                CREATE **FULLTEXT INDEX** ft\_email\_name ON `student` (`name`**(20)**)

# 2. 删除全文索引

## 2.1. 直接使用 drop index(注意：没有 drop fulltext index 这种用法)

                 DROP INDEX full\_idx\_name ON tommy.girl ;

## 2.2. 使用 alter table的方式

                ALTER TABLE tommy.girl DROP INDEX ft\_email\_abcd;

# 3.使用全文索引

     跟普通索引稍有不同

     使用全文索引的格式：  MATCH (columnName) AGAINST ('string')

     eg:

           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创建一个联合的全文索引，否则使用不了索引的。

          SELECT \* FROM `student` WHERE **MATCH**(`name`,`address`) **AGAINST**('聪 广东')

## 3.1. 使用全文索引需要注意的是：(基本单位是词)

            分词，全文索引以词为基础的，MySQL默认的分词是**所有非字母和数字的特殊符号都是分词符**(外国人嘛)

            这里推荐一篇文章：[利用mysql的全文索引实现模糊查询](http://blog.chinaunix.net/uid-24145780-id-125161.html)

       3.2. MySQL中与全文索引相关的几个变量：

             使用命令：mysql> SHOW VARIABLES LIKE 'ft%'; #ft就是FullText的简写  
  
             ft\_boolean\_syntax    **+ -><()~\*:""&|**         #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\_min\_word\_len    4                                   #最短的索引字符串，默认值为4，(通常改为1)修改后必须重建索引文件

                                                                               重新建立索引命令：**repair table tablename quick**

             ft\_max\_word\_len    84                                #最长的索引字符串，默认值为84，修改后必须重建索引文件

             ft\_query\_expansion\_limit   20                      #查询括展时取最相关的几个值用作二次查询

             ft\_stopword\_file    **(built-in)**                      #全文索引的过滤词文件，具体可以参考：[MySQL全文检索中不进行全文索引默认过滤词](http://ourmysql.com/archives/562" \t "_blank)

             特别注意：50%的门坎限制(当查询结果很多，几乎所有记录都有，或者极少的数据，都有可能会返回非所期望的结果)

                       -->可用**IN BOOLEAN MODE**即可以避开50%的限制。

                      此时使用全文索引的格式就变成了： **SELECT \* FROM `student` WHERE MATCH(`name`) AGAINST('聪' IN BOOLEAN MODE)**

                        更多内容请参考：[MySQL中的全文检索(1)](http://blog.csdn.net/navygong/article/details/4185769)

# 4. ft\_boolean\_syntax (+ -><()~\*:""&|**)**使用的例子：

## 4.1  + : 用在词的前面，表示一定要包含该词，并且必须在开始位置。

                            eg: +Apple 匹配：Apple123,     "tommy, Apple"

## 4.2  - : 不包含该词，所以不能只用「-yoursql」这样是查不到任何row的，必须搭配其他语法使用。

                            eg: MATCH (girl\_name) AGAINST ('-林志玲 +张筱雨')

                              匹配到： 所有不包含林志玲，但包含张筱雨的记录

## 4.3. 空(也就是默认情况)，表示可选的，包含该词的顺序较高。

                       例子：

**apple banana**           找至少包含上面词中的一个的记录行

**+apple +juice**               两个词均在被包含

**+apple macintosh**    包含词 “apple”，但是如果同时包含 “macintosh”，它的排列将更高一些

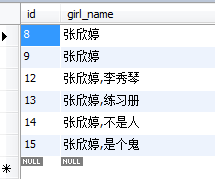
**+apple -macintosh**包含 “apple” 但不包含 “macintosh”

## 4.4. > :提高该字的相关性，查询的结果会排在比较靠前的位置。

## 4.5.< :降低相关性，查询的结果会排在比较靠后的位置。

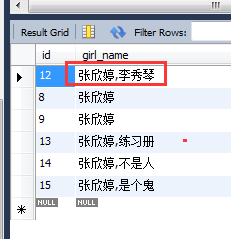
                      例子：4.5.1.先不使用 ><

**select \* from tommy.girl where match(girl\_name) against('张欣婷' in boolean mode);**

**   可以看到完全匹配的排的比较靠前**

                              4.5.2. 单独使用 >

**select \* from tommy.girl where match(girl\_name) against('张欣婷 >李秀琴' in boolean mode);**

**  使用了>的李秀琴马上就排到最前面了**

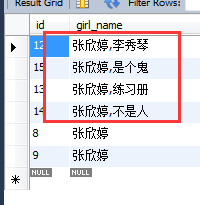
                             4.5.3. 单独使用 <

**select \* from tommy.girl where match(girl\_name) against('张欣婷 <不是人' in boolean mode);**

** 看到没，不是人也排到最前面了，这里使用的可是 < 哦，说好的降低相关性呢，往下看吧。**

                           4.5.4.同时使用><

**select \* from tommy.girl where match(girl\_name) against('张欣婷 >李秀琴 <练习册 <不是人 >是个鬼' in boolean mode);**

** 到这里终于有答案了，只要使用了 ><的都会往前排，而且>的总是排在<的前面**

**小结一下：1. 只要使用 ><的总比没用的 靠前；**

**2. 使用  >的一定比 <的排的靠前 (这就符合相关性提高和降低)；**

**3. 使用同一类的，使用的越早，排的越前。**

# 4.6. ( )：可以通过括号来使用字条件。

                       eg: +aaa +(>bbb <ccc) // 找到有aaa和bbb和ccc，aaa和bbb，或者aaa和ccc(因为bbb，ccc前面没有+，所以表示可有可无)，

                                         然后 aaa&bbb **>** aaa&bbb&ccc **>** aaa&ccc

## 4.7. ~ :将其相关性由正转负，表示拥有该字会降低相关性，但不像「-」将之排除，只是排在较后面。

                           eg:   +apple ~macintosh   先匹配apple，但如果同时包含macintosh，就排名会靠后。

## 4.8. \* :通配符，这个只能接在字符串后面。

                                 MATCH (girl\_name) AGAINST ('+\*ABC\*')   #错误，不能放前面

                                 MATCH (girl\_name) AGAINST ('+张筱雨\*')  #正确

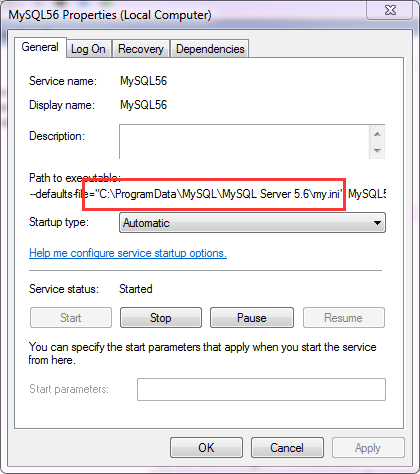
## 4.9. " " :整体匹配，用双引号将一段句子包起来表示要完全相符，不可拆字。

                                 eg:  "tommy huang" 可以匹配  **tommy huang** xxxxx   但是不能匹配 **tommy** is **huang**。

# 5.补充：Windows下无法修改 ft\_min\_word\_len的情况，

## 5. 1. 使用cmd打开 services.msc，

                找到你的 MySQL服务，右键Properties,找到你的my.ini所在的路径



## 5.2. 停止MySQL,在my.ini中增加 ft\_min\_word\_len = 1,重启MySQL,

                    然后使用命令 **show variables like 'ft\_min\_word\_len';** 查看是否生效了