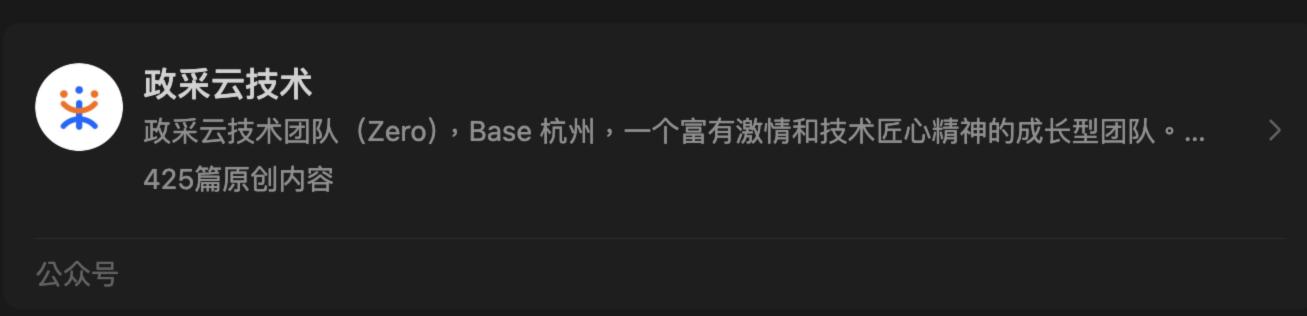
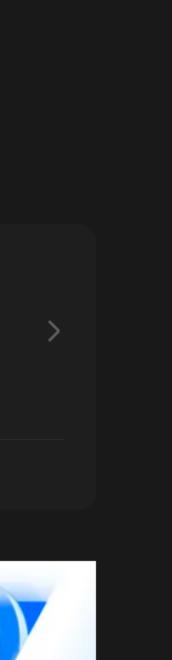
MySQL 全文索引

原创 乔木 政采云技术 2024年01月17日 09:01 浙江





微信扫一扫

关注该公众号



乔木 《 优秀创作者 》

后端开发工程师/如果生活一切如你所愿,那人生将到

1、背景简介

实际开发过程中,我们经常会遇到全文检索的述求,一般都会采用搭建ES服务器来实现。但因为数据量较少,并且不属于高并发高吞吐场景,相比较而言接入 ES,不仅会使得系统设计更加复杂,还会产生资源浪费,所以需要采用更加简单且廉价的方案来实现。一般互联网公司都会用到MySQL 服务,从 MySQL5.7 开始,MySQL 内置了 ngram 全文检索插件,用来支持中文分词,并且对 MyISAM 和InnoDB 引擎有效。因此可以通过 MySQL 服务接入 full-text 索引来实现简单地全文检索需求。

2、MySQL 全文索引简介

MySQL 的全文索引主要用于全文字段的检索场景,支持 char、varchar、text 几种字段加全文索引,仅支持 InnoDB 与 MyISAM 引擎。MySQL 内置了 ngram 解析器来支持中文、日文、韩文等语言的文本。MySQL 全文索引支持三种模式:● 布尔模式(IN BOOLEAN MODE);● 自然语言模式(NATURAL LANGUAGE MODE);● 查询拓展(QUERY EXPANSION);

3、 ngram 解析器简介

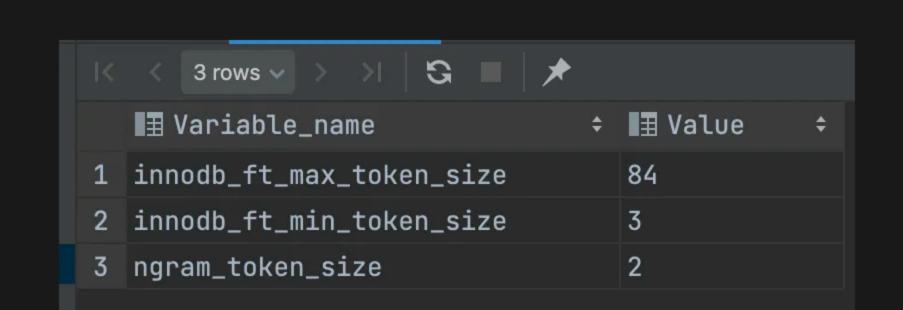
ngram 一种基于统计语言模型的算法,简单来说,就是通过一个大小为 n 的滑动窗口,将一段文本分成多个由 n 个连续单元组成的term。其中 n 为分词大小默认为 2,可通过 ngram_token_size设置分词大小。示例:使用 ngram 对于"全文索引"进行分词。

ngram_token_size =1,分词为 '全','文','索','引';ngram_token_size =2,分词为 '全文','文索','索引';ngram_token_size =3,分词为 '全文索','文索引';ngram_token_size =4,分词为 '全文索引';

3.1、 如何查看配置 ngram_token_size

#查看默认分词大小 ngram_token_size = 2 show variables like '%token%';

查询结果:



innodb_ft_min_token_size:默认 3,表示最小 3 个字符作为一个关键词,增大该值可减少全文索引的大小 innodb_ft_max_token_size:默认 84,表示最大 84 个字符作为一个关键词,限制该值可减少全文索引的大小 ngram_token_size:默认 2,表示2个字符作为内置分词解析器的一个关键词,如对"abcd"建立全文索引,关键词为'ab','bc','cd' 当使用 ngram 分词解析器时,innodb_ft_min_token_size 和 innodb_ft_max_token_size 无效

3.2、 修改配置 ngram_token_size

第一种:mysqld --ngram_token_size = 1;第二种:在配置文件中 [mysqld]ngram_token_size = 1;不可动态修改,修改后需重启 MySQL 服务,并重新建立全文索引。

4、创建全文索引

1、创建表的同时创建全文索引

```
CREATE TABLE `announcement` (
   `id` int(11) NOT NULL AUTO_INCREMENT COMMENT '主键',
   `content` text CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL COMMENT '内容',
   `title` varchar(255) CHARACTER SET utf8mb4 COLLATE utf8mb4_general_ci NULL DEFAULT NULL CO
   PRIMARY KEY (`id`) USING BTREE,
   FULLTEXT INDEX `idx_full_text`(`content`) WITH PARSER `ngram`
   ) ENGINE = InnoDB AUTO_INCREMENT = 6 CHARACTER SET = utf8mb4 COLLATE = utf8mb4_general_ci RC
```

2、通过 alter table 的方式来添加

ALTER TABLE announcement ADD FULLTEXT INDEX idx_full_text(content) WITH PARSER ngram;

3、直接通过 create index 的方式

CREATE FULLTEXT INDEX idx_full_text ON announcement(content) WITH PARSER `ngram`;

5、全文索引测试

构建测试数据:

INSERT INTO announcement (id, content, title) VALUES (1, '杭州市最近有大雪,出门多穿衣服', '杭州天 INSERT INTO announcement (id, content, title) VALUES (2, '杭州市最近温度很低,不适合举办杭州马拉松 INSERT INTO announcement (id, content, title) VALUES (3, '杭州市最近有大雪,西湖断桥会很美', '杭州 INSERT INTO announcement (id, content, title) VALUES (4, '浙江大学的雪景也很美,周末可以去杭州逛逛 INSERT INTO announcement (id, content, title) VALUES (5, '城北万象城开业,打折力度很大', '城北万象

5.1、布尔模式(IN BOOLEAN MODE)

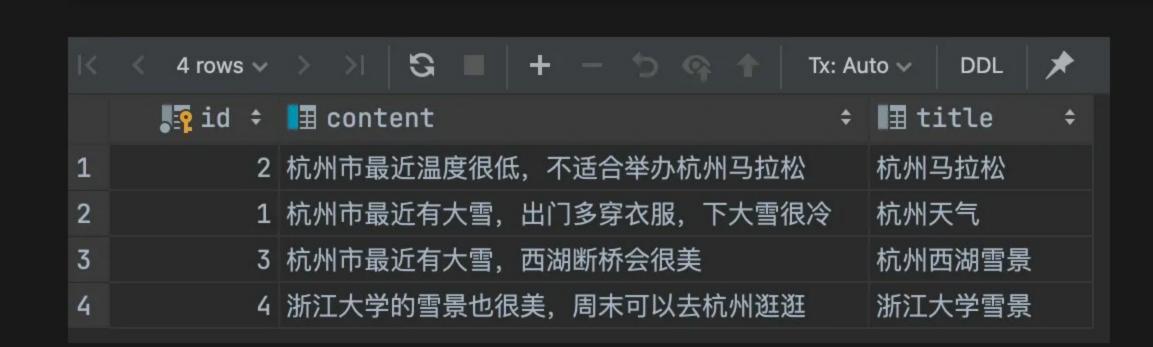
布尔模式的全文检索支持下面几种常用操作符:

+(必须出现) -(必须不出现) 无操作符(出现了,相关性会更高) < > (增加或者减少相关性) ~ (负相关性) *(通配符) "" (短语)

通过简单示例分别介绍布尔模式下几种操作符的具体用法:

1、操作符+(必须出现)

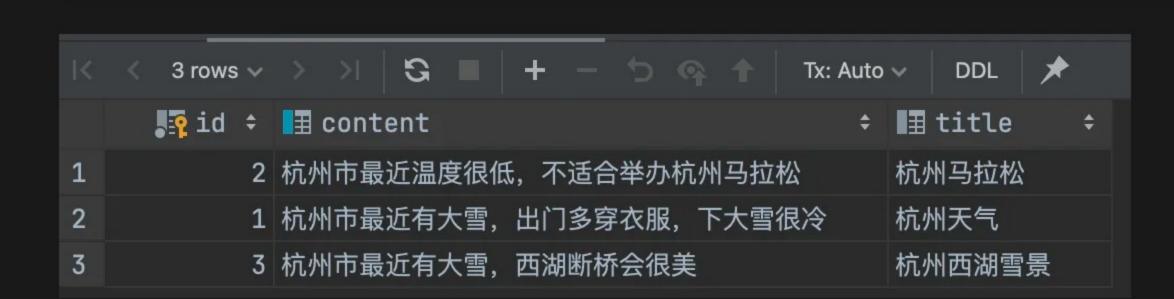
select * from announcement where MATCH (content) against ('+杭州' in Boolean MODE);



'+杭州'表示必须出现"杭州"这个分词,数据才能被检索到,并且包含杭州分词越多的代表着相关性更高。从结果可以看出,"杭州"这个分词出现次数最多的排在最前面。

2、 操作符-(必须不出现)

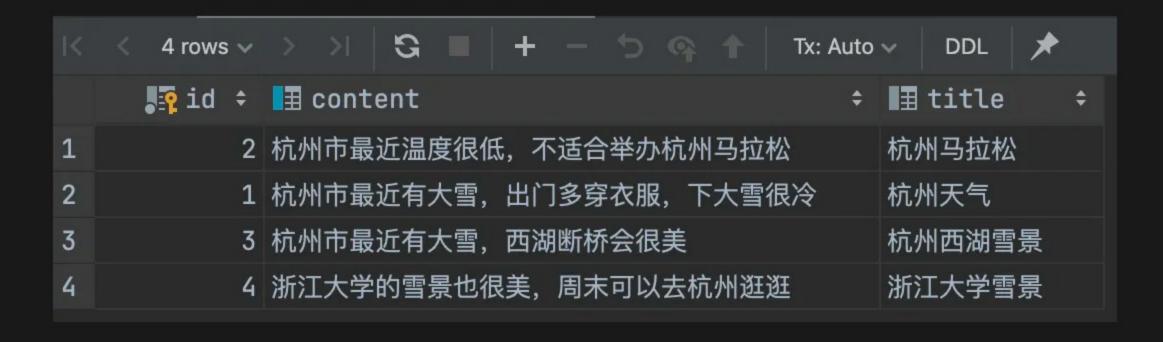
select * from announcement where MATCH (content) against ('+杭州 -大学' in Boolean MODE);



'+杭州 -大学'表示被检索到的数据必须包含"杭州"这个分词,-大学表示被检索到的数据必须不能包含"大学"这个分词。

3、 无操作符(出现了,相关性会更高)

select * from announcement where MATCH (content) against ('杭州 大雪' in Boolean MODE);



无操作符'杭州 大雪'表示出现"杭州"或者"大雪"的数据会有更高的相关性

4、<>(增加或者减少相关性)

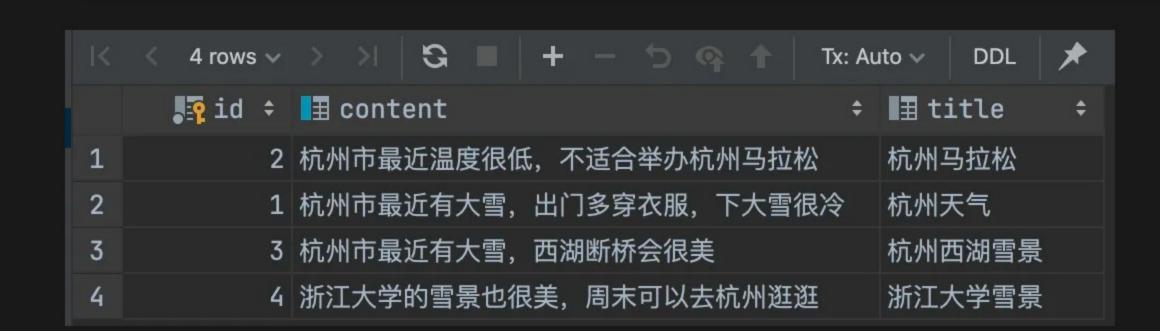
select * from announcement where MATCH (content) against ('+杭州 >大学' in Boolean MODE);



'+杭州 >大学'表示被检索到的数据必须包含"杭州"这个分词,当出现"大学"时对应数据的相关性会升高,如图可知带有大学的数据排序靠前。同理"<大学"表示当出现"大学"时对应数据的相关性会降低。

5、~(负相关性)

select * from announcement where MATCH (content) against ('+杭州 ~大学' in Boolean MODE);



'+杭州 ~大学'表示被检索到的数据必须包含"杭州"这个分词,当出现"大学"时对应数据的相关性会降低,如图可知带有大学的数据排序靠靠后,效果等同于'+杭州 -大学'。

6、*(通配符)

*操作符的作用其实与like的通配符类似

```
select * from announcement where MATCH (content) against ('杭州*' in Boolean MODE);
```

7、""(短语)

```
select * from announcement where MATCH (content) against ('"杭州"' in Boolean MODE);
```

双引号表示"杭州"以短语的方式被检索到,如果此时分词大小为1时,

5.2 自然语言模式

自然语言模式是默认全文检索模式,简单地说就是把检索关键词当做自然语言来处理,自然语言 模式也等价于布尔模式中的无操作符模式,下面三种查询,结果是一样的:

```
-- 自然语言模式
select * from announcement where MATCH (content) against ('杭州 大学' IN NATURAL LANGUAGE MOD
-- 布尔模式 无操作符
select * from announcement where MATCH (content) against ('杭州 大学' in Boolean MODE);
-- 默认模式
select * from announcement where MATCH (content) against ('杭州 大学');
```

5.3 拓展查询

拓展查询是对自然语言搜索的修改。搜索字符串用于执行自然语言搜索。然后,将搜索返回的最相关行中的单词添加到搜索字符串中,然后再次执行搜索。该查询返回第二个搜索中的行。

```
-- 首先根据'万象城'关键词 查询出 '城北'、'北万'、'万象'、'象城'、'城开'、'开业'、'打折'、'力度'、'度很'、select * from announcement where MATCH (content) against ('万象城' WITH QUERY EXPANSION);
-- 再根据 '城北'、'北万'、'万象'、'象城'、'城开'、'开业'、'打折'、'力度'、'度很'、'很大'等结果进行查询select * from announcement where MATCH (content) against ('城北 北万 万象 象城 城开 开业 打折 折力
```



6、总结

全文索引,通过建立倒排索引,可以极大的提升检索效率,解决判断字段是否包含的问题。但全文索引占有存储空间更大,如果内存一次装不下全部索引,性能会非常差。并且使用起来学习成本较高,如果没有合理的设置好分词大小等参数,会出现查询结果不尽人意的效果。

参考文献

1、https://dev.mysql.com/doc/refman/5.7/en/fulltext-boolean.html《Boolean Full-Text Searches》

看完两件事

如果你觉得这篇内容对你挺有启发,我想邀请你帮我两件小事

1.点个「**在看**」,让更多人也能看到这篇内容(点了「**在看**」,bug -1 😊)

2.关注公众号「**政采云技术**」,持续为你推送精选好文

招贤纳士

政采云技术团队(Zero),Base 杭州,一个富有激情和技术匠心精神的成长型团队。规模 500 人左右,在日常业务开发之外,还分别在云原生、区块链、人工智能、低代码平台、中间件、大数据、物料体系、工程平台、性能体验、可视化等领域进行技术探索和实践,推动并落地了一系列的内部技术产品,持续探索技术的新边界。此外,团队还纷纷投身社区建设,目前已经是 google flutter、scikit-learn、Apache Dubbo、Apache Rocketmq、Apache Pulsar、CNCF Dapr、Apache DolphinScheduler、alibaba Seata 等众多优秀开源社区的贡献者。

如果你想改变一直被事折腾,希望开始折腾事;如果你想改变一直被告诫需要多些想法,却无从破局;如果你想改变你有能力去做成那个结果,却不需要你;如果你想改变你想做成的事需要一个团队去支撑,但没你带人的位置;如果你想改变本来悟性不错,但总是有那一层窗户纸的模糊……如果你相信相信的力量,相信平凡人能成就非凡事,相信能遇到更好的自己。如果你希望参与到随着业务腾飞的过程,亲手推动一个有着深入的业务理解、完善的技术体系、技术创造价值、影响力外溢的技术团队的成长过程,我觉得我们该聊聊。任何时间,等着你写点什么,发给zcy-tc@cai-inc.com

服务器 3 # mysql 3 # 政采云技术 85

服务器・目录≡

く上一篇・前端本地化部署