链滴

Mysql 几种索引方式的区别及适用情况

作者：huihui
原文链接：https://ld246.com/article/1504165404542
来源网站：链滴

转自 董志峰Do的博客

Mysql目前主要有以下几种索引方式：FULLTEXT，HASH，BTREE，RTREE。

那么，这几种索引有什么功能和性能上的不同呢?

FULLTEXT

即为全文索引，目前只有MyISAM引擎支持。其可以在CREATE TABLE ，ALTER TABLE ，CREATE INDEX 使用，不过目前只有 CHAR、VARCHAR ，TEXT 列上可以创建全文索引。值得一提的是，在数量较大时候，现将数据放入一个没有全局索引的表中，然后再用CREATE INDEX创建FULLTEXT索引要比先为一张表建立FULLTEXT然后再将数据写入的速度快很多。

全文索引并不是和MyISAM一起诞生的，它的出现是为了解决WHERE name LIKE “%word%"这类对文本的模糊查询效率较低的问题。在没有全文索引之前，这样一个查询语句是要进行遍历数据表操的，可见，在数据量较大时是极其的耗时的，如果没有异步IO处理，进程将被挟持，很浪费时间，当这里不对异步IO作进一步讲解，想了解的童鞋，自行谷哥。

全文索引的使用方法并不复杂：

创建ALTER TABLE table ADD INDEX FULLINDEX USING FULLTEXT(cname1[,cname2...]);

使用SELECT * FROM table WHERE MATCH(cname1[,cname2...]) AGAINST ('word' MODE);

其中， MODE为搜寻方式（IN BOOLEAN MODE ，IN NATURAL LANGUAGE MODE ，IN NATURAL LANGUAGE MODE WITH QUERY EXPANSION / WITH QUERY EXPANSION）。

关于这三种搜寻方式，在这里也不多做交代，简单地说，就是，布尔模式，允许word里含一些特殊符用于标记一些具体的要求，如+表示一定要有，-表示一定没有，*表示通用匹配符，是不是想起了则，类似吧；自然语言模式，就是简单的单词匹配；含表达式的自然语言模式，就是先用自然语言模处理，对返回的结果，再进行表达式匹配。

对搜索引擎稍微有点了解的同学，肯定知道分词这个概念，FULLTEXT索引也是按照分词原理建立索的。西文中，大部分为字母文字，分词可以很方便的按照空格进行分割。但很明显，中文不能按照这方式进行分词。那又怎么办呢？这个向大家介绍一个Mysql的中文分词插件**Mysqlcft**，有了它，就可对中文进行分词，想了解的同学请移步Mysqlcft，当然还有其他的分词插件可以使用。

HASH

Hash这个词，可以说，自打我们开始码的那一天起，就开始不停地见到和使用到了。其实，hash就一种（key=>value）形式的键值对，如数学中的函数映射，允许多个key对应相同的value，但不允一个key对应多个value。正是由于这个特性，hash很适合做索引，为某一列或几列建立hash索引，会利用这一列或几列的值通过一定的算法计算出一个hash值，对应一行或几行数据（这里在概念上和数映射有区别，不要混淆）。在java语言中，每个类都有自己的hashcode()方法，没有显示定义的都承自object类，该方法使得每一个对象都是唯一的，在进行对象间equal比较，和序列化传输中起到很重要的作用。hash的生成方法有很多种，足可以保证hash码的唯一性，例如在MongoDB中，每个document都有系统为其生成的唯一的objectID（包含时间戳，主机散列值，进程PID，和自增ID也是一种hash的表现。额，我好像扯远了-_-!

由于hash索引可以一次定位，不需要像树形索引那样逐层查找,因此具有极高的效率。那为什么还需其他的树形索引呢?

在这里就不自己总结了。引用下园子里其他大神的文章：来自 14的路 的MySQL的btree索引和hash引的区别

（1）Hash 索引仅仅能满足"="，"IN"和"<=>"查询，不能使用范围查询。

由于 Hash 索引比较的是进行 Hash 运算之后的 Hash 值，所以它只能用于等值的过滤，不能用于基范围的过滤，因为经过相应的 Hash 算法处理之后的 Hash 值的大小关系，并不能保证和Hash运算完全一样。

（2）Hash 索引无法被用来避免数据的排序操作。

由于 Hash 索引中存放的是经过 Hash 计算之后的 Hash 值，而且Hash值的大小关系并不一定和 Has 运算前的键值完全一样，所以数据库无法利用索引的数据来避免任何排序运算；

（3）Hash 索引不能利用部分索引键查询。

对于组合索引，Hash 索引在计算 Hash 值的时候是组合索引键合并后再一起计算 Hash 值，而不是独计算Hash 值，所以通过组合索引的前面一个或几个索引键进行查询的时候，Hash 索引也无法被用。

（4）Hash 索引在任何时候都不能避免表扫描。

前面已经知道，Hash 索引是将索引键通过 Hash 运算之后，将 Hash运算结果的 Hash 值和所对应行指针信息存放于一个 Hash 表中，由于不同索引键存在相同 Hash 值，所以即使取满足某个 Hash 值的数据的记录条数，也无法从 Hash 索引中直接完成查询，还是要通过访问表中的实际数据进行相的比较，并得到相应的结果。

（5）Hash 索引遇到大量Hash值相等的情况后性能并不一定就会比B-Tree索引高。

对于选择性比较低的索引键，如果创建 Hash 索引，那么将会存在大量记录指针信息存于同一个 Hash 值相关联。这样要定位某一条记录时就会非常麻烦，会浪费多次表数据的访问，而造成整体性能低下。

我稍作补充，讲一下HASH索引的过程，顺便解释下上面的第4,5条：

当我们为某一列或某几列建立hash索引时（目前就只有MEMORY引擎显式地支持这种索引），会在盘上生成类似如下的文件：

| hash值　　| 存储地址

|| 1db54bc745a1 | 77#45b5 |

| 4bca452157d4 | 76#4556,77#45cc... |

...

hash值即为通过特定算法由指定列数据计算出来，磁盘地址即为所在数据行存储在硬盘上的地址（也可能是其他存储地址，其实MEMORY会将hash表导入内存）。

这样，当我们进行WHERE age = 18 时，会将18通过相同的算法计算出一个hash值==>在hash表中到对应的储存地址==>根据存储地址取得数据。

所以，每次查询时都要遍历hash表，直到找到对应的hash值，如（4），数据量大了之后，hash表也变得庞大起来，性能下降，遍历耗时增加，如（5）。

BTREE

BTREE索引就是一种将索引值按一定的算法，存入一个树形的数据结构中，相信学过数据结构的童鞋对当初学习二叉树这种数据结构的经历记忆犹新，反正我当时为了软考可是被这玩意儿好好地折腾了番，不过那次考试好像没怎么考这个。如二叉树一样，每次查询都是从树的入口root开始，依次遍历node，获取leaf。

BTREE在MyISAM里的形式和Innodb稍有不同

在 Innodb里，有两种形态：一是primary key形态，其leaf node里存放的是数据，而且不仅存放了

引键的数据，还存放了其他字段的数据。二是secondary index，其leaf node和普通的BTREE差不多只是还存放了指向主键的信息.

而在MyISAM里，主键和其他的并没有太大区别。不过和Innodb不太一样的地方是在MyISAM里，lea node里存放的不是主键的信息，而是指向数据文件里的对应数据行的信息.

RTREE

RTREE在mysql很少使用，仅支持geometry数据类型，支持该类型的存储引擎只有MyISAM、BDb、nnoDb、NDb、Archive几种。

相对于BTREE，RTREE的优势在于范围查找.

各种索引的使用情况

（1）对于BTREE这种Mysql默认的索引方式，具有普遍的适用性

（2）由于FULLTEXT对中文支持不是很好，在没有插件的情况下，最好不要使用。其实，一些小的博应用，只需要在数据采集时，为其建立关键字列表，通过关键字索引，也是一个不错的方法，至少我经常这么做的。

（3）对于一些搜索引擎级别的应用来说，FULLTEXT同样不是一个好的处理方法，Mysql的全文索引立的文件还是比较大的，而且效率不是很高，即便是使用了中文分词插件，对中文分词支持也只是一。真要碰到这种问题，Apache的Lucene或许是你的选择。

（4）正是因为hash表在处理较小数据量时具有无可比拟的素的优势，所以hash索引很适合做缓存（存数据库）。如mysql数据库的内存版本Memsql，使用量很广泛的缓存工具Mencached，NoSql数库redis等，都使用了hash索引这种形式。当然，不想学习这些东西的话Mysql的MEMORY引擎也是以满足这种需求的。

（5）至于RTREE，至今还没有使用过，它具体怎么样，我就不知道了。有RTREE使用经历的同学，时可以交流下！