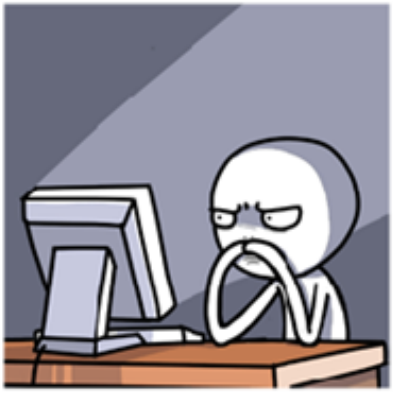
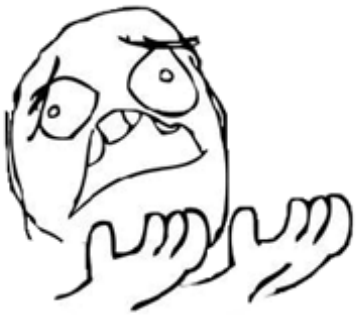
MySQL索引学习漫画

2015-09-18 戴骁雄 微DPer

各位好，我是DBA小熊，本次教程的目的旨在用通俗易懂的文字，帮助大家理解MySQL的索引查询原理，以便各位开发大爷能在设计表时，就能合理地创建好索引。





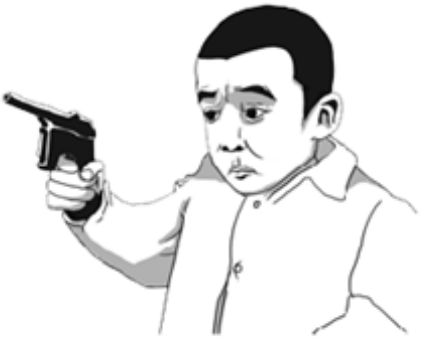
什么？你说你想看那种一本正经的教程？

像这样的？

好吧，请baidu 高性能MySQL 自行阅读第5章

全文终

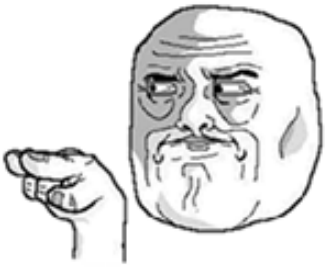
那么，我们就从一个简单的 栗子 例子开始吧



比如淘宝我的宝贝那个页面，如果要按照时间排序

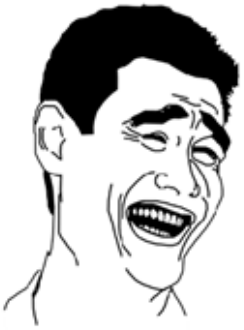
```
select * from xxx where userid = 123 order by addtime desc
```

这个索引要肿么加？



首先最基本的，userId这个字段肯定要加索引

那addTime建立索引与没有索引这样排序的性能会有多大差别？



我对索引的理解不是很深

好吧。。那就得从头讲起了。。

索引是树的结构你知道吧

xxx.addtime
今天
前天
昨天
明天
大年夜
宇宙大爆炸
23世纪

叶子节点是索引字段的值，默认是从小到大排好序的

所以查询时，假设前面没有where条件，只是

select * from xxx order by addtime

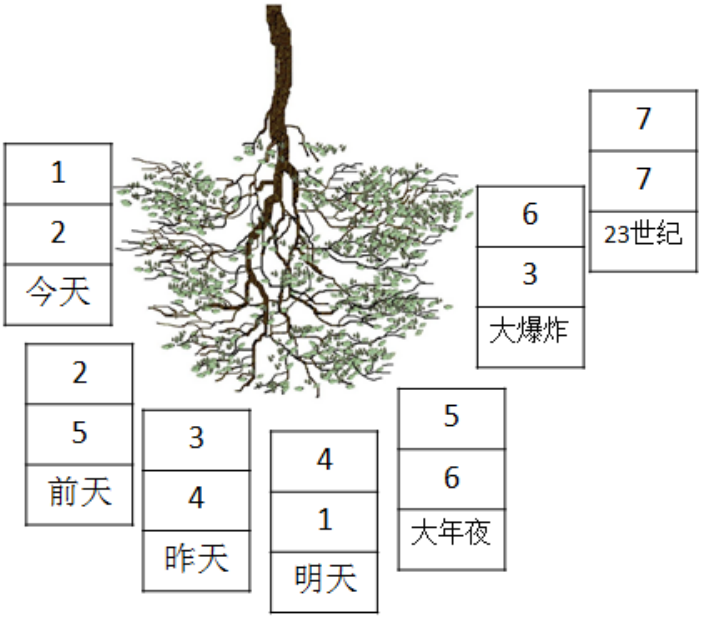
addtime有索引的话，那么只要从索引的树上，把数据直接拎出来就好了

那么这个索引就只存addTime，然后怎么拿到这个addTime所在数据条目中数据的值？

好问题!

首先每个表有一个主键，MySQL会以主键的值构造成一颗树，叶子节点存放着该主键对应的**整行数据**。所以一张表在数据结构上等价于一颗以主键排好序的树。

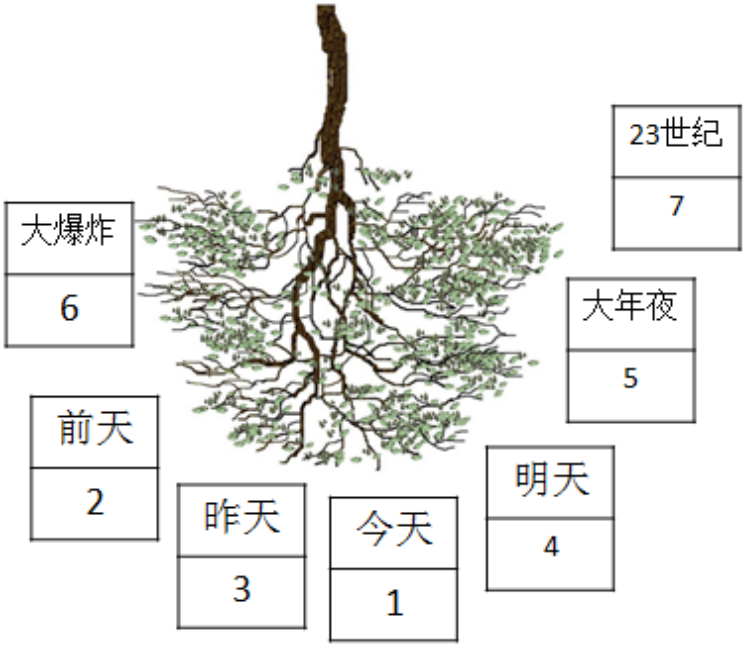
xxx		
主键	userid	addtime
1	2	今天
2	5	前天
3	4	昨天
4	1	明天
5	6	大年夜
6	3	宇宙大爆炸
7	7	23世纪



那么索引的树，和这张表的树，怎么关联呢？

对于其他自己建的索引，一般是叫辅助索引

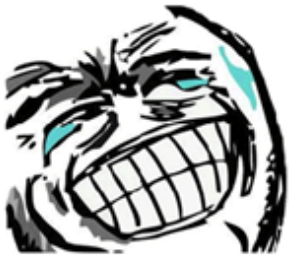
那么辅助索引的树，叶子节点存了两样东西



一是 addtime，并且是排好序的

二是，这个addtime对应的主键的值

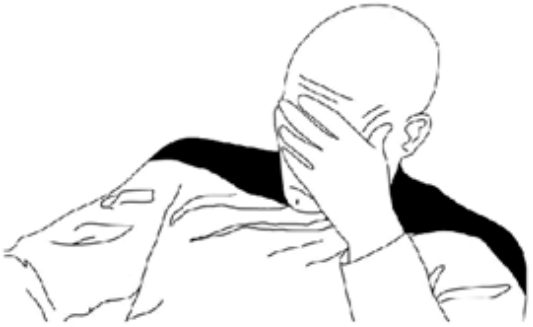
当有where条件，比如 where addtime = '今天'，MySQL直接从 addtime索引，先找到符合条件的addtime，再从叶子节点中取出主键值，跑去主键的那个树，取回整行数据



明白了

如果查询where条件字段没有索引，比如 where updatetime = xxx

那就只能从主键的树，全表扫描... 查所有叶子节点的行数据中，符合条件的



所以可以这样理解么？

对于MySQL表中，常需要用到的查询分组和排序分组条件，最好是加索引

对**区分度高的字段**加索引

有些像什么 status 啊 type 啊，可能就 0 1 2 没几个值的 意义就不大

就像右边的树，就只有3个叉。。并没有什么鸟用

还有就是，SELECT 后面你要查哪些列 也是有讲究的

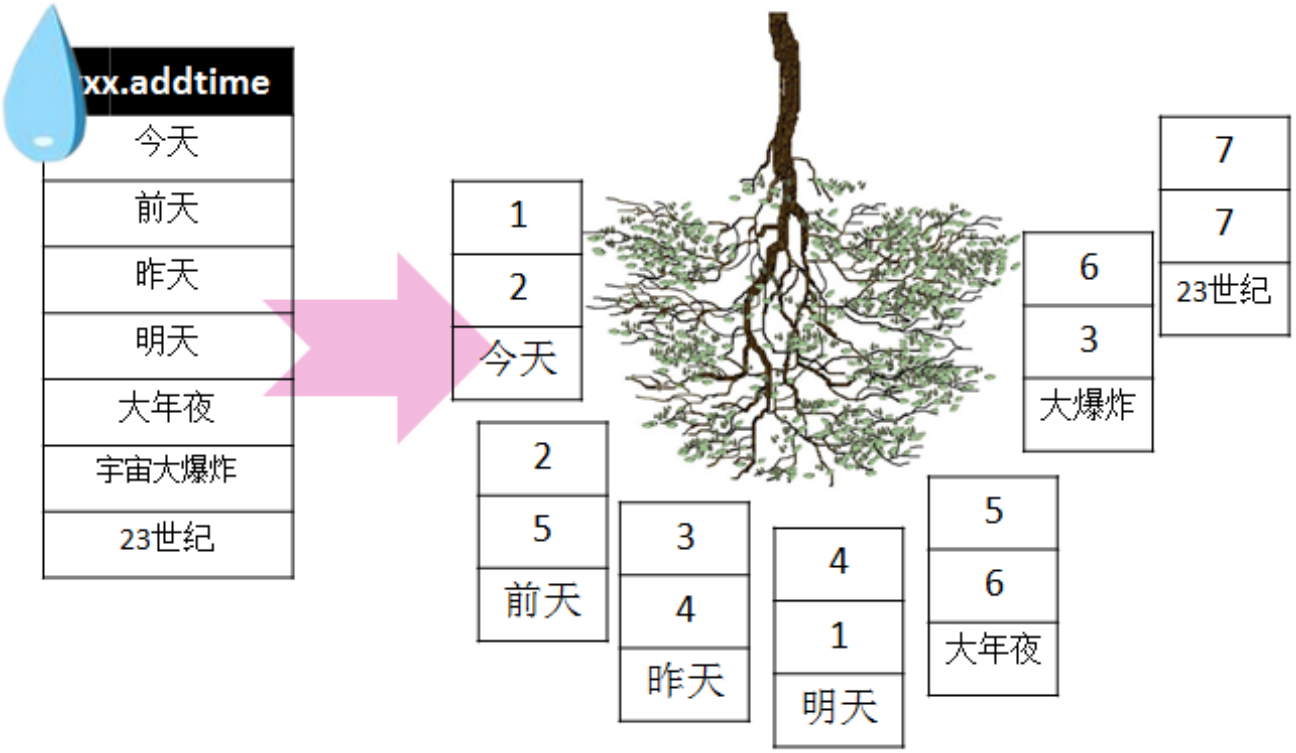
xxx.addtime
今天
前天
昨天
明天
大年夜
宇宙大爆炸
23世纪

比如，SELECT addtime FROM xxx WHERE addtime = 大年夜

因为只需要addtime字段，在addtime索引就能取到，那么就不用回主键索引去查整行数据

而，SELECT addtime, haha FROM xxx WHERE addtime = 大年夜

为了取这个haha的值，就必须要通过addtime索引中取到的主键值，再去主键索引树的叶子节点找到它的值。



所以如果想避免二次查找，可以把 haha 也放到索引里，(addtime, haha)，这样直接从辅助索引中就能取出完整的结果。

组合索引的话，比如KEY（userid， addtime）
就是叶子节点存储userid和addtime的配对，加上对应的主键？

然后顺序是你指定的顺序，先排userid，userid相同的再按addtime排序

所以如果一句查询 只查 WHERE addtime = xxx 是利用不到这个索引的。

那对前面 where userid = 123 order by addtime

这样就很快了，联合索引的话 拿到userId = xxx的时候，addTime已经排好序了

如果查询条件有多个字段，有部分建立索引的，

比如 a = 1 AND b = 2 AND c = 3， a和b分别有单独的索引

一定会优先找有索引的先查询缩小数据量么？还是根据sql语句写的顺序来的

和顺序无关

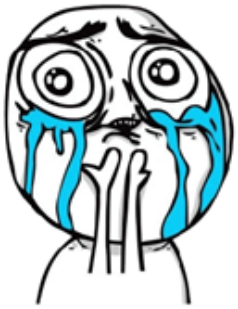
MySQL会看看到底是 按a取的少 还是按b取的少。

如果是a取的少呢？先把满足 a = 1的数据拿出来

这种时候 拿出了部分数据，b的索引怎么用？

b的索引就不用了，一**次查询**同一张表**只能用一条索引**

因此如果一个表上 奇怪的索引多了的话，会影响MySQL判断的，从而走不到正确的索引

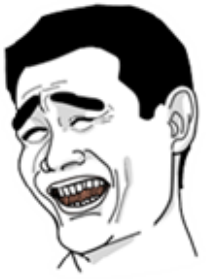


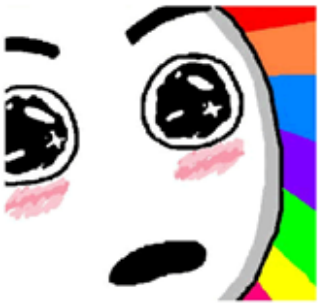
那如果确实有很多索引的话有什么办法吗

可以强制指定使用某个索引

select * from xxx FORCE INDEX (IX_addtime)

这样就是死活都会走addtime上的索引





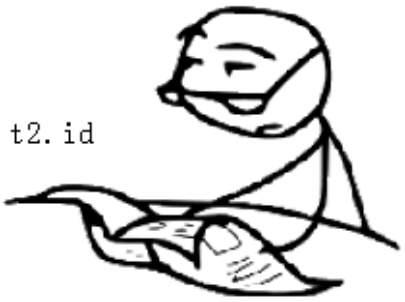
还可以强制索引的～又学到了

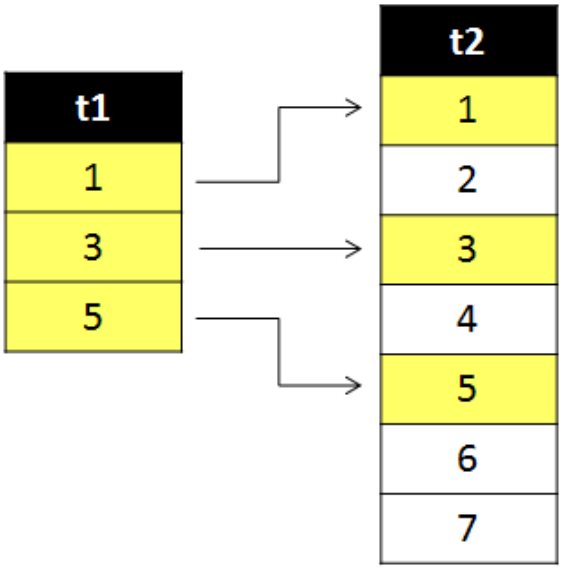
那JOIN查询要肿么整呢

先看简单的

```
SELECT xxx FROM t1 JOIN t2 ON t1.id = t2.id
```

暂时无视SELECT 后面的字段





MySQL先比较t1、t2哪个表小，假设t1小，这里小主要指行数少。然后就把t1的id全部取出来，这步是全表扫描，然后到t2里去根据这些id一个个查。

所以这里的优化关键是，大表t2的id要加索引，小表反正是要全表扫了，逃不掉，加了索引也白加

小表决定循环的次数

大表决定每次循环的查询时间

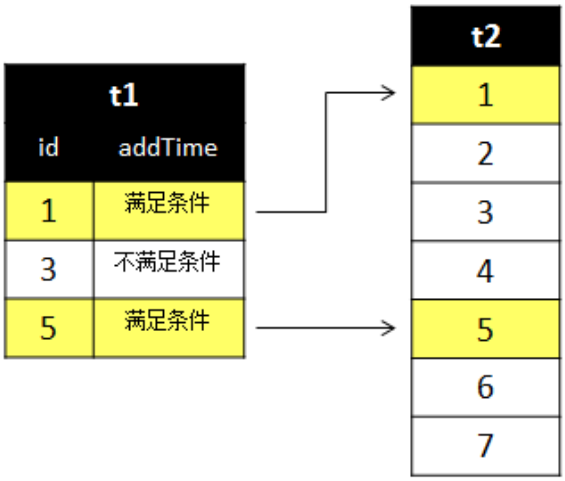
懂了，都是要找小表的id到大表去匹配，大表有id索引就很不一样了

SELECT xxx FROM t1 JOIN t2 ON t1.id = t2.id

WHERE t1.addtime = xxx

这时候小表有查询条件了，怎么办？

如果小表有查询条件，先把小表符合条件的数据过滤出来，再根据id到大表查数据？



对t1小表， 要在 addTime上加索引，能增加过滤的速度

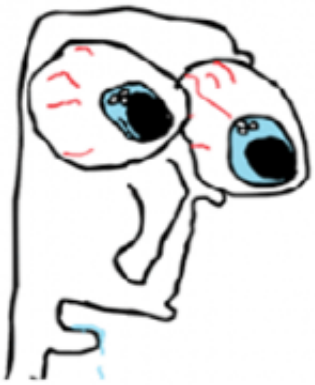
仅在id上加索引并没有什么鸟用，为了过滤addTime还是要全表扫描

那如果大表有查询条件，那么就先小表id找出来，大表根据条件过滤数据，再根据id在过滤完数据的子集里匹配?

SELECT xxx FROM t1 JOIN t2 ON t1.id = t2.id

WHERE t1.addtime = xxx AND t2.addtime = xxxx

首先如果是这样的话，这时候小表大表就不绝对了 =-=



t1	
id	addTime
1	满足条件
3	满足条件
5	满足条件

t2	
id	addTime
1	满足条件
2	不满足条件
3	不满足条件
4	不满足条件
5	不满足条件
6	不满足条件
7	不满足条件

有可能之前的大表t2，根据addTime出来的结果集反而小。

这时候大表小表，其实是按，对两张表分别执行对应的查询条件，**哪个开销小**，哪个就是小表。上面的例子中，假设t2.addTime有索引，t1.addTime无索引，那么会将t2认作小表

那假设还是t1小吧。。到了t2这头，就要根据 id 和addTime 同时查。此时，如果只有id索引，那么就是id查到主键值，再走主键索引查整行，判断addtime是不是符合。

嗯...所以如果id和addtime有联合索引 是最优的！

你已经能抢答了

那老师帮我看一下这几条SQL语句吧

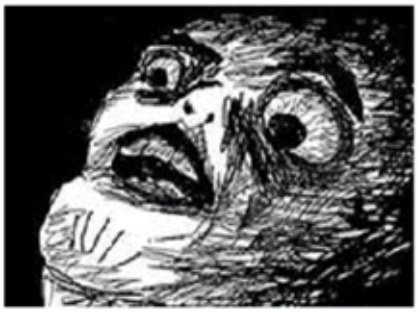
select ... like '%xxx%'

select ... not in ()

select ... != xxx

select ... <> xxx

这都是谁教的？！



like 通配符放最前，还有这几种运算符，都无法用到索引，给我记住

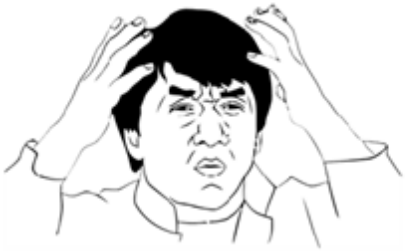
还有...

select ... where md5(password) = 'xxx'

对列不准进行函数运算，也是无法用到索引的

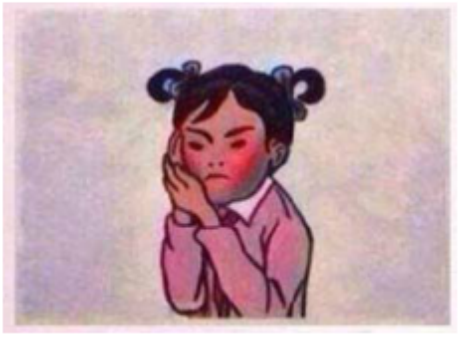
select ... where mobile = 13711112222

mobile就算是存了数值的字符串类型
也不能不写引，数据类型转换走不到索引



select ... where a = xxx OR b = xxx

OR也用不到索引



快下课吧...老师

那么就来做一个总结吧。

1. MySQL以主键的值构造成一颗树，叶子节点存放着该主键对应的整行数据。此为聚簇索引。
2. 其他的索引为辅助索引，叶子节点存放着索引字段的值及对应的主键值。
3. 一般情况下，一次查询只能使用一条索引
4. 对查询where条件中区分度高的字段加索引
5. 联合索引，叶子节点存储的顺序以创建时指定的顺序为准，因此区分度高的放左边，能被多个查询复用到的放左边
6. 只select需要用到的字段，尽量避免select *
7. 如有必要，可使用FORCE INDEX强制索引
8. 多表JOIN，先按各表的查询条件比较哪个开销小，从小表取出所有符合条件的，到大表循环查找
9. 以下情况无法使用到索引，like通配符在最左，not in，!=，<>，对列做函数运算，隐式数据类型转换，OR子句

其他想到再补充咯～ 下课!

微信扫一扫
关注该公众号