

05 | 深入浅出索引 (下)

2018-11-23 林晓斌

MySQL实战45讲

[进入课程 >](#)



讲述：林晓斌

时长 09:53 大小 4.53M



在上一篇文章中，我和你介绍了 InnoDB 索引的数据结构模型，今天我们再继续聊聊跟 MySQL 索引有关的概念。

在开始这篇文章之前，我们先来看一下这个问题：

在下面这个表 T 中，如果我执行 `select * from T where k between 3 and 5`，需要执行几次树的搜索操作，会扫描多少行？

下面是这个表的初始化语句。

复制代码

```
1 mysql> create table T (
```

```

2 ID int primary key,
3 k int NOT NULL DEFAULT 0,
4 s varchar(16) NOT NULL DEFAULT '',
5 index k(k))
6 engine=InnoDB;
7
8 insert into T values(100,1, 'aa'),(200,2,'bb'),(300,3,'cc'),(500,5,'ee'),(600,6,'ff'),(700,7,'gg');

```

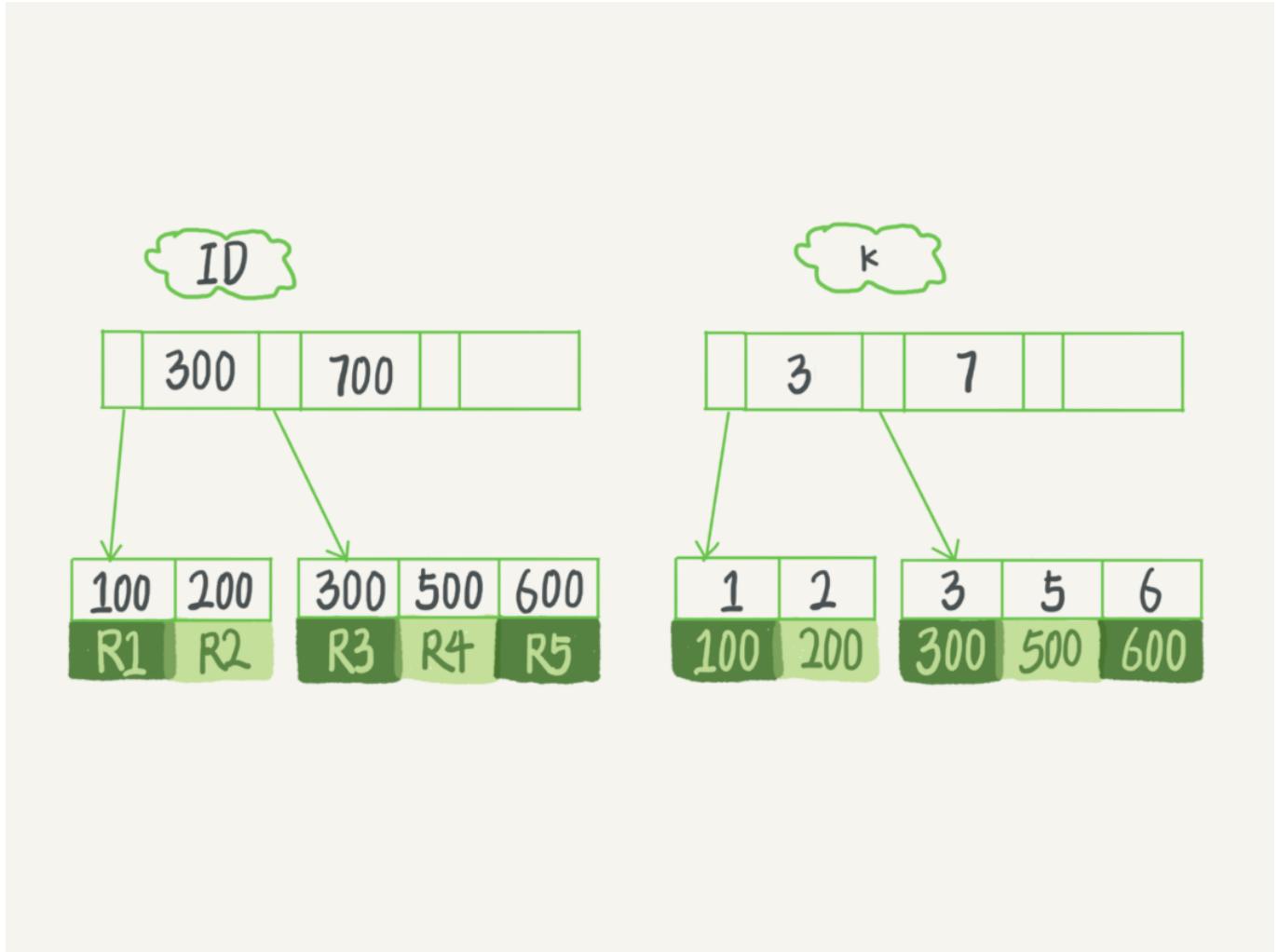


图 1 InnoDB 的索引组织结构

现在，我们一起来看看这条 SQL 查询语句的执行流程：

1. 在 k 索引树上找到 $k=3$ 的记录，取得 $ID = 300$ ；
2. 再到 ID 索引树查到 $ID=300$ 对应的 $R3$ ；
3. 在 k 索引树取下一个值 $k=5$ ，取得 $ID=500$ ；
4. 再回到 ID 索引树查到 $ID=500$ 对应的 $R4$ ；
5. 在 k 索引树取下一个值 $k=6$ ，不满足条件，循环结束。

在这个过程中，回到主键索引树搜索的过程，我们称为回表。可以看到，这个查询过程读了 k 索引树的 3 条记录（步骤 1、3 和 5），回表了两次（步骤 2 和 4）。

在这个例子中，由于查询结果所需要的数据只在主键索引上有，所以不得不回表。那么，有没有可能经过索引优化，避免回表过程呢？

覆盖索引

如果执行的语句是 select ID from T where k between 3 and 5，这时只需要查 ID 的值，而 ID 的值已经在 k 索引树上了，因此可以直接提供查询结果，不需要回表。也就是说，在这个查询里面，索引 k 已经“覆盖了”我们的查询需求，我们称为覆盖索引。

由于覆盖索引可以减少树的搜索次数，显著提升查询性能，所以使用覆盖索引是一个常用的性能优化手段。

需要注意的是，在引擎内部使用覆盖索引在索引 k 上其实读了三个记录，R3~R5（对应的索引 k 上的记录项），但是对于 MySQL 的 Server 层来说，它就是找引擎拿到了两条记录，因此 MySQL 认为扫描行数是 2。

备注：关于如何查看扫描行数的问题，我将会在第 16 文章《如何正确地显示随机消息？》中，和你详细讨论。

基于上面覆盖索引的说明，我们来讨论一个问题：**在一个市民信息表上，是否有必要将身份证号和名字建立联合索引？**

假设这个市民表的定义是这样的：

 复制代码

```
1 CREATE TABLE `tuser` (
2   `id` int(11) NOT NULL,
3   `id_card` varchar(32) DEFAULT NULL,
4   `name` varchar(32) DEFAULT NULL,
5   `age` int(11) DEFAULT NULL,
6   `ismale` tinyint(1) DEFAULT NULL,
7   PRIMARY KEY (`id`),
8   KEY `id_card` (`id_card`),
9   KEY `name_age` (`name`, `age`)
10 ) ENGINE=InnoDB
```

我们知道，身份证号是市民的唯一标识。也就是说，如果有根据身份证号查询市民信息的需求，我们只要在身份证号字段上建立索引就够了。而再建立一个（身份证号、姓名）的联合索引，是不是浪费空间？

如果现在有一个高频请求，要根据市民的身份证号查询他的姓名，这个联合索引就有意义了。它可以在高频请求上用到覆盖索引，不再需要回表查整行记录，减少语句的执行时间。

当然，索引字段的维护总是有代价的。因此，在建立冗余索引来支持覆盖索引时就需要权衡考虑了。这正是业务 DBA，或者称为业务数据架构师的工作。

最左前缀原则

看到这里你一定有一个疑问，如果为每一种查询都设计一个索引，索引是不是太多了。如果我现在要按照市民的身份证号去查他的家庭地址呢？虽然这个查询需求在业务中出现的概率不高，但总不能让它走全表扫描吧？反过来说，单独为一个不频繁的请求创建一个（身份证号，地址）的索引又感觉有点浪费。应该怎么做呢？

这里，我先和你说结论吧。**B+ 树这种索引结构，可以利用索引的“最左前缀”，来定位记录。**

为了直观地说明这个概念，我们用（name，age）这个联合索引来分析。

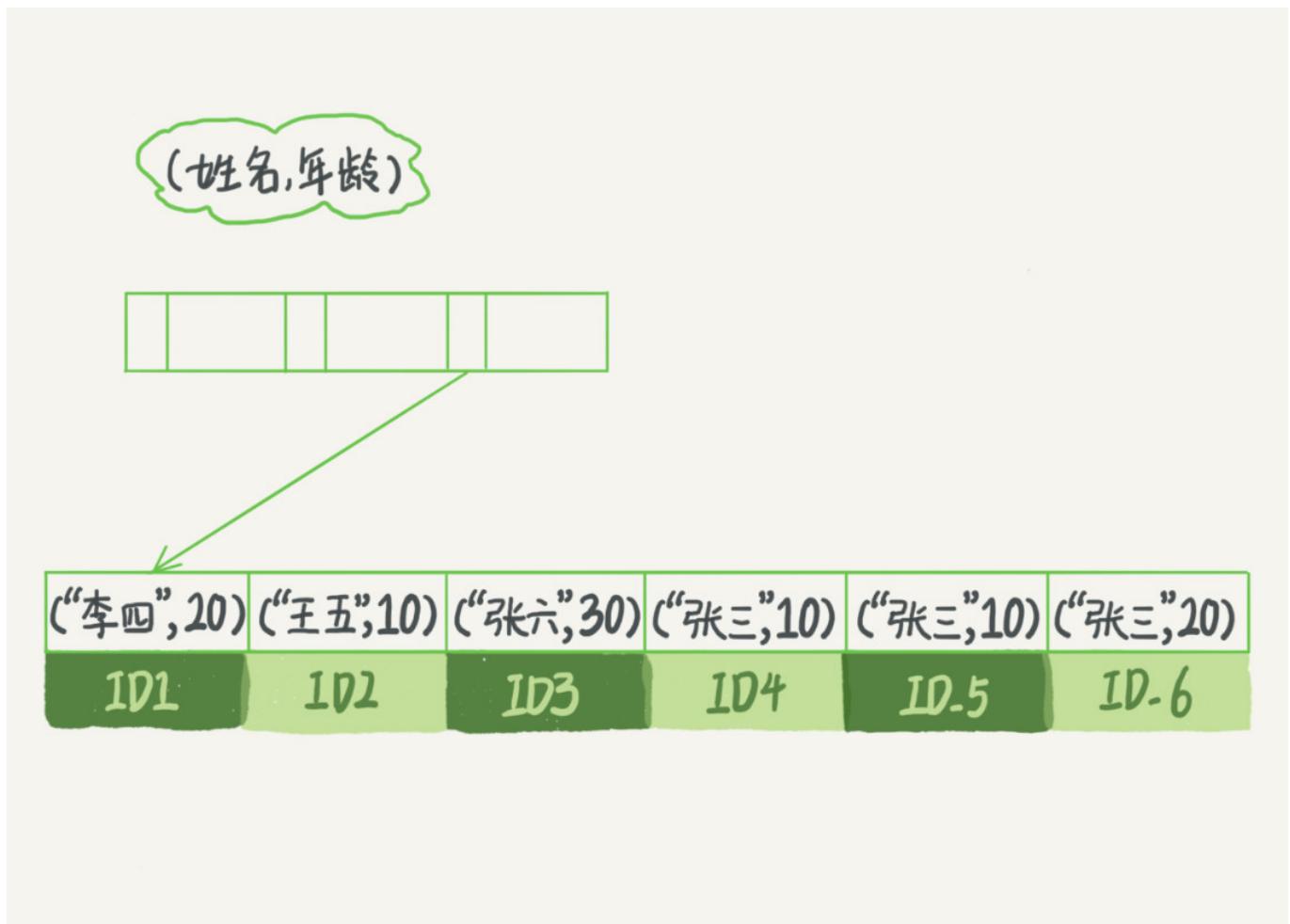


图 2 (name , age) 索引示意图

可以看到，索引项是按照索引定义里面出现的字段顺序排序的。

当你的逻辑需求是查到所有名字是 “张三” 的人时，可以快速定位到 ID4，然后向后遍历得到所有需要的结果。

如果你要查的是所有名字第一个字是 “张” 的人，你的 SQL 语句的条件是 "where name like '张 %'"。这时，你也能够用上这个索引，查找到第一个符合条件的记录是 ID3，然后向后遍历，直到不满足条件为止。

可以看到，不只是索引的全部定义，只要满足最左前缀，就可以利用索引来加速检索。这个最左前缀可以是联合索引的最左 N 个字段，也可以是字符串索引的最左 M 个字符。

基于上面对最左前缀索引的说明，我们来讨论一个问题：**在建立联合索引的时候，如何安排索引内的字段顺序。**

这里我们的评估标准是，索引的复用能力。因为可以支持最左前缀，所以当已经有了 (a,b) 这个联合索引后，一般就不需要单独在 a 上建立索引了。因此，**第一原则是，如果通过调整顺序，可以少维护一个索引，那么这个顺序往往就是需要优先考虑采用的。**

所以现在你知道了，这段开头的问题里，我们要为高频请求创建 (身份证号，姓名) 这个联合索引，并用这个索引支持“根据身份证号查询地址”的需求。

那么，如果既有联合查询，又有基于 a、b 各自的查询呢？查询条件里面只有 b 的语句，是无法使用 (a,b) 这个联合索引的，这时候你不得不维护另外一个索引，也就是说你需要同时维护 (a,b)、(b) 这两个索引。

这时候，我们要**考虑的原则就是空间**了。比如上面这个市民表的情况，name 字段是比 age 字段大的，那我就建议你创建一个 (name,age) 的联合索引和一个 (age) 的单字段索引。

索引下推

上一段我们说到满足最左前缀原则的时候，最左前缀可以用于在索引中定位记录。这时，你可能要问，那些不符合最左前缀的部分，会怎么样呢？

我们还是以市民表的联合索引 (name, age) 为例。如果现在有一个需求：检索出表中“名字第一个字是张，而且年龄是 10 岁的所有男孩”。那么，SQL 语句是这么写的：

 复制代码

```
1 mysql> select * from tuser where name like '张 %' and age=10 and ismale=1;
```

你已经知道了前缀索引规则，所以这个语句在搜索索引树的时候，只能用“张”，找到第一个满足条件的记录 ID3。当然，这还不错，总比全表扫描要好。

然后呢？

当然是判断其他条件是否满足。

在 MySQL 5.6 之前，只能从 ID3 开始一个个回表。到主键索引上找出数据行，再对比字段值。

而 MySQL 5.6 引入的索引下推优化 (index condition pushdown)，可以在索引遍历过程中，对索引中包含的字段先做判断，直接过滤掉不满足条件的记录，减少回表次数。

图 3 和图 4，是这两个过程的执行流程图。



图 3 无索引下推执行流程

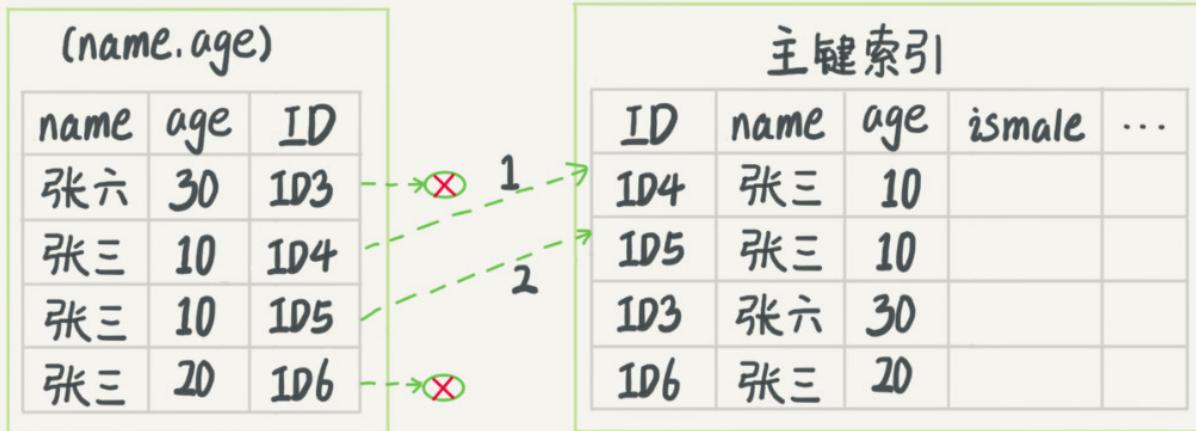


图 4 索引下推执行流程

在图 3 和 4 这两个图里面，每一个虚线箭头表示回表一次。

图 3 中，在 (name,age) 索引里面我特意去掉了 age 的值，这个过程 InnoDB 并不会去看 age 的值，只是按顺序把 “name 第一个字是‘ 张’ ” 的记录一条条取出来回表。因此，需要回表 4 次。

图 4 跟图 3 的区别是，InnoDB 在 (name,age) 索引内部就判断了 age 是否等于 10，对于不等于 10 的记录，直接判断并跳过。在我们的这个例子中，只需要对 ID4、ID5 这两条记录回表取数据判断，就只需要回表 2 次。

小结

今天这篇文章，我和你继续讨论了数据库索引的概念，包括了覆盖索引、前缀索引、索引下推。你可以看到，在满足语句需求的情况下，尽量少地访问资源是数据库设计的重要原则之一。我们在使用数据库的时候，尤其是在设计表结构时，也要以减少资源消耗作为目标。

接下来我给你留下一个问题吧。

实际上主键索引也是可以使用多个字段的。DBA 小吕在入职新公司的时候，就发现自己接手维护的库里面，有这么一个表，表结构定义类似这样的：

 复制代码

```
1 CREATE TABLE `geek` (
2   `a` int(11) NOT NULL,
3   `b` int(11) NOT NULL,
4   `c` int(11) NOT NULL,
5   `d` int(11) NOT NULL,
6   PRIMARY KEY (`a`,`b`),
7   KEY `c` (`c`),
8   KEY `ca` (`c`,`a`),
9   KEY `cb` (`c`,`b`)
10 ) ENGINE=InnoDB;
```

公司的同事告诉他说，由于历史原因，这个表需要 a、b 做联合主键，这个小吕理解了。

但是，学过本章内容的小吕又纳闷了，既然主键包含了 a、b 这两个字段，那意味着单独在字段 c 上创建一个索引，就已经包含了三个字段了呀，为什么要创建 “ca” “cb” 这两个索引？

同事告诉他，是因为他们的业务里面有这样的两种语句：

 复制代码

```
1 select * from geek where c=N order by a limit 1;
2 select * from geek where c=N order by b limit 1;
```

我给你的问题是，这位同事的解释对吗，为了这两个查询模式，这两个索引是否都是必须的？为什么呢？

你可以把你的思考和观点写在留言区里，我会在下一篇文章的末尾和你讨论这个问题。感谢你的收听，也欢迎你把这篇文章分享给更多的朋友一起阅读。

上期问题时间

上期的问题是，通过两个 alter 语句重建索引 k，以及通过两个 alter 语句重建主键索引是否合理。

在评论区，有同学问到为什么要重建索引。我们文章里面有提到，索引可能因为删除，或者页分裂等原因，导致数据页有空洞，重建索引的过程会创建一个新的索引，把数据按顺序插入，这样页面的利用率最高，也就是索引更紧凑、更省空间。

这道题目，我给你的“参考答案”是：

重建索引 k 的做法是合理的，可以达到省空间的目的。但是，重建主键的过程不合理。不论是删除主键还是创建主键，都会将整个表重建。所以连着执行这两个语句的话，第一个语句就白做了。这两个语句，你可以用这个语句代替：alter table T engine=InnoDB。在专栏的第 12 篇文章《为什么表数据删掉一半，表文件大小不变？》中，我会和你分析这条语句的执行流程。

评论区留言中，@壹笙漂泊 做了很详细的笔记，@高枕 帮同学解答了问题，@约书亚 提了一个很不错的面试问题。在这里，我要和你们道一声感谢。

PS：如果你在面试中，曾有过被 MySQL 相关问题难住的经历，也可以把这个问题发到评论区，我们一起来讨论。如果解答这个问题，需要的篇幅会很长的话，我可以放到答疑文章展开。



MySQL 实战 45 讲

从原理到实战，丁奇带你搞懂 MySQL

林晓斌

网名丁奇
前阿里资深技术专家



上一篇 04 | 深入浅出索引（上）

下一篇 06 | 全局锁和表锁：给表加个字段怎么有这么多阻碍？

精选留言 (305)

写留言



我来也 置顶

2018-11-24

126

老师的每一篇都会讲到平常工作用遇到的事情。这个专栏真的很值。

今天这个 alter table T engine=InnoDB 让我想到了我们线上的一个表, 记录日志用的, 会定期删除过早之前的数据。最后这个表实际内容的大小才10G, 而他的索引却有30G. 在阿里云控制面板上看, 就是占了40G空间. 这可花的是真金白银啊.

后来了解到是 InnoDB 这种引擎导致的, 虽然删除了表的部分记录, 但是它的索引还在, 并...

展开 ▼

作者回复: 😊 确实例子都是血泪史, 有些是我的血泪、有些是帮助人擦眼泪▷

也鼓励大家把平时碰到的问题提出来, 大家一起未雨绸缪♡



约书亚 置顶

2018-11-23

25

疑问：

1. 有些资料提到，在不影响排序结果的情况下，在取出主键后，回表之前，会在对所有获取到的主键排序，请问是否存在这种情况？
2. 索引下推那个例子，感觉5.6之前的机制很匪夷所思：感觉判断'张%'之后再“看age的值”是顺理成章的事。难道联合索引的底层实现结构在这期间发生了变化？

展开 ▼

作者回复: 1. 有的， Multi-Range Read (MRR) 由于不论是否使用这个策略，SQL语句写法不变，就没有在正文中提

2. 不是，是接口能力发生了变化，以前只能传“搜索关键字”。

如果你用过5.1 甚至5.0 , 在从现在的观点看 , 你会发现很多 “匪夷所思” 。还有 : 并行复制官方
5.6才引入、MDL 5.5 才有、Innodb 自增主键持久化、多源复制、online DDL ...

只能说 , 持续进化 , 幸甚至哉 😊



发条橙子 ... 置顶

2018-11-24

24

老师 , 因为正文不能无限细节和篇幅的缘故 , 有些细节点没有说 , 我也一直很困惑 , 希望能帮忙解答下 , 辛苦了

1. 表的逻辑结构 , 表 —> 段 —> 段中存在数据段(leaf node segment) , 索引段(Non-leaf node segment) , 请问数据段就是主键索引的数据 , 索引段就是二级索引的数据么
2. 建立的每个索引都有要维护一个数据段么 ? ? 那么新插入一行值 , 岂不是每个索引...

展开 ▼

作者回复: 1. 这样理解也算对 , 不过要记得 主键也是索引的一种哈

2. 是的 , 所以说索引越多 , “维护成本” 越大
3. 如果是几百个儿子节点共用一个父节点 , 是不是就不会看上去那么浪费啦
4. 树高其实取决于叶子树 (数据行数) 和 “N叉树” 的N。而N是由页大小和索引大小决定的。
5. 基本是你说的流程。不过不是 “优化器” 去取的 , 是执行器调用引擎 , 引擎内部才管理了你说的 段、页这些数据



locust 置顶

2018-11-28

16

老师 , 有这么个问题

一张表两个字段id, uname,id主键 , uname普通索引

SELECT * FROM test_like WHERE uname LIKE 'j' / 'j%' / '%j' / '%j%

模糊查询like后面四种写法都可以用到uname的普通索引

...

展开 ▼

作者回复: 好问题 , 这个是关于 “用索引” 和 “用索引快速定位记录” 的区别。

08 篇讲到这个问题了 , 周五关注一下。

简单回答 : “用索引” 有一种用法是 “顺序扫描索引”



老北 置顶

2018-11-24

10

背景:

我们现在有一张表,每天生成300W数据, 然后每天用delete xx where id = x 这样的方式来删除.

不用truncate是因为DBA说truncate会重建自适应哈希索引,可能对整个库性能有影响.

...

展开 ▼

作者回复: 额你们DBA可能对自适应哈希索引 (AHI) 有误解...有其他同学也在评论中有提到 AHI , 我答疑文章会安排说明。

看你的描述 , 最好就是rename 重建一个新的 , 然后找低峰期删掉旧的表。

还有你这么说 , 应该id就是这个表的自增主键了 , 正常即使删除也不会全表扫描。不过我现在怀疑可能删的事务有没提交的 , 导致MySQL 没法回收复用旧空间。 (这个可以简单从文件大小判断)

不过总之 , rename +新建表 , 上面这个问题也自动解决了 😊



某、人

2018-11-23

62

先回答老师的问题:

如果c列上重复率很低的情况下,两个索引都可以不用建。因为如果过滤只剩下几条数据,排序也不影响

如果C列重复度比较高,就需要建立(c,b)的联合索引了,来消除排序了。因为在数据量大的情况下,排序是一个非常耗时的操作,...

展开 ▼

作者回复: 回答得很好。

1. 没有存 , 就是一个临时内存 , 读出来马上判断 , 然后扫描下一行可以复用
2. Server层。 接上面的逻辑 , 读完以后顺便判断一下够不够limit 的数了 , 够就结束循环
3. 嗯 , 你很细心 , 其实它表示的是 “可以下推” , 实际上是 “可以 , 但没有” 😊



狼的诱惑
2018-11-27

43

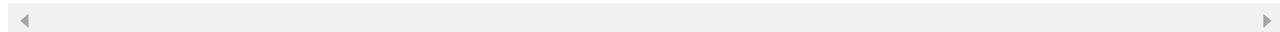
踩过坑：有人问我联合索引的技巧，回答的不是很好

总结：

1、覆盖索引：如果查询条件使用的是普通索引（或是联合索引的最左原则字段），查询结果是联合索引的字段或是主键，不用回表操作，直接返回结果，减少IO磁盘读写读取正行数据...

展开 ▾

作者回复: 赞，下次再问你就这么答，棒棒哒



benson42
2018-11-23

40

这两个语句，你可以用这个语句代替：alter table T engine=InnoDB。可以解释一下原理吗。



gether
2018-11-23

37

ca索引可以去掉，cb索引可以保留。

ca索引，通过索引对数据进行筛选，回表的时候，a本身就是主键索引，所以可以保证有序；

cb索引，b上并没有索引，ab索引也无法满足最左匹配原则，可以保留加快排序速度。

但如果查询结果很少的话，内存中排序也够了吧，也没必要建立cb索引。老师，我理解...

展开 ▾



老杨同志
2018-11-23

29

表记录

--a--|--b--|--c--

1 2 3

1 3 2

1 4 3...

展开 ▾



melon

24



2018-11-27

关于联合索引我的理解是这样的：比如一个联合索引(a,b,c)，其实质是按a,b,c的顺序拼接成了一个二进制字节数组，索引记录是按该字节数组逐字节比较排序的，所以其是先按a排序，再按b排序，再按c排序的，至于其为什么是按最左前缀匹配的也就显而易见了，没看过源码，不知道理解的对不对，希望老师指正。

...

展开 ▾

作者回复: 非常赞，尤其是第三段对“军规”的理解凸



工资不交税

2018-11-23

17

老师好，文章提到建立一个（身份证号、姓名）的联合索引，是不是浪费空间？后文解释如果根据身份证号查询姓名和年龄就会用到覆盖索引。这里我不理解的是，年龄并不在联合索引内，那是不是应该回表啊？



grey

2018-11-23

14

老师你好，上篇文章中有人提问“数据量很大的时候，二级索引比主键索引更快”，这个结论是只有在使用覆盖索引时才成立吧，非覆盖索引还是要回表查询。

展开 ▾

作者回复: 是的凸



壹笙漂泊

2018-11-23

13

总结：

回表：回到主键索引树搜索的过程，称为回表

覆盖索引：某索引已经覆盖了查询需求，称为覆盖索引，例如：select ID from T where k between 3 and 5

在引擎内部使用覆盖索引在索引K上其实读了三个记录，R3~R5(对应的索引k上的记录…)

展开 ▾

作者回复: 是的，查询语句的where里面各个判断调换顺序没关系的



北天魔狼

2018-11-23

11

老师，我是非科班做开发的小白（公司没有DBA），最近一直认为数据库和程序运行环境都比开发语言本身重要。尤其是数据库，数据行上亿必须在数据库上想办法。也买了一本高性能MySQL，看完类型，索引，查询，后面就看不懂了。特别期待后面的章节，六点上地铁，看的入迷差点坐过站

展开▼



dior

2018-12-19

10

面试官问：说下怎么让mysql的myisam引擎支持事务，网上搜了下，也没有结果！

作者回复: 面试官是魔鬼吗😊

我怀疑他是想说用lock table 来实现，但是这样只能实现串行化隔离级别，

其它隔离都实现不了。

但是因为mysiam不支持崩溃恢复，所以即使用lock table硬实现，也是问题多多：

ACID里面，原子性和持久性做不到；

隔离性只能实现基本用不上的串行化；

一致性在正常运行的时候依赖于串行化，在异常崩溃的时候也不能保证。

这样实现的事务不要也罢。

你这么答复面试官，应该能加到分吧😊



HwangZHen

2018-11-23

9

包含主键后应该是cab，根据最左匹配原则，cb是有必要的，ca没有必要



lionetes

6

alter table T engine=InnoDB 是用来释放 delete 操作引起的页的空洞,也就是碎片空间操作时候尽量避免当前表的dml 操作.

表数据很大情况 建议使用 Percona Toolkit 工具来执行

...

展开 ▼



null

2019-01-08

5

【备忘】

locust 童鞋 like 'j' 或 'j%' 或 '%j' 或 '%j%' 使用索引的问题：

- a. 只有 id 和 uname 字段。
- b. 添加了 age 字段，即 id、uname、age 字段。...

展开 ▼

作者回复: 赞，很好的总结



▶



Jefitar

2019-02-09

4

总结：

- 高频查询，可以建立联合索引来使用覆盖索引，不用回表。
- 非高频查询，再已有的联合索引基础上，使用最左前缀原则来快速查询。
- 对于MySQL 5.6 引入索引下推，减少回表次数。

展开 ▼

作者回复: ▶



▶