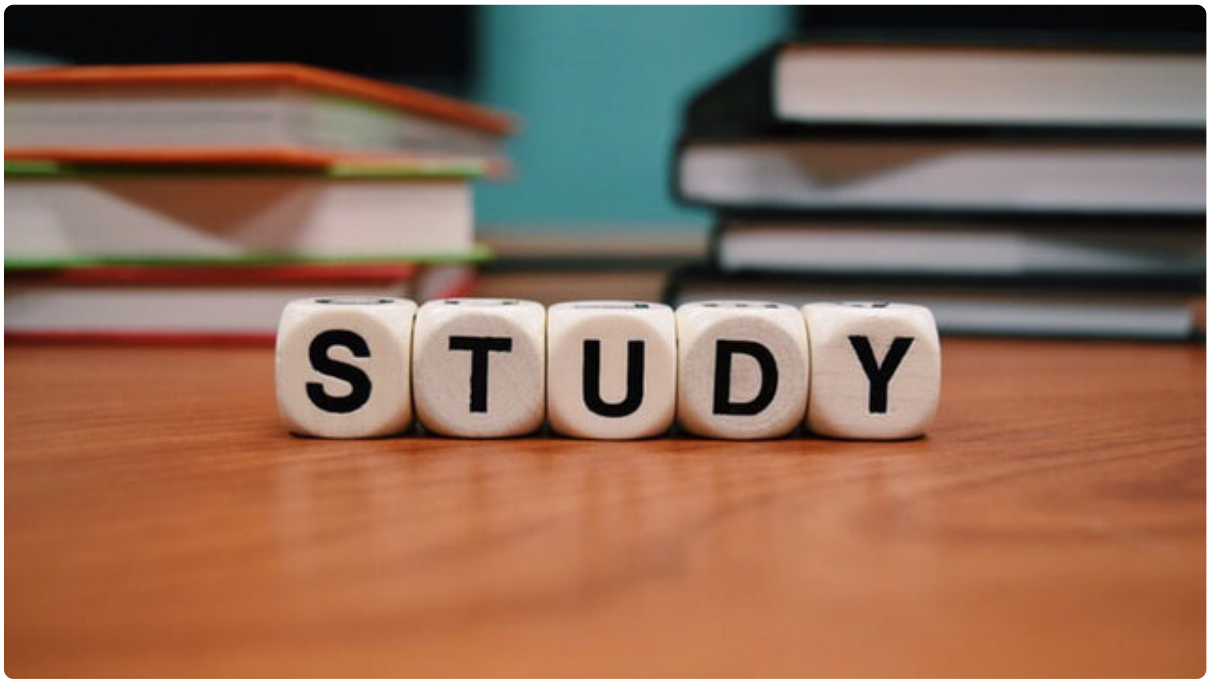
加餐：答疑篇（一）

更新时间：2019-09-10 11:26:46



> 能够生存下来的物种,并不是那些最强壮的,也不是那些最聪明的,而是那些对变化作出快速反应的。
>
> ——达尔文

到这里，专栏已经更新了 16 节。在这段时间里，我收集了一些比较好的问题。这里就增加一节答疑篇，跟大家分享这些好问题。

1 关于条件字段做函数操作的问题

在第 4 节中，讲到了条件字段做函数操作的例子，我们知道了下面这条 SQL 走不了索引：

```
select * from t1 where data(c) = '2019-05-21';
```

原因是，索引树中存储的是列的实际值和主键值。如果拿 ‘2019-05-21’ 去匹配，将无法定位到索引树中的值。

有同学就问，如果把语句 “=” 前面的 DATE_FORMAT 按照实际值转换，比如：

```
select * from t1 where DATE_FORMAT(c,'%Y-%m-%d %H:%i:%s')='2019-05-21 00:00:00';
```

已经转换成跟实际值一样，为什么也不走索引？

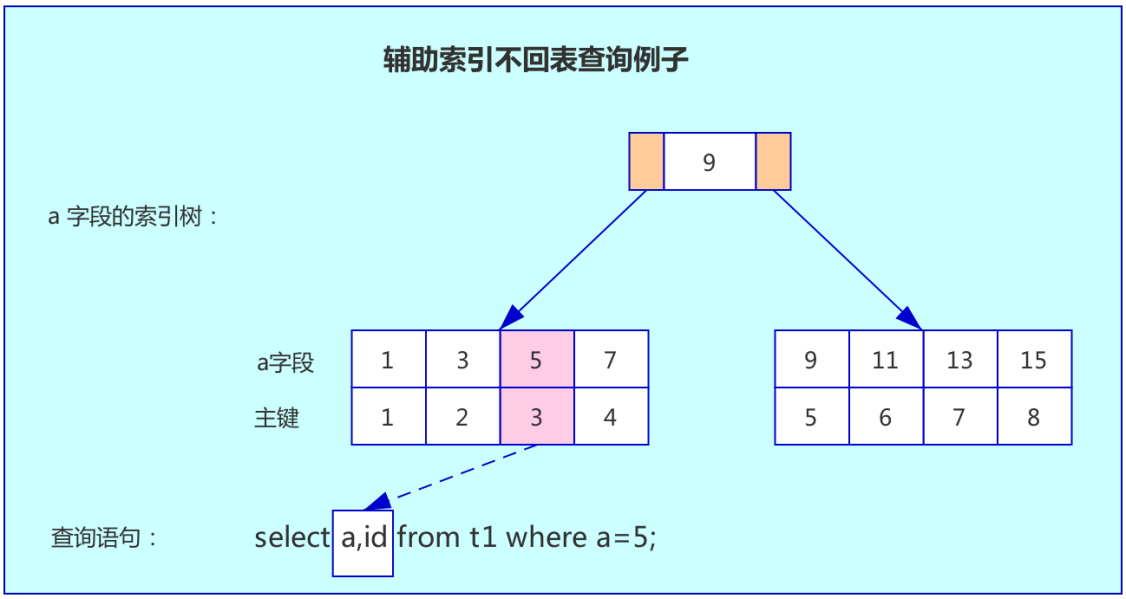
这里就跟大家分享一下不走索引的原因。

原因是：这条 SQL 执行过程是先找出所有 c 字段的值，然后经过 DATE_FORMAT 加工后形成新的值集合（并且是无序的），因此最终是这个新值集合跟 ‘2019-05-21’ 进行比较。而索引中存储的是 c 字段的集合，所以无法使用索引过滤。

2 是不是所有的辅助索引都需要回表？

并不是所有的辅助索引查询都需要回表，如果条件字段跟需要查询的字段在一颗索引树上，可以直接在索引树上找到结果，就不用回表了，通常称为覆盖索引（这个在第 13 节也有提到）。

我们来看下使用覆盖索引的场景：



如上图，如果 a 字段有索引，而查询条件是 a 字段的等值查询，查询结果是 a 字段和主键，那么可以直接通过 a 字段的索引树查询到结果，而不需要回表。

3 Btree 索引和 HASH 索引有什么区别？

两者区别：

- Btree 索引可能需要多次运用折半查找来找到对应的数据库；
- 而 HASH 索引是通过 HASH 函数，计算出 HASH 值，在表中找出对应的数据。

优缺点对比：

- 大量不同数据等值精确查询，HASH 索引效率通常比 B+TREE 高；
- 但是 HASH 索引不支持模糊查询、排序、范围查询和联合索引中的最左匹配原则，而这些 Btree 索引都支持。

4 压力测试的时候，数据库有哪些配置需要重点关注的？

数据库层面需要关注的是：

- MySQL 版本
- innodb_buffer_pool_size
- innodb_flush_log_at_trx_commit
- sync_binlog
- innodb_io_capacity
- 等等

另外就是系统层面的：

- CPU

- 内存
- 磁盘类型
- 等等

5 MySQL 5.6 从库延迟怎么办

临时解决方案是调整这两个参数：

- innodb_flush_log_at_trx_commit = 0
- sync_binlog=200

长久方案是建议开启并行复制，5.6 的多线程复制也只是按库并行的，可以尝试升级到 5.7。

MySQL 5.7 除了支持基于库的并行复制外，还支持基于组提交的并行复制。

6 关于隐式转换的一个问题

在第 4 节里，我们讲解了 隐式转换的例子

表 t1 中 a 字段是字符串类型，有索引，但是如下 SQL 走不了索引：

```
select * from t1 where a=1000;
```

原因是：MySQL 会把 a 转换成 int 型，再去做判断，也相当于对索引字段做函数操作，导致不能使用索引。

有同学提问：表 t1 中，为 int 型的 b 字段，有索引，下面 SQL 为什么可以走索引？

```
select * from t1 where b='1';
```

这是因为：MySQL 中如果条件字段和后面的值不是同一类型，那么优先将字符串转成数字类型做比较。因此上面的 SQL 是转的 “=” 后面的值，而不是字段 b，因此 SQL 其实等价于：

```
select * from t1 where b=1;
```

如下图：

```
mysql> select * from t1 where b=2;
+----+------+------+---------------------+
| id | a    | b    | c                   |
+----+------+------+---------------------+
|  2 | 2    |    2 | 2019-05-22 00:00:00 |
+----+------+------+---------------------+
1 row in set (0.00 sec)

mysql> update t1 set b=0 where b=2;
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql>  select * from t1 where b='a';
+----+------+------+---------------------+
| id | a    | b    | c                   |
+----+------+------+---------------------+
|  2 | 2    |    0 | 2019-05-22 00:00:00 |
+----+------+------+---------------------+
1 row in set (0.00 sec)
```

如果后面的条件是字母，将转换成数字 0。

7 总结

感谢各位在留言区或者社群提问的同学，以上是选取了前面 16 节部分好的问题，我想可能也会有其他同学存在相

同的疑问，因此设置了本节答疑篇。

8 问题

对于前面 16 节的学习，你是否还存在疑问？欢迎在本节评论区留言讨论。

}

16 行锁：InnoDB替代MyISAM的重要原因

17 间隙锁的意义