

# Innodb的覆盖索引实践

原创

大数据模型

2024-02-19

279



## 前言

MySQL的索引分为6种级别，级别如下

- system：系统表，少量数据，往往不需要进行磁盘IO
- const：常量连接
- eq\_ref：主键索引(primary key)或者非空唯一索引(unique not null)等值扫描
- ref：非主键非唯一索引等值扫描
- range：范围扫描
- index：索引树扫描
- ALL：全表扫描(full table scan)

6种级别中，type扫描方式由快到慢，system > const > eq\_ref > ref > range > index > ALL

5.6X开始，现在的MySQL默认引擎是INNODB,innodb是索引组织结构，首先从主键去扫描，按照表的主键构造一颗B+树，非主键索引指向主键 管理叶子节点中存放的就是整张表的行记录数据 主键索引又称为 clustered索引，非主键索引称为辅助索引、二级索引。

主键索引的点查询【where 索引=XXX】隶属于 const，非主键索引的点查询则属于 ref，非主键索引一般情况总是要回表，那么二级索引有没有可能 更快，有可能，那就是覆盖索引。

```
-----+
| test | CREATE TABLE `test` (
| id` int(1) NOT NULL AUTO_INCREMENT,
| id2` int(2) NOT NULL,
| name` varchar(8) DEFAULT NULL,
| PRIMARY KEY (`id`),
| KEY `key_id2` (`id2`)
| ENGINE=InnoDB AUTO_INCREMENT=156 DEFAULT CHARSET=latin1 |
+-----+

1 row in set (0.00 sec)

mysql> select * from test;
+----+-----+-----+
| id | id2 | name |
+----+-----+-----+
| 1  | 1   | test1|
| 2  | 2   | test2|
| 7  | 7   | test7|
| 15 | 15  | test15|
| 17 | 17  | test7 |
| 22 | 22  | test22|
+----+-----+-----+
6 rows in set (0.00 sec)

mysql> explain select * from test where id=2;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | SIMPLE     | test | NULL      | const | PRIMARY      | PRIMARY | 4      | const | 1    | 100.00 | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)

mysql> explain select * from test where id2=2;
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1  | SIMPLE     | test | NULL      | ref | key_id2       | key_id2 | 4      | const | 1    | 100.00 | NULL |
+----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
1 row in set, 1 warning (0.00 sec)
```

主键索引

非主键索引

同样点查询，一个用了主键，一个用了非主键，按理const比ref快

## 实验环境

单表数据集	数据结构	查询SQL
100万	主键、索引	sum\group by



大数据模型

关注

78

文章

57

粉丝

109K+

浏览量

- 👍 获得了 394 次点赞
- 💬 内容获得 140 次评论
- 🌟 获得了 94 次收藏

### TA的专栏

- 
- 大数据模型  
收录 66 篇内容
- 
- oceanbase  
收录 10 篇内容
- 
- 数据库文章  
收录 8 篇内容

### 热门文章

- Doris、StarRocks、SelectDB安装体验及对比

2023-09-01

10105浏览
- 国产主流数据库调研

2021-07-07

8441浏览
- 一次OBCP认证的心路历程总结

2022-02-23

6756浏览
- 暗恋达梦那十年:我的职业生涯与数据产品技术发展的历史

2021-09-10

4723浏览
- 国产数据库MogDB理论认识和案例实践应用开发

2022-08-12

4095浏览

### 最新文章

- 游戏用户生命周期观测 | 时序模型业务洞悉

2024-10-10

132浏览
- 国产时序数据库IoTDB调研实践归纳

2024-06-10

262浏览
- 25载春秋-金仓数据库解剖探索

2024-05-18

377浏览
- 「YashanDB迁移体验官」数据迁移工具YMP评测



```
mysql> select version();
+-----+
| version() |
+-----+
| 5.7.34    |
+-----+
1 row in set (0.01 sec)

mysql> select count(*) from tt2;
+-----+
| count(*) |
+-----+
| 1000000  |
+-----+
1 row in set (0.37 sec)

其中c0和c1是同样的数据，
CREATE TABLE `tt2` (
  `c0` int(11) NOT NULL DEFAULT '0',
  `c1` int(11) NOT NULL DEFAULT '0',
  `c2` timestamp(6) NOT NULL DEFAULT CURRENT_TIMESTAMP(6) ON UPDATE CURRENT_TIMESTAMP(6),
  `c3` double NOT NULL
)
```

### 主键查询

```
mysql> alter table  tt2 add primary key(c0);

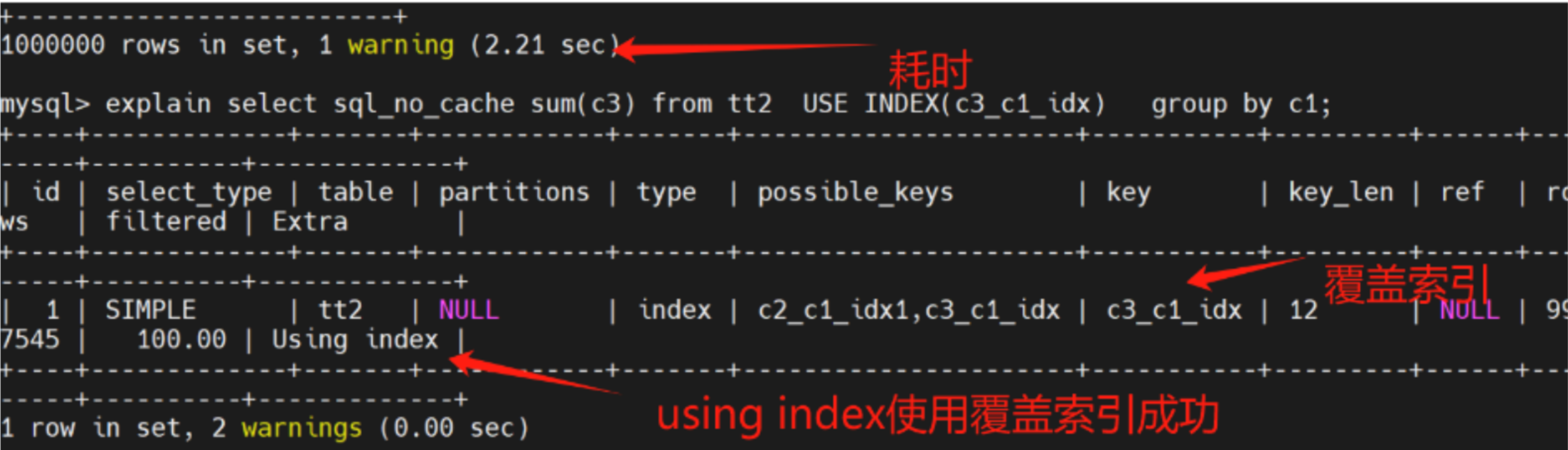
0.0169409732334316 |
| 0.24051994131878 |
+-----+
1000000 rows in set, 1 warning (2.77 sec)

mysql> explain select sql_no_cache sum(c3) from tt2 group by c0;
+---+-----+-----+-----+-----+-----+
| id | select_type | table | partitions | type | possible_keys |
+---+-----+-----+-----+-----+-----+
| 1  | SIMPLE     | tt2   | NULL       | index | PRIMARY | PRIMARY | 4 | NULL | 9975 |
+---+-----+-----+-----+-----+-----+
1 row in set, 2 warnings (0.01 sec)
```

### 非主键索引

创建普通索引

mysql> create index c1\_idx1 on tt2(c1);
Query OK, 0 rows affected (1.95 sec)
Records: 0 Duplicates: 0 Warnings: 0



### 正确的覆盖索引

创建覆盖索引

mysql> create index c3\_c1\_idx on tt2(c1,c3);
Query OK, 0 rows affected (3.47 sec)
Records: 0 Duplicates: 0 Warnings: 0

2024-05-15

362浏览

openGauss的数据处理能力调优实践

2024-04-25

414浏览

#### 目录

• [前言](#)

• [实验环境](#)

• [主键查询](#)

• [非主键索引](#)

• [正确的覆盖索引](#)

• [错误的覆盖索引](#)

• [性能记录](#)

• [总结](#)



```
+-----+
1000000 rows in set, 1 warning (6.41 sec)
mysql> explain select sql_no_cache sum(c3) from tt2 USE INDEX(c1_idx1) group by c1;
+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+
| 1 | SIMPLE | tt2 | NULL | index | c2_c1_idx1,c3_c1_idx,c1_c3_idx,c1_idx1 | c1_idx1 | 4 | NULL | 997545 | 100.00 | NULL |
+-----+
1 row in set, 2 warnings (0.00 sec)
```

耗时  
普通索引  
NULL没有用上覆盖索引

### 错误的覆盖索引

```
mysql> create index c1_c3_idx on tt2(c3,c1);
Query OK, 0 rows affected (4.02 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

```
+-----+
1000000 rows in set, 1 warning (9.44 sec)
mysql> explain select sql_no_cache sum(c3) from tt2 USE INDEX(c3_c1_idx) group by c1;
+-----+
| id | select_type | table | partitions | type | possible_keys | key | key_len | ref | rows | filtered | Extra |
+-----+
| 1 | SIMPLE | tt2 | NULL | index | c2_c1_idx1,c3_c1_idx | c3_c1_idx | 12 | NULL | 997545 | 100.00 | Using index; Using temporary; Using filesort |
+-----+
1 row in set, 2 warnings (0.00 sec)
```

消耗  
覆盖索引  
出现temporary\filesort

错误的覆盖索引导致了filesort和temporary现象，filesort和temporary的技术原理如下。为什么会发生filesort和temporary，主要是innodb是索引组织的有序排序结构，不遵从它的玩法，内存管理上需要耗费额外多的性能。

#### 0. using filesort

filesort主要用于查询数据结果集的排序操作，首先MySQL会使用sort\_buffer\_size大小的内存进行排序，如果结果集超过了sort\_buffer\_size大小，会把这一个排序后的chunk转移到file上，最后使用多路归并排序完成所有数据的排序操作。

MySQL filesort有两种使用模式：

模式1: sort的item保存了所需要的所有字段，排序完成后，没有必要再回表扫描。

模式2: sort的item仅包括，待排序完成后，根据rowid查询所需要的columns。

很明显，模式1能够极大的减少回表的随机IO。

#### 2. using temporary

MySQL使用临时表保存临时的结构，以用于后续的处理，MySQL首先创建heap引擎的临时表，如果临时的数据过多，超过max\_heap\_table\_size的大小，会自动把临时表转换成MyISAM引擎的表来使用。

从上面的解释上来看，filesort和temporary的使用场景的区别并不是很明显，不过，有以下的原则：

filesort只能应用在单个表上，如果有多个表的数据需要排序，那么MySQL会先使用using temporary保存临时数据，然后在临时表上使用filesort进行排序，最后输出结果。

### 性能记录

主键查询(单位秒)	非主键索引	正确覆盖索引	错误覆盖索引
2.71	6.41	2.21	9.44

### 总结

- 主键索引未必是最快的，未必比非主键索引快，根据各种场景而定。
- 组合索引【覆盖索引】注意**先后排序**，否则使用了出现临时空间表、文件排序
- MySQL的索引6种级别，我们做到 ref 就不错了，大部分的情况是 index
- 主键索引是innodb的核心性能的标配，可以选配UUID或者递增ID，熟悉MySQL的人都会选配递增ID，因为只有递增ID才能发挥innodb的性能。递增ID能够使innodb有序存放数据，但是大部分业务场景，我们可能无法使用递增ID做索引，基于安全的需要也不能暴露主键ID。最后二级索引是常态，进一步的优化就是覆盖索引了。

🔗 墨力计划

「喜欢这篇文章，您的关注和赞赏是给作者最好的鼓励」

关注作者

赞赏

【版权声明】本文为墨天轮用户原创内容，转载时必须标注文章的来源（墨天轮），文章链接，文章作者等基本信息，否则作者和墨天轮有权追究责任。如果您发现墨天轮中有涉嫌抄袭或者侵权的内容，欢迎发送邮件至：contact@modb.pro进行举报，并提供相关证据，一经查实，墨天轮将立刻删除相关内容。



评论

分享你的看法，一起交流吧~

相关阅读

本地部署deepseek，创建你的DBA小助理

多明戈教你玩狼人杀 2242次阅读 2025-01-28 18:01:36

2025年2月中国数据库排行榜：OceanBase迎来开门红，金仓、GBASE排名节节高

墨天轮编辑部 1741次阅读 2025-02-11 15:43:30

【干货】2024年下半年墨天轮最受欢迎的50篇技术文章+文档

墨天轮编辑部 1562次阅读 2025-02-13 10:42:44

2025年1月国产数据库大事记

墨天轮编辑部 1149次阅读 2025-01-26 13:56:43

猜灯谜、赢奖品，祝墨友们2025元宵节快乐！

墨天轮福利君 831次阅读 2025-02-10 11:37:13

Oracle Alert 日志频繁告警 12170 TNS-12535/TNS-00505，我看看怎么个事儿？

Lucifer三思而后行 809次阅读 2025-02-06 13:25:43

【大盘点】2024年国产数据库行业有哪些大事发生？

墨天轮编辑部 790次阅读 2025-01-20 12:30:33

总中标金额超亿元！2025年1月国产数据库中标情况一览

通讯员 712次阅读 2025-02-12 16:58:17

MySQL性能分析的“秘密武器”，深度剖析SQL问题

szrsu 682次阅读 2025-01-23 09:59:26

亏麻了~2024年数据库行业遭遇严峻挑战，厂商普遍面临亏损！

通讯员 666次阅读 2025-01-26 12:00:15