MySQL生产实战优化（利用Index skip scan优化性能提升257倍）

原创 陈刚 春田花花程序园 2025年03月31日 15:12 四川

背景

我们生产环境均采用单表多租户设计模式，即一个表中存有多个租户数据，在实际的业务中每个租户的数据是隔离的，即A租户不可能访问到B租户数据，假设租户ID的字段为profileid，我们每个索引都会将profileid放在最左第一列(主键除外)，如（profileid，date,status）。

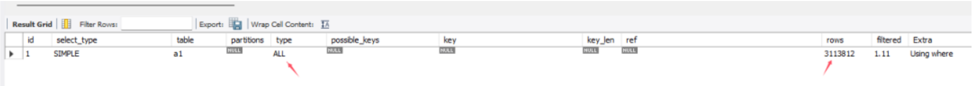
昨天运营需要统计一个线上数据，会跨租户查询,wherer条件中将不能带有租户ID字段，这将导致所有索引均失效，整个SQL查询非常非常慢。

问题SQL

实际语句更复 ，我将语句简单化了，说明优化过程

研发同学写的原版SQL

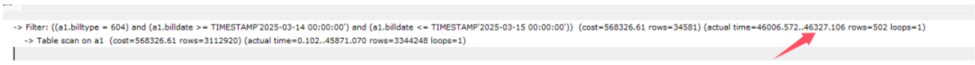
```
explain
select  profileid,billcode,billtype,billdate,billid from a1
where a1.billtype IN( 604) and a1.billdate >='2025-03-14' and  a1.billdate
<='2025-03-15'
```



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	a1	NULL	ALL	NULL	NULL	NULL	NULL	3113812	1.11	Using where

通过执行计划看到，将会是全表扫描，扫描300多万行

实际执行需要46327ms



```
-> Filter: ((a1.billtype = 604) and (a1.billdate >= TIMESTAMP'2025-03-14 00:00:00') and (a1.billdate <= TIMESTAMP'2025-03-15 00:00:00'))  (cost=568326.61 rows=34581) (actual time=46006.572..46327.106 rows=502 loops=1)
    -> Table scan on a1  (cost=568326.61 rows=3112920) (actual time=0.102..45871.070 rows=3344248 loops=1)
```

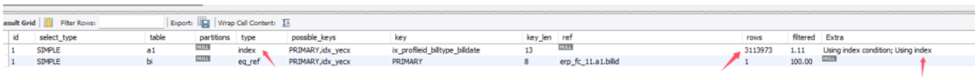
性能优化

优化思路：
实际上此表有（profileid,billtype,billdate）联合索引，而billid 是主键字段，在整个语句中只有billcode不在上述联合索引之内，所以改造成先走Index skip scan，然后再自关联取billcode即可。

一个表中租户一般低于1000,所以完全符合Index skip scan的条件。我来改写，让这个语句走Index skip scan

优化的第一版SQL

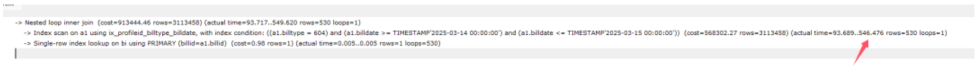
```
explain
select
 t.*,bi.billcode from
(
select  profileid,billtype,billdate,billid from a1
where a1.billtype IN( 604) and a1.billdate >='2025-03-14' and  a1.billdate
<='2025-03-15'
) t
inner join erp_bill_index a1 as bi on t.billid = bi.billid
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	a1	NULL	index	PRIMARY,idx_yecx	ix_profileid_billtype_billdate	13	NULL	3113973	1.11	Using index condition; Using index
1	SIMPLE	bi	NULL	eq_ref	PRIMARY,idx_yecx	PRIMARY	8	erp_fc_11.a1.billid	1	100.00	NULL

通过执行计划发现，MySQL给我合并了子查询，虽然扫描行数未变，但走了索引覆盖扫描
虽然没有走到 index skip scan 但性能还是有较大提升 从46327ms提升到了546ms 提升84倍

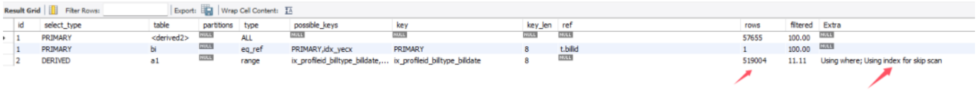


```
-> Nested loop inner join  (cost=913444.46 rows=3113458) (actual time=93.717..549.620 rows=530 loops=1)
    -> Index scan on a1 using ix_profileid_billtype_billdate, with index condition: ((a1.billtype = 604) and (a1.billdate >= TIMESTAMP'2025-03-14 00:00:00') and (a1.billdate <= TIMESTAMP'2025-03-15 00:00:00'))  (cost=568302.27 rows=3113458) (actual time=93.689..546.476 rows=530 loops=1)
    -> Single-row index lookup on bi using PRIMARY (billid=a1.billid)  (cost=0.98 rows=1) (actual time=0.005..0.005 rows=1 loops=530)
```

优化的第二版SQL

```
explain
select  /*+no_merge() */
 t.*,bi.billcode from
(
select  profileid,billtype,billdate,billid from  a1
where a1.billtype IN( 604) and a1.billdate >='2025-03-14' and  a1.billdate
<='2025-03-15'
) t
inner join a1 bi on t.billid = bi.billid
```

强制用hint no_merge() 阻止MySQL的合并优化

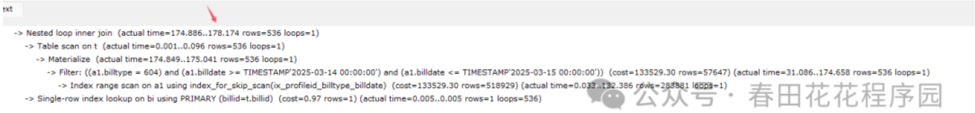


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	57655	100.00	NULL
1	PRIMARY	bi	NULL	eq_ref	PRIMARY,idx_yecx	PRIMARY	8	t.billid	1	100.00	NULL
2	DERIVED	a1	NULL	range	ix_profileid_billtype_billdate,...	ix_profileid_billtype_billdate	8	NULL	519004	11.11	Using where; Using index for skip scan

在执行计划中看到扫描行数少了，**也终于走到了 Index skip scan**

性能继续提升 从46327ms提升到了178ms 提升260倍



```
-> Nested loop inner join  (actual time=174.886..178.174 rows=536 loops=1)
    -> Table scan on t  (actual time=0.001..0.096 rows=536 loops=1)
        -> Materialize  (actual time=174.849..175.041 rows=536 loops=1)
            -> Filter: ((a1.billtype = 604) and (a1.billdate >= TIMESTAMP'2025-03-14 00:00:00') and (a1.billdate <= TIMESTAMP'2025-03-15 00:00:00'))  (cost=133529.30 rows=57647) (actual time=31.086..174.658 rows=536 loops=1)
                -> Index range scan on a1 using index_for_skip_scan(ix_profileid_billtype_billdate)  (cost=133529.30 rows=518929) (actual time=0.033..132.386 rows=283881 loops=1)
    -> Single-row index lookup on bi using PRIMARY (billid=t.billid)  (cost=0.97 rows=1) (actual time=0.005..0.005 rows=1 loops=536)
```

Index skip scan的要求

1、索引为复合索引
2、查询为单表查询
3、查询不使用GROUP BY或DISTINCT
4、查询仅引用索引中的列
5、索引为（a,b,c）条件用到了b,c 且b、c设计不能为null

贴上官方的说明

> Using this strategy decreases the number of accessed rows because MySQL skips the rows that do not qualify for each constructed range. This Skip Scan access method is

> applicable under the following conditions:
> Table T has at least one compound index with key parts of the form ([A_1, …, A_k,] B_1, …, B_m, C [, D_1, …, D_n]). Key parts A and D may be empty, but B and C must be nonempty.
> The query references only one table.
> The query does not use GROUP BY or DISTINCT.
> The query references only columns in the index.
> The predicates on A_1, …, A_k must be equality predicates and they must be constants. This includes the IN() operator.
> The query must be a conjunctive query; that is, an AND of OR conditions: (cond1(key_part1) OR cond2(key_part1)) AND (cond1(key_part2) OR …) AND …
> There must be a range condition on C.
> Conditions on D columns are permitted. Conditions on D must be in conjunction with the range condition on C.
> Use of Skip Scan is indicated in EXPLAIN output as follows:
> Using index for skip scan in the Extra column indicates that the loose index Skip Scan access method is used.
> If the index can be used for Skip Scan, the index should be visible in the possible_keys column.

range-access-skip-scan