```
 | rows | filtered | Extra                           |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+---------------------------------+
|  1 | SIMPLE      | t2    | NULL       | ALL  | NULL          | NULL | NULL    | NULL |   24 |   100.00 | Using temporary; Using filesort |
+----+-------------+-------+------------+------+---------------+------+---------+------+------+----------+---------------------------------+
1 row in set, 1 warning (0.00 sec)
```

从执行计划Extra: Using temporary可以看到这里使用到了临时表

* 使用索引的group by，没有使用到临时表

```
mysql> explain select c1,count(*) from t2 group by c1;
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+-------------+
| id | select_type | table | partitions | type  | possible_keys | key           | key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | t2    | NULL       | index | c1_c2_c3_idx  | c1_c2_c3_idx  | 768     | NULL |   24 |   100.00 | Using index |
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.01 sec)
```

Extra：Using index，且type为index，表示全索引扫描

这种情况下，如果表数据量很大，还是会比较耗时

有索引的情况
======

对于使用索引的情况，对于索引的访问有两种算法

* Loose Index Scan：松散索引扫描

+ 不需要扫描所有的索引key，根据分组前缀（group by字段）跳跃扫描部分
+ 执行计划Extra有：Using index for group-by
* Tight Index Scan：紧凑索引扫描

+ 需要扫描范围或全部的索引key
+ 执行计划Extra有：Using index

另外还有一种两种算法结合使用的方式：

* 在松散索引扫描的成本大于紧凑索引扫描时可能用到，后文说明

下面是两条SQL分别使用Loose Index Scan和Tight Index Scan：

```
mysql> explain SELECT c1,MIN(c2) FROM t2 GROUP BY c1;
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+--------------------------+
| id | select_type | table | partitions | type  | possible_keys | key           | key_len | ref  | rows | filtered | Extra                    |
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+--------------------------+
|  1 | SIMPLE      | t2    | NULL       | range | c1_c2_c3_idx  | c1_c2_c3_idx  | 256     | NULL |    7 |   100.00 | Using index for group-by |
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)
```

```
mysql> explain SELECT c1,COUNT(*) FROM t2 GROUP BY c1;
+----+-------------+-------+------------+-------+---------------
-------+-------------+---------+------+------+----------
+-------------+
| id | select_type | table | partitions | type  | possible_keys | key          |
key_len | ref  | rows | filtered | Extra       |
+----+-------------+-------+------------+-------+---------------
-------+-------------+---------+------+------+----------
+-------------+
|  1 | SIMPLE      | t2    | NULL       | index | c1_c2_c3_idx  | c1_c2_c3_idx |
768         (null,'y','d','y','y'), (null,'f','b','f','y');

-- 其数据分布
mysql> select c1,c2,count(*) from t2 group by c1,c2;
+----+----+----------+
| c1 | c2 | count(*) |
+----+----+----------+
| a  | b  |        3 |
| a  | d  |        1 |
| d  | b  |        2 |
| e  | b  |        2 |
| f  | b  |        2 |
| f  | c  |        2 |
| j  | b  |        1 |
| k  | c  |        3 |
| k  | g  |        1 |
| m  | b  |        1 |
| t  | c  |        1 |
| x  | b  |        1 |
| x  | d  |        1 |
| y  | d  |        2 |
| z  | c  |        1 |
+----+----+----------+
15 rows in set (0.00 sec)
```

执行计划Extra：Using index for group-by (scanning)

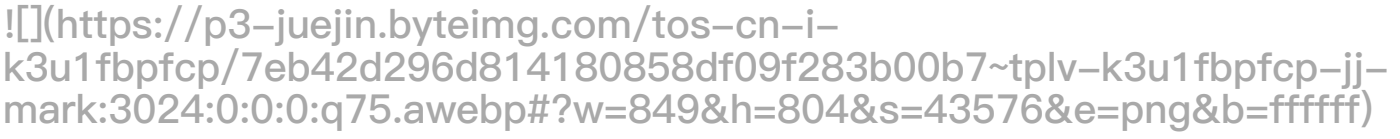
```
mysql> explain select count(distinct c1,c2) from t2;
+----+-------------+-------+------------+-------+---------------
-------+-------------+---------+------+------+----------
+-------------------------------------+
| id | select_type | table | partitions | type  | possible_keys | key          |
key_len | ref  | rows | filtered | Extra                               |
```

```
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+-----------------------------------------+
|  1 | SIMPLE      | t2    | NULL       | range | c1_c2_c3_idx  | c1_c2_c3_idx  | 512     | NULL |   16 |   100.00 | Using index for group-by (scanning) |
+----+-------------+-------+------------+-------+---------------+---------------+---------+------+------+----------+-----------------------------------------+
1 row in set, 1 warning (0.00 sec)
```

该方式可以理解为Loose Index Scan的扩展或两种方式的结合：

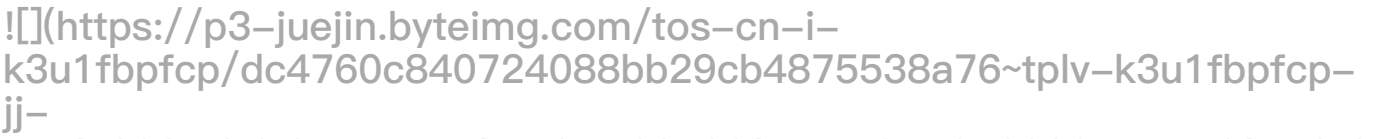
* 索引顺序扫描的同时进行去重

其执行的示意图如下：

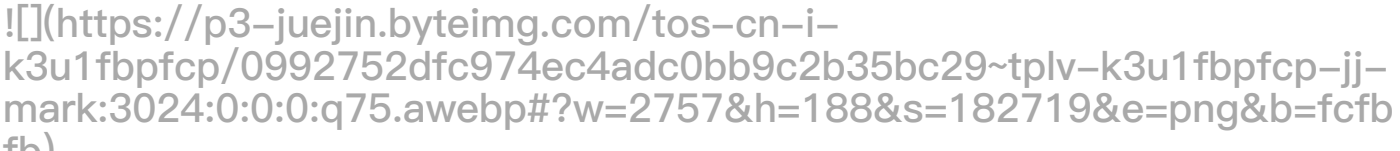


最后，再回到文章开头的案例，其执行计划如下：其核心就是将紧凑索引扫描转化为了松散索引扫描

* 优化前执行计划：



* 优化后执行计划：



小结
==

对于group by可以使用索引进行优化，Loose Index Scan相对于Tight Index Scan在一些情况下可以大大减少扫描的行数，使用Loose Index Scan时，执行计划Extra有：Using index for group-by

在Loose Index Scan的成本大于Tight Index 一些情况Loose Index Scan的一些情况下，可能可以用到两者的结合的方式，执行计划中Extra有：Using index for group-by (scanning)

Loose Index Scan更适用于分组内重复值相对较多，分组个数相对较少的情况

可根据实际情况进行优化

参考链接
====

[dev.mysql.com/doc/refman/...](http://cxyroad.com/”https://dev.mysql.com/doc/refman/5.7/en/group-by-optimization.html”)

[dev.mysql.com/blog-archiv...](http://cxyroad.com/”https://dev.mysql.com/blog-archive/what-is-the-scanning-variant-of-a-loose-index-scan/”)

[MySQL 怎么用索引实现 group by? -鸿蒙开发者社区-51CTO.COM](http://cxyroad.com/”https://ost.51cto.com/posts/11914”)

[www.oreilly.com/library/vie...](http://cxyroad.com/”https://www.oreilly.com/library/view/high-performance-mysql/9780596101718/ch04.html”)