知数堂公开课

最有良心、最有品质的在线培训品牌

QQ群：579036588

官网：http://zhishuedu.com

微信公众号：知数堂（izhishuedu）

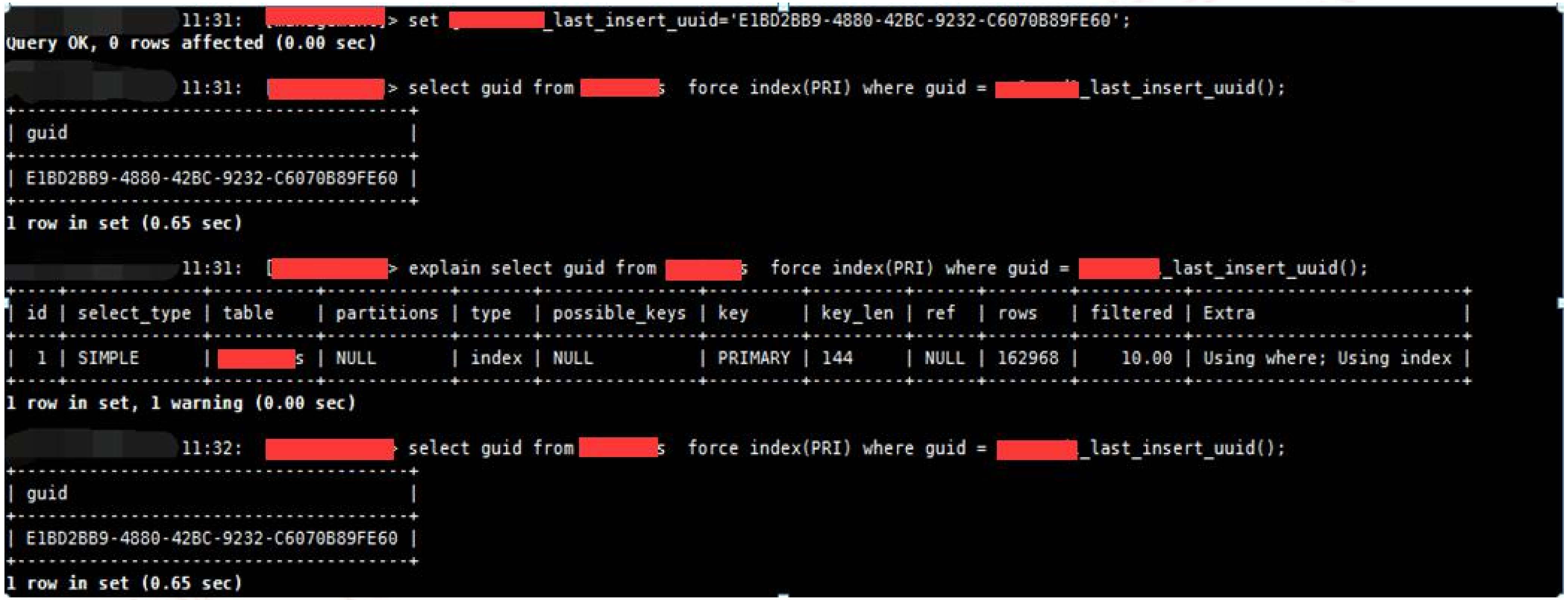
《经典SQL优化实例剖析》

知数堂培训公开课

郑松华

今天的案例

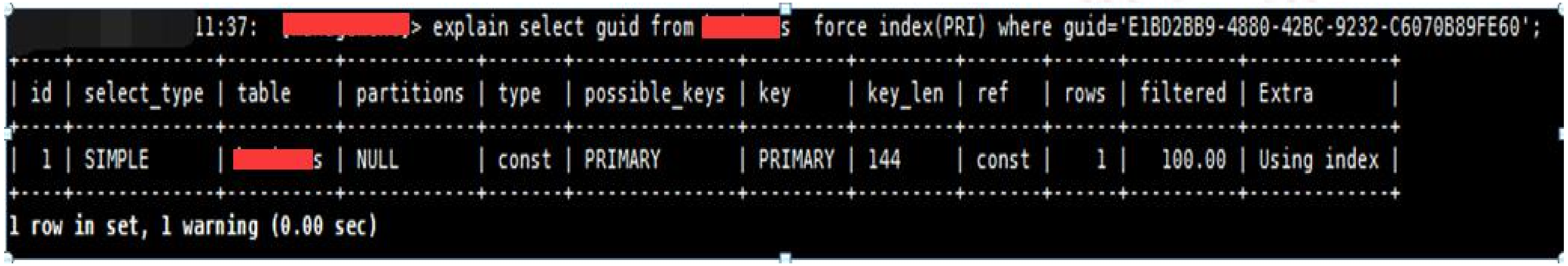
- 1 UDF 当作谓词情况下下优化案例



```
[redacted] 11:31: [redacted]> set [redacted]_last_insert_uuid='E1BD2BB9-4880-42BC-9232-C6070B89FE60';
Query OK, 0 rows affected (0.00 sec)

[redacted] 11:31: [redacted]> select guid from [redacted]s  force index(PRI) where guid = [redacted]_last_insert_uuid();
+--------------------------------------+
| guid                                 |
+--------------------------------------+
| E1BD2BB9-4880-42BC-9232-C6070B89FE60 |
+--------------------------------------+
1 row in set (0.65 sec)

[redacted] 11:31: [redacted]> explain select guid from [redacted]s  force index(PRI) where guid = [redacted]_last_insert_uuid();
+----+-------------+-----------+------------+-------+---------------+---------+---------+------+--------+----------+--------------------------+
| id | select_type | table     | partitions | type  | possible_keys | key     | key_len | ref  | rows   | filtered | Extra                    |
+----+-------------+-----------+------------+-------+---------------+---------+---------+------+--------+----------+--------------------------+
|  1 | SIMPLE      | [redacted]s | NULL     | index | NULL          | PRIMARY | 144     | NULL | 162968 |    10.00 | Using where; Using index |
+----+-------------+-----------+------------+-------+---------------+---------+---------+------+--------+----------+--------------------------+
1 row in set, 1 warning (0.00 sec)

[redacted] 11:32: [redacted]> select guid from [redacted]s  force index(PRI) where guid = [redacted]_last_insert_uuid();
+--------------------------------------+
| guid                                 |
+--------------------------------------+
| E1BD2BB9-4880-42BC-9232-C6070B89FE60 |
+--------------------------------------+
1 row in set (0.65 sec)
```

今天的案例

- 使用常数 效果非常好！



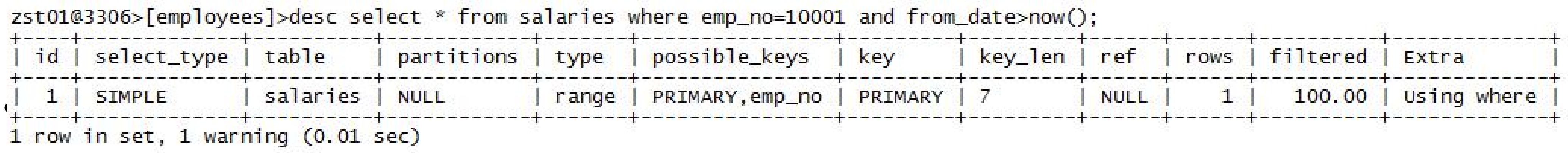
```
[redacted] 11:37: [redacted]> explain select guid from [redacted]s  force index(PRI) where guid='E1BD2BB9-4880-42BC-9232-C6070B89FE60';
+----+-------------+-----------+------------+-------+---------------+---------+---------+-------+------+----------+-------------+
| id | select_type | table     | partitions | type  | possible_keys | key     | key_len | ref   | rows | filtered | Extra       |
+----+-------------+-----------+------------+-------+---------------+---------+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | [redacted]s | NULL       | const | PRIMARY       | PRIMARY | 144     | const |    1 |   100.00 | Using index |
+----+-------------+-----------+------------+-------+---------------+---------+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

今天的案例

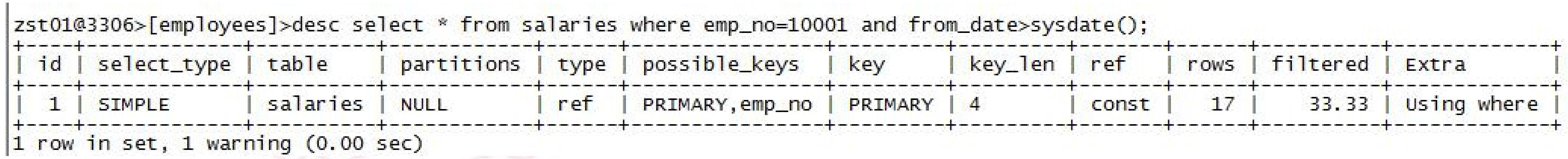
- 一看 常数效果 非常好 问题就非常明确了 是因为where条件了的谓词是函数，是时刻变化的 无法给定具体的值，导致执行计划变差
- 跟 sysdate 特性 非常像！

时间函数 now , sysdate

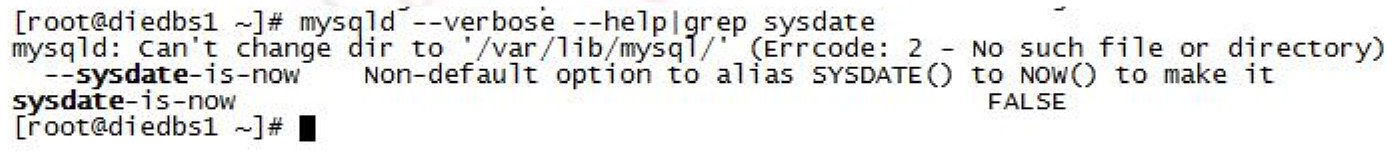
- desc select * from salaries where emp_no=10001 and from_date>now();



```
zst01@3306>[employees]>desc select * from salaries where emp_no=10001 and from_date>now();
+----+-------------+----------+------------+-------+----------------+---------+---------+------+------+----------+-------------+
| id | select_type | table    | partitions | type  | possible_keys  | key     | key_len | ref  | rows | filtered | Extra       |
+----+-------------+----------+------------+-------+----------------+---------+---------+------+------+----------+-------------+
|  1 | SIMPLE      | salaries | NULL       | range | PRIMARY,emp_no | PRIMARY | 7       | NULL |    1 |   100.00 | Using where |
+----+-------------+----------+------------+-------+----------------+---------+---------+------+------+----------+-------------+
1 row in set, 1 warning (0.01 sec)
```



```
zst01@3306>[employees]>desc select * from salaries where emp_no=10001 and from_date>sysdate();
+----+-------------+----------+------------+------+----------------+---------+---------+-------+------+----------+-------------+
| id | select_type | table    | partitions | type | possible_keys  | key     | key_len | ref   | rows | filtered | Extra       |
+----+-------------+----------+------------+------+----------------+---------+---------+-------+------+----------+-------------+
|  1 | SIMPLE      | salaries | NULL       | ref  | PRIMARY,emp_no | PRIMARY | 4       | const |   17 |    33.33 | Using where |
+----+-------------+----------+------------+------+----------------+---------+---------+-------+------+----------+-------------+
1 row in set, 1 warning (0.00 sec)
```

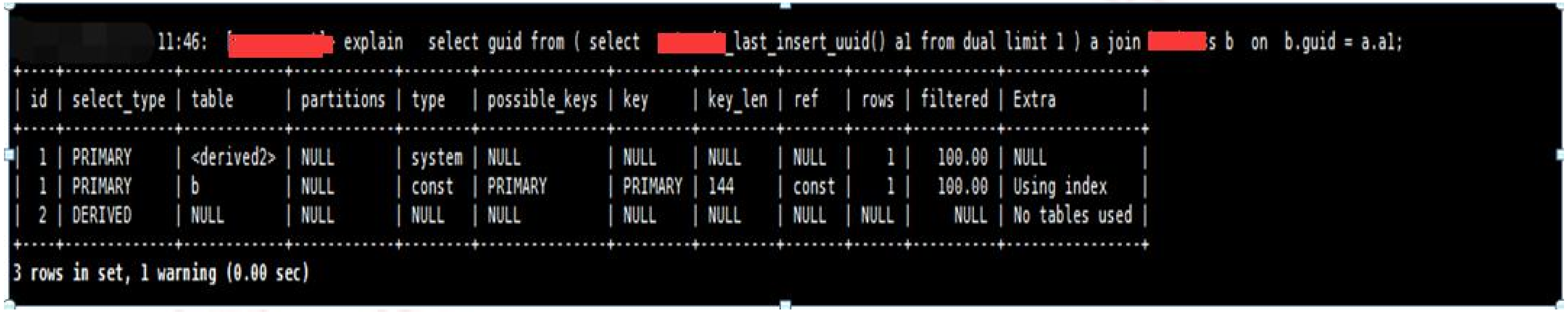


```
[root@diedbs1 ~]# mysqld --verbose --help|grep sysdate
mysqld: Can't change dir to '/var/lib/mysql/' (Errcode: 2 - No such file or directory)
  --sysdate-is-now    Non-default option to alias SYSDATE() to NOW() to make it
sysdate-is-now                                          FALSE
[root@diedbs1 ~]# █
```

最终修改的SQL

我们知道了原理 我们就用子查询把动态变化的函数，变成静态化，常数化 然后进行Join

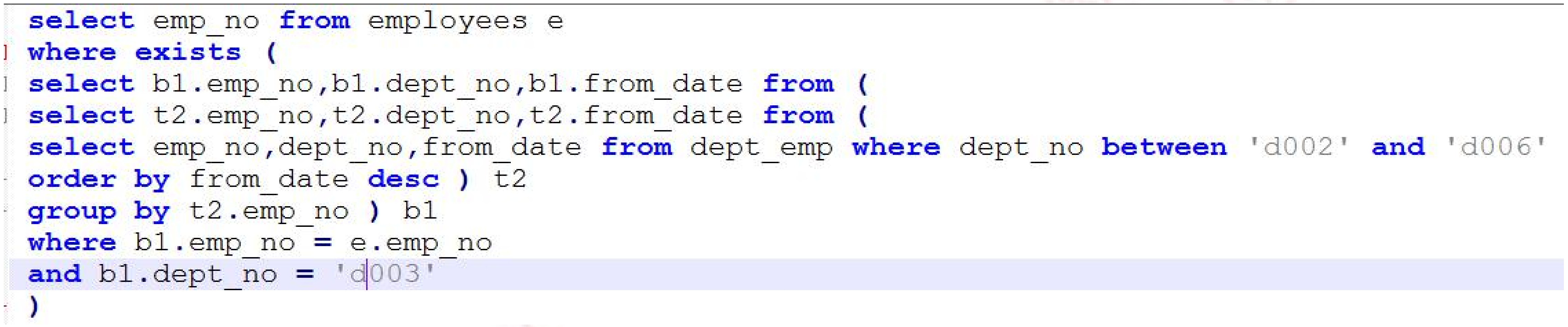
达到优化的效果!



```
[redacted] 11:46: [redacted] explain  select guid from ( select [redacted]_last_insert_uuid() a1 from dual limit 1 ) a join [redacted]s b  on  b.guid = a.a1;
+----+-------------+------------+------------+--------+---------------+---------+---------+-------+------+----------+-----------------+
| id | select_type | table      | partitions | type   | possible_keys | key     | key_len | ref   | rows | filtered | Extra           |
+----+-------------+------------+------------+--------+---------------+---------+---------+-------+------+----------+-----------------+
|  1 | PRIMARY     | <derived2> | NULL       | system | NULL          | NULL    | NULL    | NULL  |    1 |   100.00 | NULL            |
|  1 | PRIMARY     | b          | NULL       | const  | PRIMARY       | PRIMARY | 144     | const |    1 |   100.00 | Using index     |
|  2 | DERIVED     | NULL       | NULL       | NULL   | NULL          | NULL    | NULL    | NULL  | NULL |     NULL | No tables used  |
+----+-------------+------------+------------+--------+---------------+---------+---------+-------+------+----------+-----------------+
3 rows in set, 1 warning (0.00 sec)
```

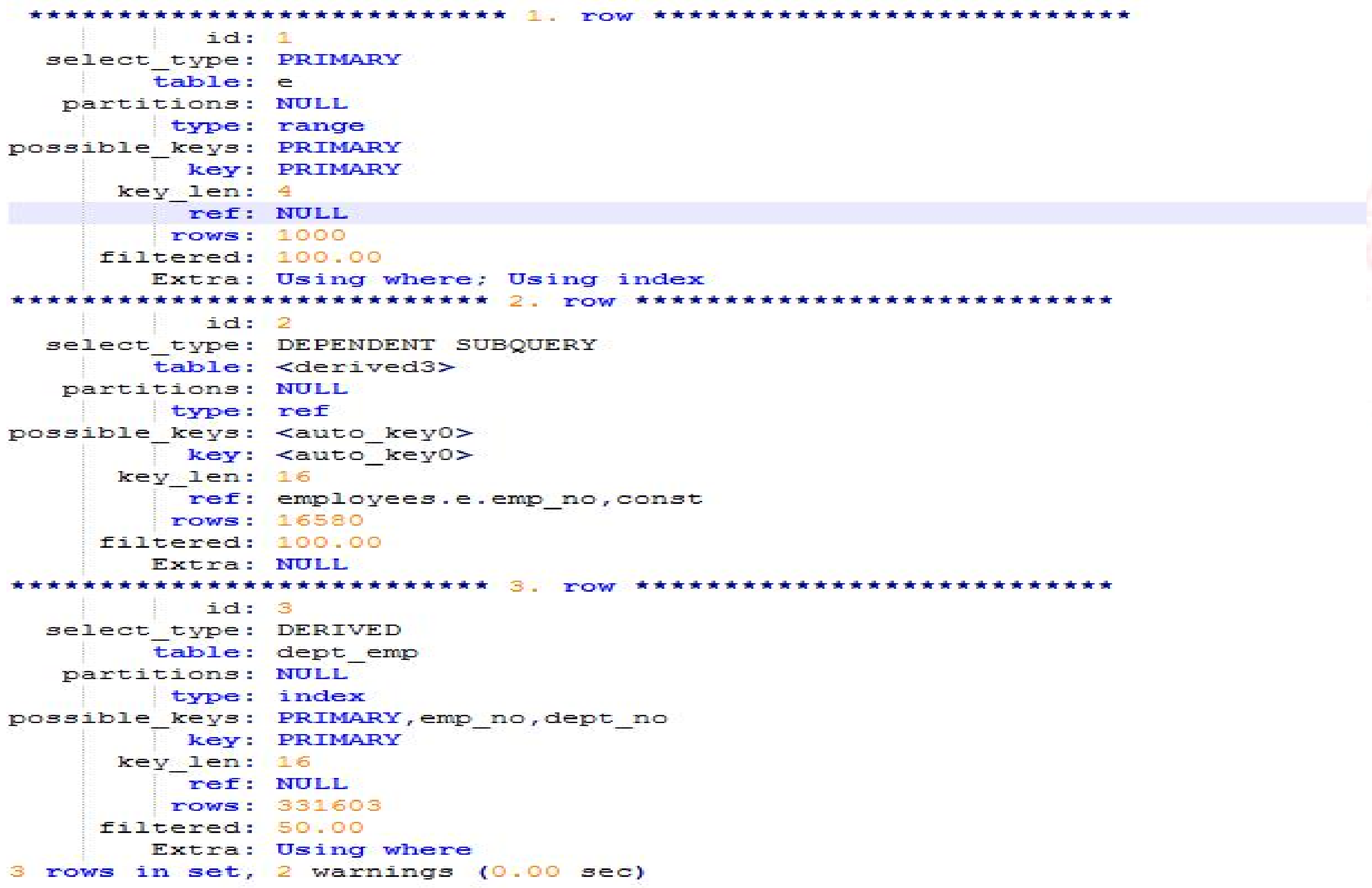
今天的案例

- 2 下面的sql 是根据问题sql 核心部分重现



```
select emp_no from employees e
where exists (
select b1.emp_no,b1.dept_no,b1.from_date from (
select t2.emp_no,t2.dept_no,t2.from_date from (
select emp_no,dept_no,from_date from dept_emp where dept_no between 'd002' and 'd006'
order by from_date desc ) t2
group by t2.emp_no ) b1
where b1.emp_no = e.emp_no
and b1.dept_no = 'd003'
)
```

执行计划如下



```
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: e
   partitions: NULL
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 1000
     filtered: 100.00
        Extra: Using where; Using index
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: <derived3>
   partitions: NULL
         type: ref
possible_keys: <auto_key0>
          key: <auto_key0>
      key_len: 16
          ref: employees.e.emp_no,const
         rows: 16580
     filtered: 100.00
        Extra: NULL
*************************** 3. row ***************************
           id: 3
  select_type: DERIVED
        table: dept_emp
   partitions: NULL
         type: index
possible_keys: PRIMARY,emp_no,dept_no
          key: PRIMARY
      key_len: 16
          ref: NULL
         rows: 331603
     filtered: 50.00
        Extra: Using where
3 rows in set, 2 warnings (0.00 sec)
```

从执行计划看主要性能消耗点

- 1 怎么读这个执行计划 ？
- 2 从执行计划中发现主要消耗点在哪里 ？

1 怎么读这个执行计划 ？

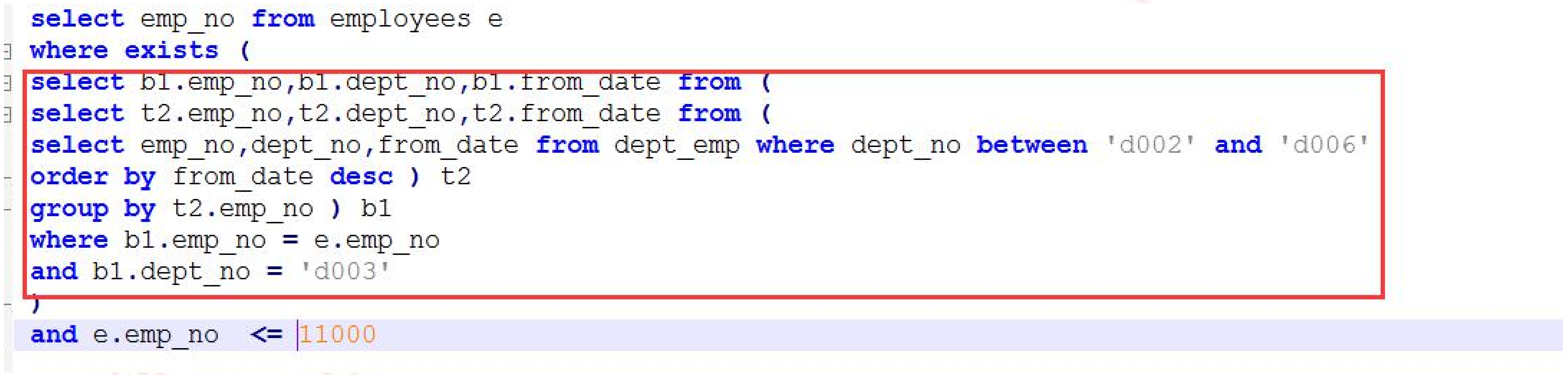
- 1 mysql 现在只支持nested loop join
- 2 从上到下 第一行开始读
- 3 table 列里的值含有 < derived+id> 就先跳到对应的id中 然后返回
- 4 type 列中的 range ， ref ， index 分别表示索引范围扫描 ， join 中等号连接 不能保证唯一性 ，索引全扫描
- 5 rows 是预估的行数 越少越好！

2 从执行计划中发现主要消耗点在哪里 ？

- 1 根据上面讲的内容 这个执行计划分析如下
- 先对e 表进行 range 扫描 然后 和 dept_emp 的结果集 进行 nested loop join
- 2 因为id 3 进行了 index 扫描 所以导致 rows 很大 所以优化策略出来了 想尽办法对 id=3 rows减少！

怎么减少 rows ？

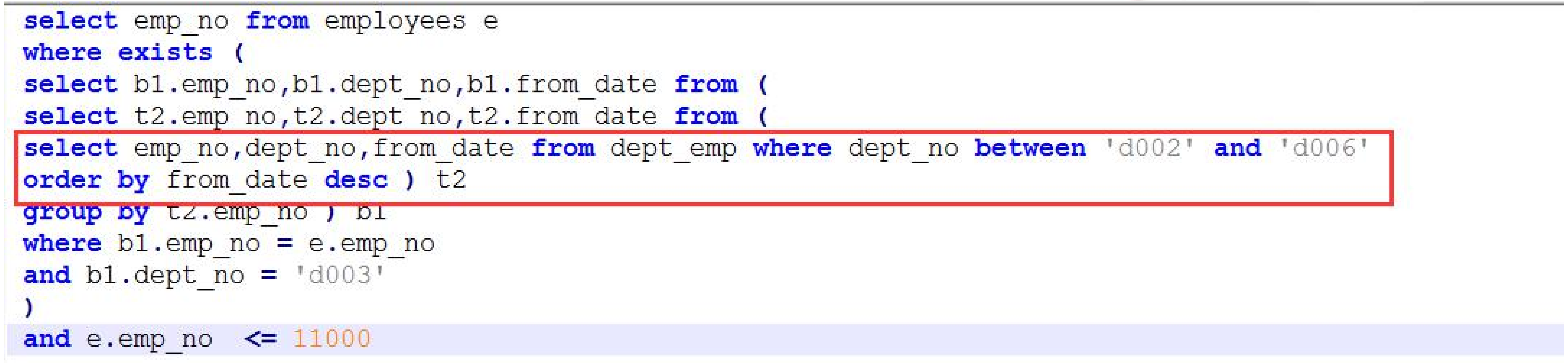
- 1 我们在来分析这个sql
- 红线里面就是id=3的部分
- 这部分开发到底想表达什么 ？



```
select emp_no from employees e
where exists (
select b1.emp_no,b1.dept_no,b1.from_date from (
select t2.emp_no,t2.dept_no,t2.from_date from (
select emp_no,dept_no,from_date from dept_emp where dept_no between 'd002' and 'd006'
order by from_date desc ) t2
group by t2.emp_no ) b1
where b1.emp_no = e.emp_no
and b1.dept_no = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

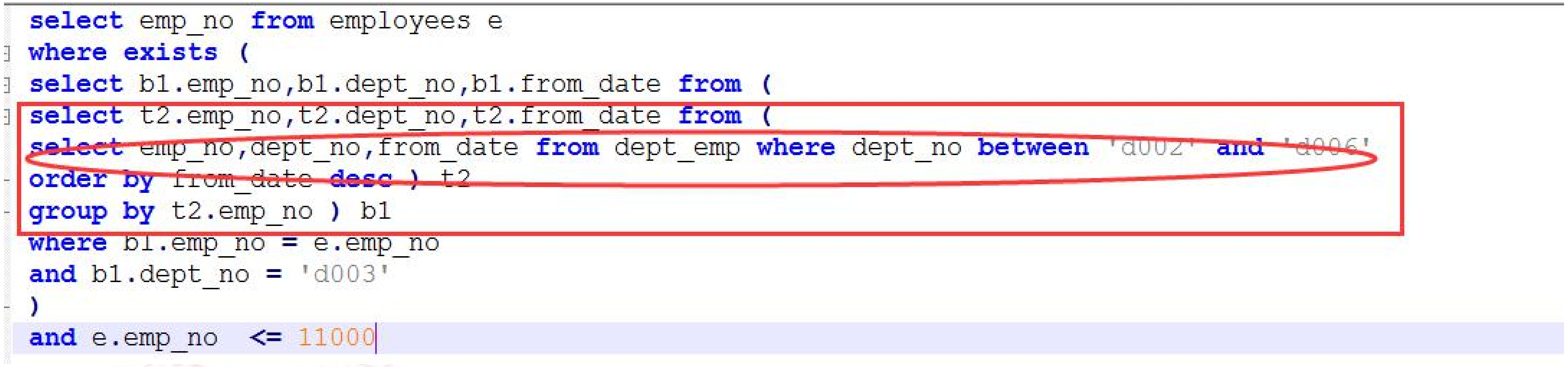
- 1 从最里面的视图开始说起
- 2 这部分表达的是 经过where 条件过滤之后根据 From_date 进行倒序排序！



```
select emp_no from employees e
where exists (
select b1.emp_no,b1.dept_no,b1.from_date from (
select t2.emp_no,t2.dept_no,t2.from_date from (
select emp_no,dept_no,from_date from dept_emp where dept_no between 'd002' and 'd006'
order by from_date desc ) t2
group by t2.emp_no ) b1
where b1.emp_no = e.emp_no
and b1.dept_no = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

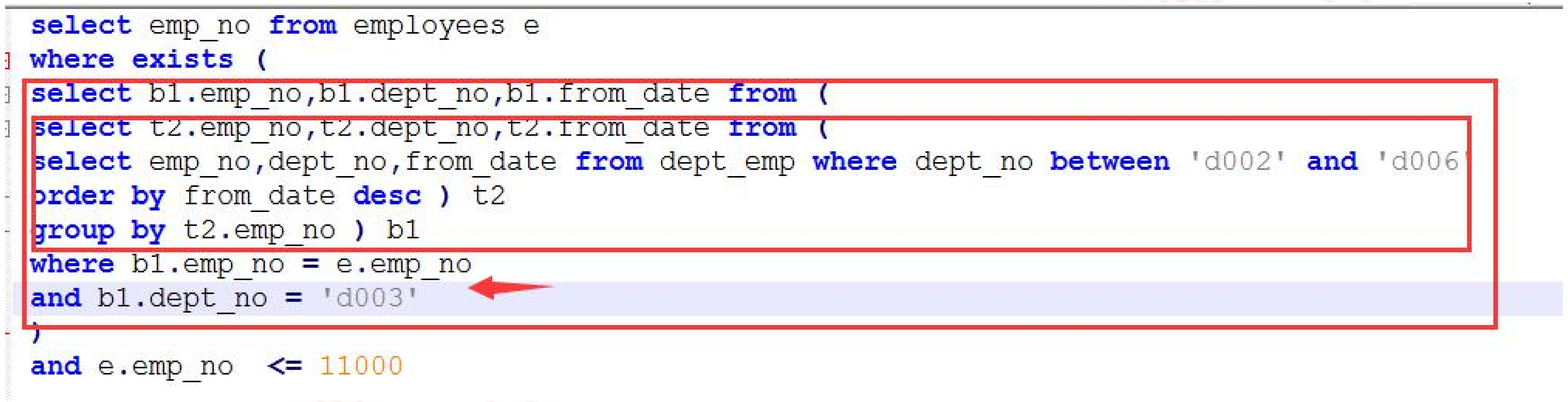
- 1 在红框以内椭圆形以外的部分
- 2 然后通过emp_no 进行了分组 ！
- 3 这是5.6的时候是可以的但5.7报错！
- 4 相当于 求 emp_no为唯一值 from_date 为 最大值的那一行 跟oracle 的 row_number over 类似



```
select emp_no from employees e
where exists (
select b1.emp_no,b1.dept_no,b1.from_date from (
select t2.emp_no,t2.dept_no,t2.from_date from (
select emp_no,dept_no,from_date from dept_emp where dept_no between 'd002' and 'd006'
order by from_date desc ) t2
group by t2.emp_no ) b1
where b1.emp_no = e.emp_no
and b1.dept_no = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

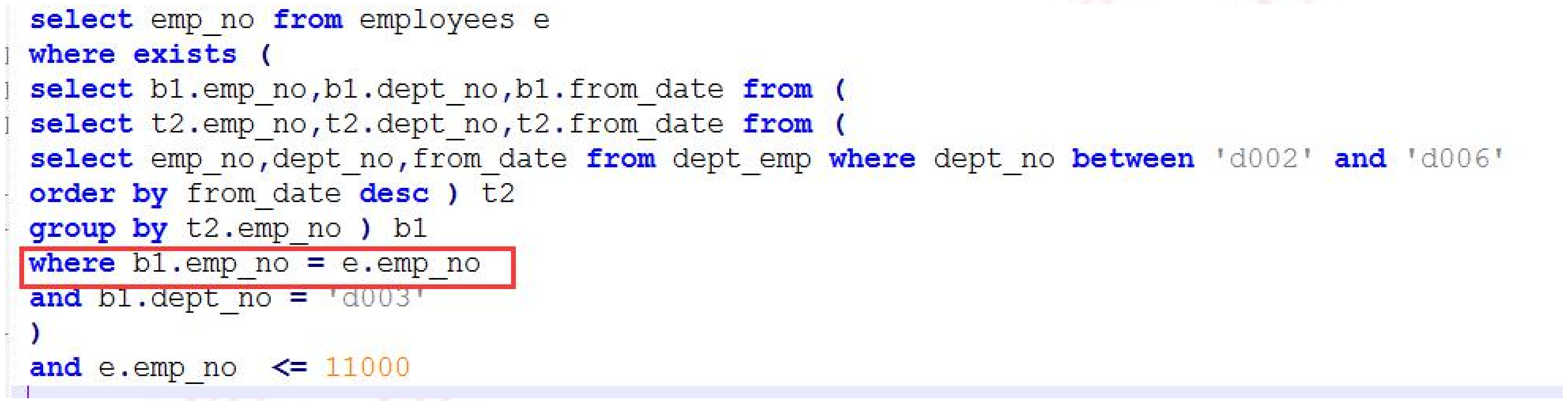
- 1 红箭头部分表示 对from_date为最大值的行如果dept_no为 d003 就满足条件！



```
select emp_no from employees e
where exists (
select b1.emp_no,b1.dept_no,b1.from_date from (
select t2.emp_no,t2.dept_no,t2.from_date from (
select emp_no,dept_no,from_date from dept_emp where dept_no between 'd002' and 'd006'
order by from_date desc ) t2
group by t2.emp_no ) b1
where b1.emp_no = e.emp_no
and b1.dept_no = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

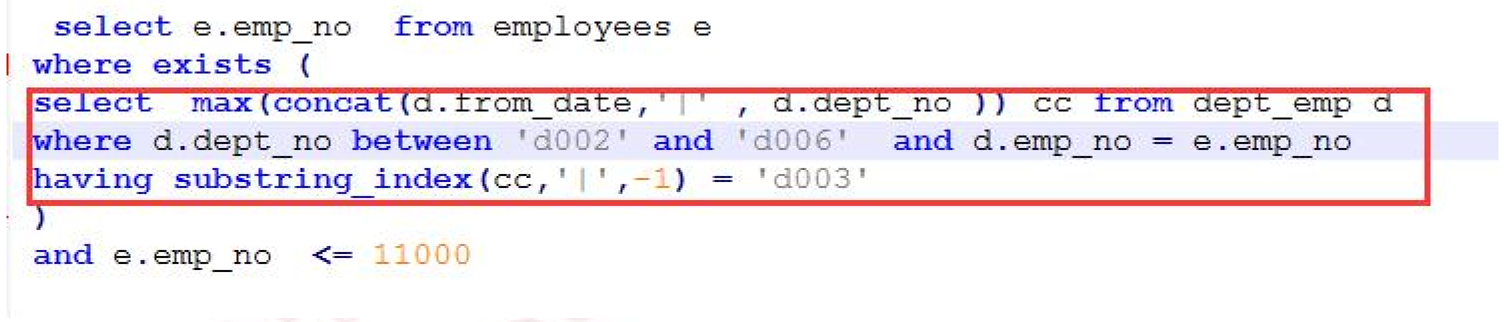
- 1 我们想把rows 减少就必须得把红框部分的条件塞进去 ！



```
select emp_no from employees e
where exists (
select b1.emp_no,b1.dept_no,b1.from_date from (
select t2.emp_no,t2.dept_no,t2.from_date from (
select emp_no,dept_no,from_date from dept_emp where dept_no between 'd002' and 'd006'
order by from_date desc ) t2
group by t2.emp_no ) b1
where b1.emp_no = e.emp_no
and b1.dept_no = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

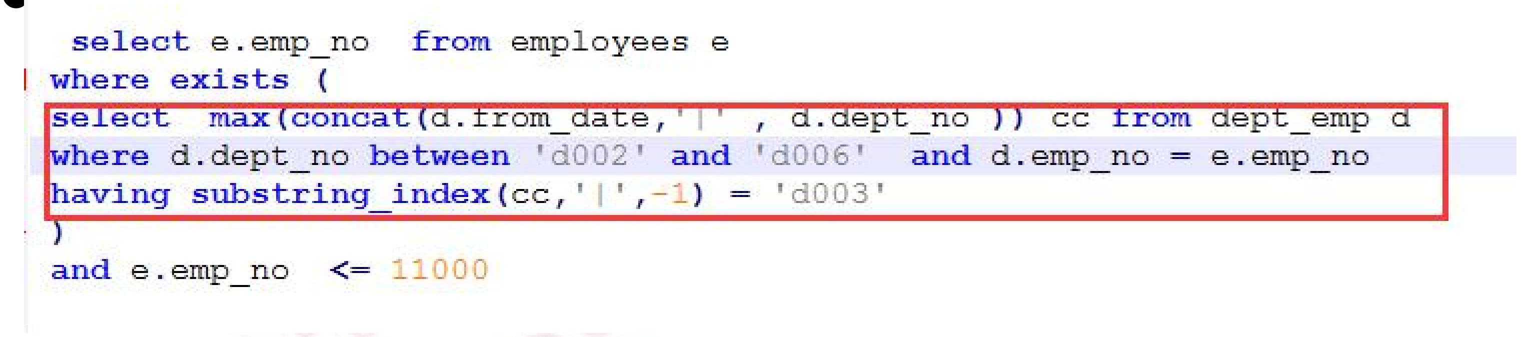
- 1 利用max having 改写sql 达到跟原来的语意一样的sql
- 我们通过 concat 合并 两个列之后 利用max
- 进行筛选 然后利用having 的特点进行过滤！



```
 select e.emp_no  from employees e
where exists (
select  max(concat(d.from_date,'|' , d.dept_no )) cc from dept_emp d
where d.dept_no between 'd002' and 'd006'  and d.emp_no = e.emp_no
having substring_index(cc,'|',-1) = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

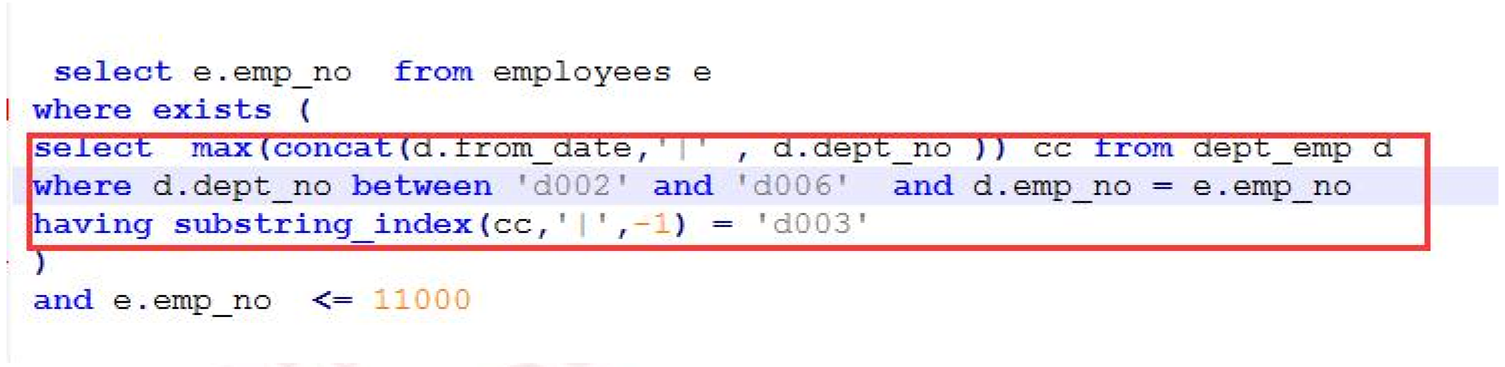
- 1 为什么去掉group by了 ？
- 因为 在where 条件上有了 emp_no 就可以去掉！
- 2 为什么去掉 order by desc 了 ？
- 因为 [illegible] 就可以代替！



```
 select e.emp_no  from employees e
where exists (
select  max(concat(d.from_date,'|' , d.dept_no )) cc from dept_emp d
where d.dept_no between 'd002' and 'd006'  and d.emp_no = e.emp_no
having substring_index(cc,'|',-1) = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

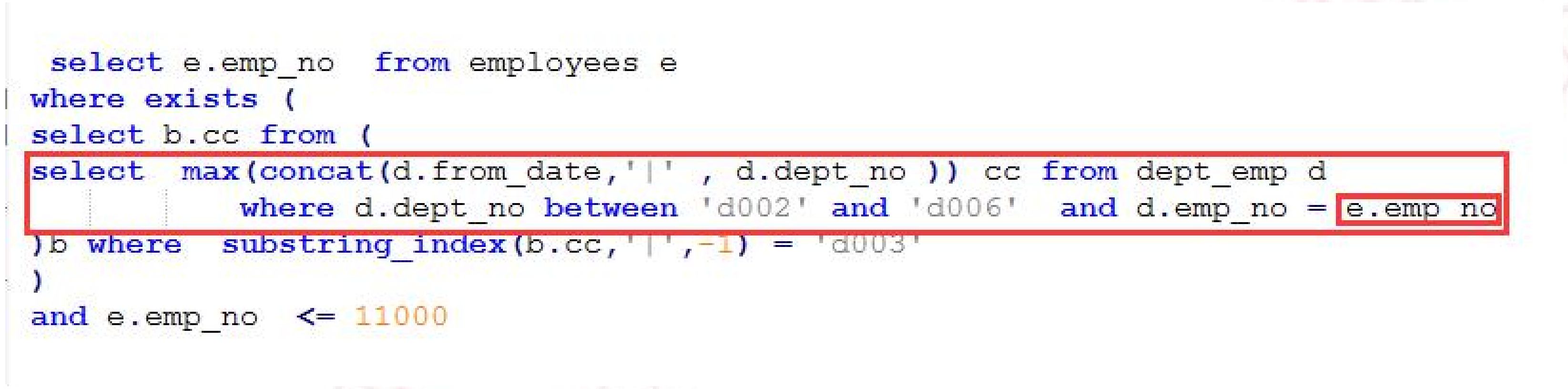
- 3 为什么要使用having ？
- 因为不想再嵌套一层子查询了
- 如果不用 having 用子查询是什么情况 ？



```
 select e.emp_no  from employees e
where exists (
select  max(concat(d.from_date,'|' , d.dept_no )) cc from dept_emp d
where d.dept_no between 'd002' and 'd006'  and d.emp_no = e.emp_no
having substring_index(cc,'|',-1) = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

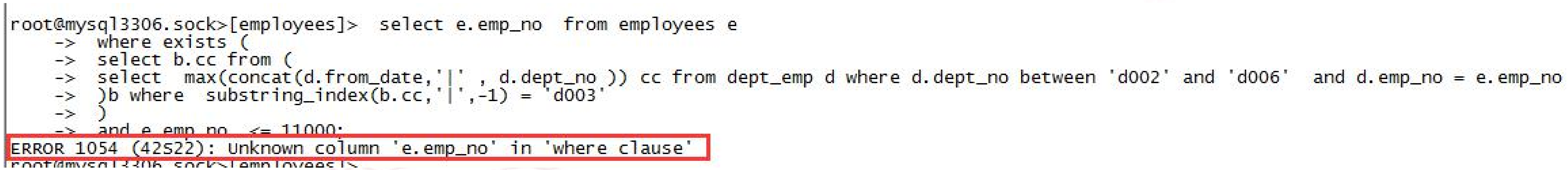
- 如果不用 having 用子查询是什么情况 ？
- 改写成如下　看样子没啥问题！



```
 select e.emp_no  from employees e
where exists (
select b.cc from (
select  max(concat(d.from_date,'|' , d.dept_no )) cc from dept_emp d
           where d.dept_no between 'd002' and 'd006'  and d.emp_no = e.emp_no
)b where  substring_index(b.cc,'|',-1) = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

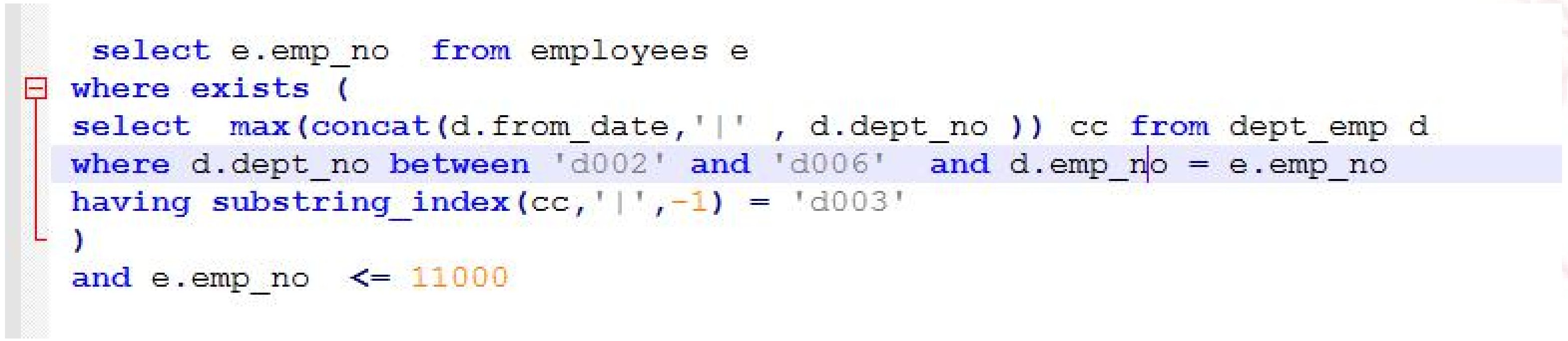
- 如果不用 having 用子查询是什么情况 ？
- 直接报错了！ Mysql 不支持这种语法！



```
root@mysql3306.sock>[employees]>  select e.emp_no  from employees e
    ->  where exists (
    ->  select b.cc from (
    ->  select  max(concat(d.from_date,'|' , d.dept_no )) cc from dept_emp d where d.dept_no between 'd002' and 'd006'  and d.emp_no = e.emp_no
    ->  )b where  substring_index(b.cc,'|',-1) = 'd003'
    ->  )
    ->  and e.emp_no  <= 11000;
ERROR 1054 (42S22): Unknown column 'e.emp_no' in 'where clause'
```

怎么减少 rows ？

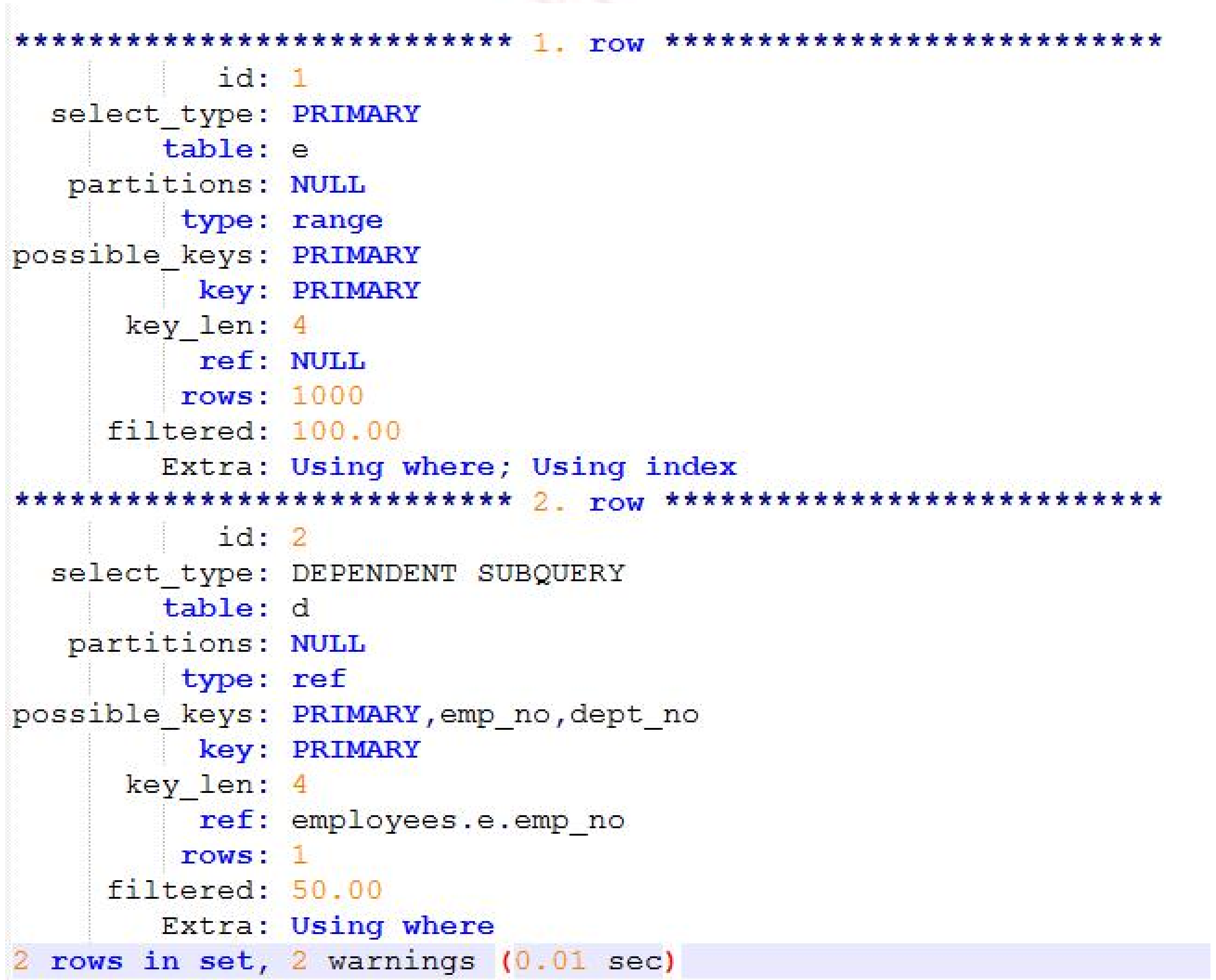
修改之后的SQL



```
 select e.emp_no  from employees e
where exists (
select  max(concat(d.from_date,'|' , d.dept_no )) cc from dept_emp d
where d.dept_no between 'd002' and 'd006'  and d.emp_no = e.emp_no
having substring_index(cc,'|',-1) = 'd003'
)
and e.emp_no  <= 11000
```

怎么减少 rows ？

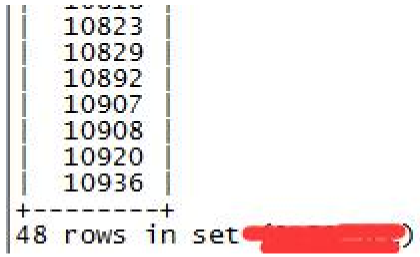
查看修改之后的执行计划



```
*************************** 1. row ***************************
           id: 1
  select_type: PRIMARY
        table: e
   partitions: NULL
         type: range
possible_keys: PRIMARY
          key: PRIMARY
      key_len: 4
          ref: NULL
         rows: 1000
     filtered: 100.00
        Extra: Using where; Using index
*************************** 2. row ***************************
           id: 2
  select_type: DEPENDENT SUBQUERY
        table: d
   partitions: NULL
         type: ref
possible_keys: PRIMARY,emp_no,dept_no
          key: PRIMARY
      key_len: 4
          ref: employees.e.emp_no
         rows: 1
     filtered: 50.00
        Extra: Using where
2 rows in set, 2 warnings (0.01 sec)
```

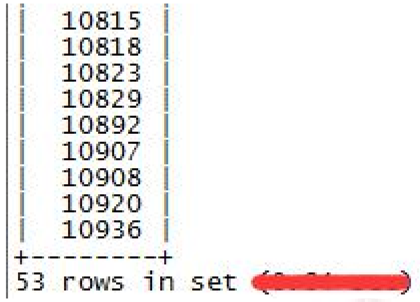
检验数据

1 运行原来的 sql



```
|   10823 |
|   10829 |
|   10892 |
|   10907 |
|   10908 |
|   10920 |
|   10936 |
+---------+
48 rows in set [redacted]
```

2 我们修改的sql



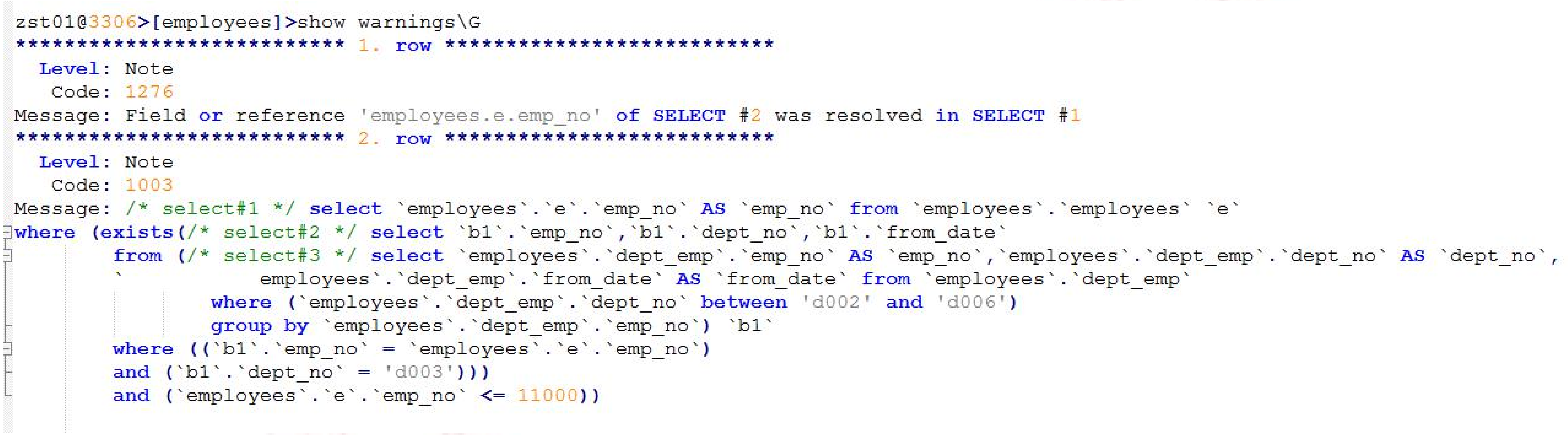
```
|   10815 |
|   10818 |
|   10823 |
|   10829 |
|   10892 |
|   10907 |
|   10908 |
|   10920 |
|   10936 |
+---------+
53 rows in set [redacted]
```

数据怎么不对了 ？ 到底哪里出问题了？

检验数据

1 原SQL的 转换之后的SQL

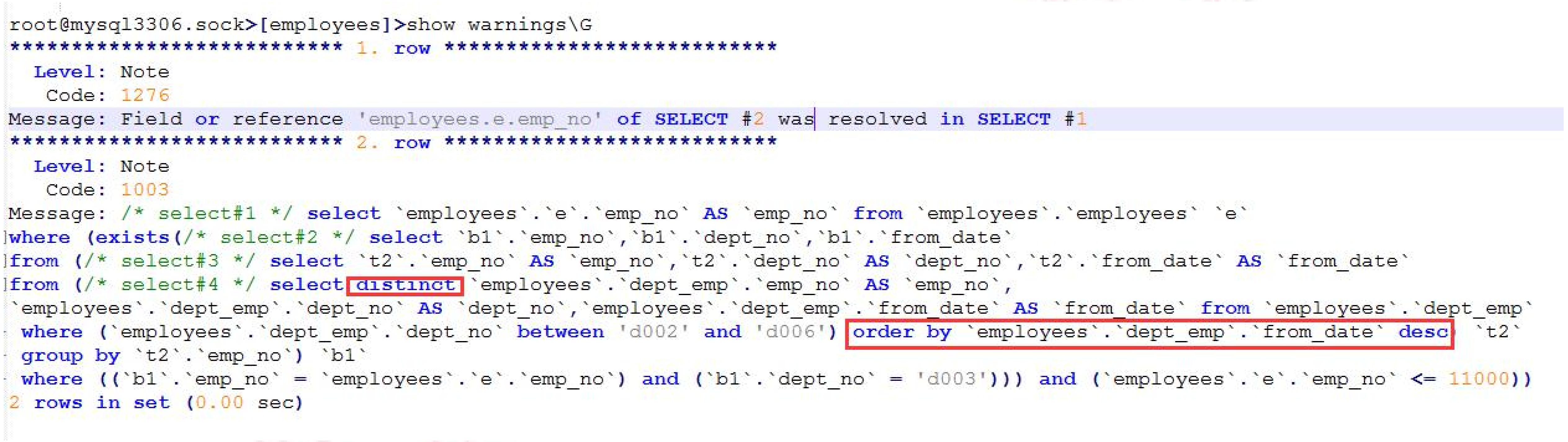
发现了 原来的order by 没了 ！



```
zst01@3306>[employees]>show warnings\G
*************************** 1. row ***************************
  Level: Note
   Code: 1276
Message: Field or reference 'employees.e.emp_no' of SELECT #2 was resolved in SELECT #1
*************************** 2. row ***************************
  Level: Note
   Code: 1003
Message: /* select#1 */ select `employees`.`e`.`emp_no` AS `emp_no` from `employees`.`employees` `e`
where (exists(/* select#2 */ select `b1`.`emp_no`,`b1`.`dept_no`,`b1`.`from_date`
        from (/* select#3 */ select `employees`.`dept_emp`.`emp_no` AS `emp_no`,`employees`.`dept_emp`.`dept_no` AS `dept_no`,
        `            employees`.`dept_emp`.`from_date` AS `from_date` from `employees`.`dept_emp`
                where (`employees`.`dept_emp`.`dept_no` between 'd002' and 'd006')
                group by `employees`.`dept_emp`.`emp_no`) `b1`
        where ((`b1`.`emp_no` = `employees`.`e`.`emp_no`)
        and (`b1`.`dept_no` = 'd003')))
        and (`employees`.`e`.`emp_no` <= 11000))
```

检验数据

在原来的SQL中添加distinct 之后的结果发现含有order by ！

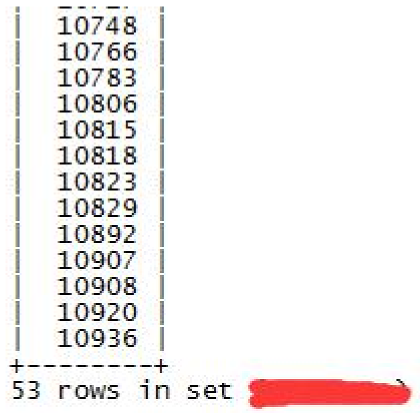


```
root@mysql3306.sock>[employees]>show warnings\G
*************************** 1. row ***************************
  Level: Note
   Code: 1276
Message: Field or reference 'employees.e.emp_no' of SELECT #2 was resolved in SELECT #1
*************************** 2. row ***************************
  Level: Note
   Code: 1003
Message: /* select#1 */ select `employees`.`e`.`emp_no` AS `emp_no` from `employees`.`employees` `e`
where (exists(/* select#2 */ select `b1`.`emp_no`,`b1`.`dept_no`,`b1`.`from_date`
from (/* select#3 */ select `t2`.`emp_no` AS `emp_no`,`t2`.`dept_no` AS `dept_no`,`t2`.`from_date` AS `from_date`
from (/* select#4 */ select distinct `employees`.`dept_emp`.`emp_no` AS `emp_no`,
`employees`.`dept_emp`.`dept_no` AS `dept_no`,`employees`.`dept_emp`.`from_date` AS `from_date` from `employees`.`dept_emp`
 where (`employees`.`dept_emp`.`dept_no` between 'd002' and 'd006') order by `employees`.`dept_emp`.`from_date` desc) `t2`
 group by `t2`.`emp_no`) `b1`
 where ((`b1`.`emp_no` = `employees`.`e`.`emp_no`) and (`b1`.`dept_no` = 'd003'))) and (`employees`.`e`.`emp_no` <= 11000))
2 rows in set (0.00 sec)
```

检验数据

运行之后的结果 跟我们修改之后的结果一样！

发现 原来的SQL是有问题的！



```
|   10748 |
|   10766 |
|   10783 |
|   10806 |
|   10815 |
|   10818 |
|   10823 |
|   10829 |
|   10892 |
|   10907 |
|   10908 |
|   10920 |
|   10936 |
+---------+
53 rows in set [redacted]
```

知数堂培训是由资深MySQL专家叶金荣、吴炳锡联合推出专业优质在线培训课程，目前主要有MySQL DBA实战优化和Python运维开发两个课程，是业内最有良心、最有品质的培训课程。

现有课程
MySQL DBA 实战班
MySQL DBA 优化班
Python运维开发班
SQL开发优化班