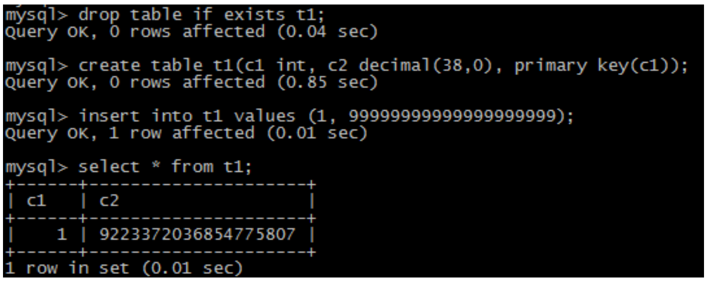
1. 针对一个 decimal 类型属性，在插入数据时（未超精度范围），插入结果会出错。示例如下：

drop table if exists t1;

create table t1(c1 int, c2 decimal(38,0), primary key(c1));

insert into t1 values (1, 99999999999999999999);

select * from t1;



```
mysql> drop table if exists t1;
Query OK, 0 rows affected (0.04 sec)

mysql> create table t1(c1 int, c2 decimal(38,0), primary key(c1));
Query OK, 0 rows affected (0.85 sec)

mysql> insert into t1 values (1, 99999999999999999999);
Query OK, 1 row affected (0.01 sec)

mysql> select * from t1;
+------+---------------------+
| c1   | c2                  |
+------+---------------------+
|    1 | 9223372036854775807 |
+------+---------------------+
1 row in set (0.01 sec)
```

结果错误（**已知错误，其使用手册中有说明**）

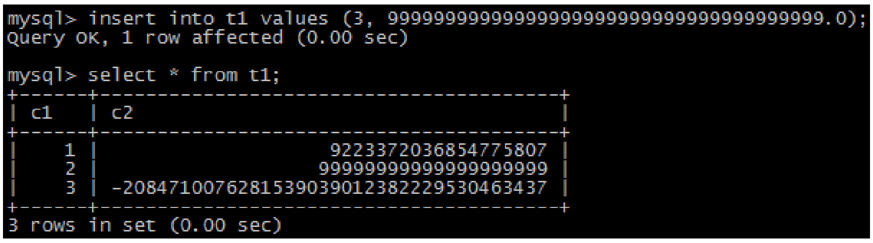
按照其使用手册的使用方法，在上述数据后面加上小数，就可正常插入，但是实际执行结果依然错误。

insert into t1 values (2, 99999999999999999999.0);

select * from t1;

insert into t1 values (3, 99999999999999999999999999999999999999.0);

select * from t1;



```
mysql> insert into t1 values (3, 99999999999999999999999999999999999999.0);
Query OK, 1 row affected (0.00 sec)

mysql> select * from t1;
+------+----------------------------------------+
| c1   | c2                                     |
+------+----------------------------------------+
|    1 |                    9223372036854775807 |
|    2 |                   99999999999999999999 |
|    3 | -2084710076281539039012382229530463437 |
+------+----------------------------------------+
3 rows in set (0.00 sec)
```

该错误已解决

2. 针对一个 decimal 类型属性，插入一个整数部分超过精度范围的数据，未报错，没有越界判断。示例如下：

drop table if exists t1;

create table t1(c1 int, c2 decimal(38,0), primary key(c1));

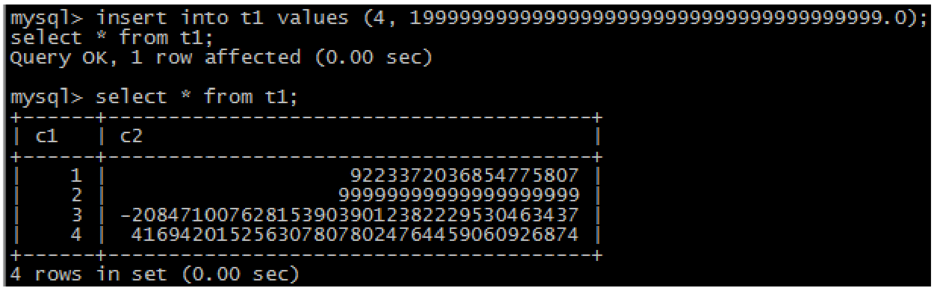
insert into t1 values (1, 99999999999999999999);

insert into t1 values (2, 99999999999999999999.0);

insert into t1 values (3, 99999999999999999999999999999999999999.0);

insert into t1 values (4, 199999999999999999999999999999999999999.0);

select * from t1;

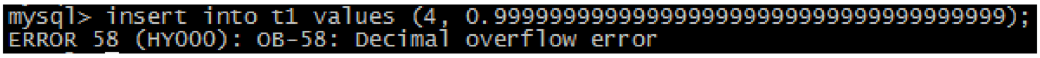


```
mysql> insert into t1 values (4, 1999999999999999999999999999999999999999.0);
select * from t1;
Query OK, 1 row affected (0.00 sec)

mysql> select * from t1;
+------+----------------------------------------+
| c1   | c2                                     |
+------+----------------------------------------+
|    1 |                    9223372036854775807 |
|    2 |                   99999999999999999999 |
|    3 |  -2084710076281539039012382229530463437 |
|    4 |   4169420152563078078024764459060926874 |
+------+----------------------------------------+
4 rows in set (0.00 sec)
```

该错误已解决

3. 针对一个 decimal 类型属性，插入一个小数位数较多但未超过其小数位精度的数据时，会报异常，正常情况应该可以执行。示例如下：

drop table if exists t1;
create table t1(c1 int, c2 decimal(37,37), primary key(c1));
insert into t1 values (4, 0.9999999999999999999999999999999999999); （小数位为 37 个 9）

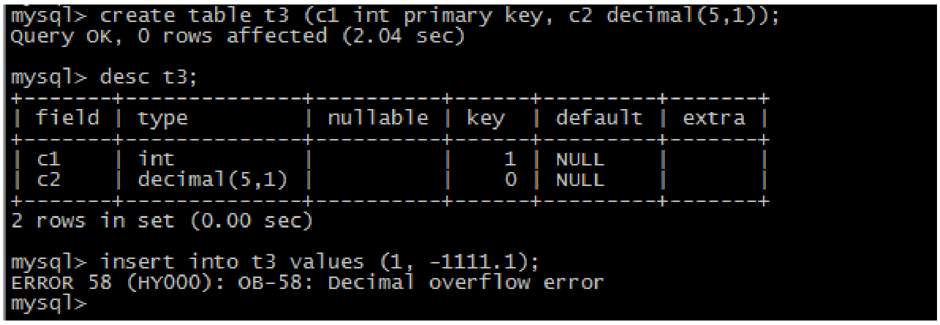


```
mysql> insert into t1 values (4, 0.9999999999999999999999999999999999999);
ERROR 58 (HY000): OB-58: Decimal overflow error
```

该错误已解决

4. 针对一个 decimal 类型属性，无法插入一个负数。示例如下：

drop table if exists t1;
create table t1(c1 int primary key, c2 decimal(5,1));
insert into t1 values (1, -1111.1);

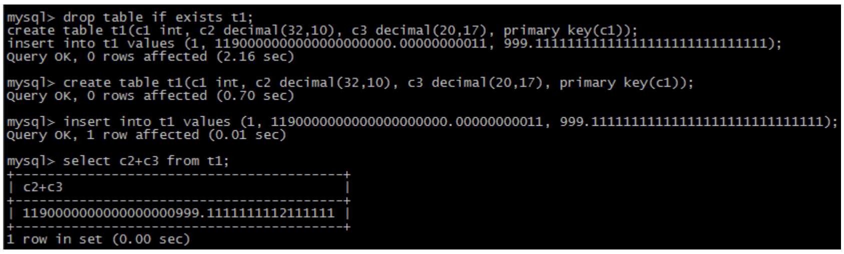


```
mysql> create table t3 (c1 int primary key, c2 decimal(5,1));
Query OK, 0 rows affected (2.04 sec)

mysql> desc t3;
+-------+--------------+----------+-----+---------+-------+
| field | type         | nullable | key | default | extra |
+-------+--------------+----------+-----+---------+-------+
| c1    | int          |          |   1 | NULL    |       |
| c2    | decimal(5,1) |          |   0 | NULL    |       |
+-------+--------------+----------+-----+---------+-------+
2 rows in set (0.00 sec)

mysql> insert into t3 values (1, -1111.1);
ERROR 58 (HY000): OB-58: Decimal overflow error
mysql>
```

该错误已解决

5. 两个 decimal 类型属性做加法运算时，精度控制与文档描述不符。示例如下：

drop table if exists t1;
create table t1(c1 int, c2 decimal(32,10), c3 decimal(20,17), primary key(c1));
insert into t1 values (1, 1190000000000000000000.00000000011,
999.1111111111111111111111111111);
select c2+c3 from t1;



```
mysql> drop table if exists t1;
create table t1(c1 int, c2 decimal(32,10), c3 decimal(20,17), primary key(c1));
insert into t1 values (1, 1190000000000000000000.00000000011, 999.111111111111111111111111111111);
Query OK, 0 rows affected (2.16 sec)

mysql> create table t1(c1 int, c2 decimal(32,10), c3 decimal(20,17), primary key(c1));
Query OK, 0 rows affected (0.70 sec)

mysql> insert into t1 values (1, 1190000000000000000000.00000000011, 999.111111111111111111111111111111);
Query OK, 1 row affected (0.01 sec)

mysql> select c2+c3 from t1;
+-----------------------------------------+
| c2+c3                                   |
+-----------------------------------------+
| 1190000000000000000999.1111111112111111 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

	p	s
ADD	Min(38, max(p-s, p’-s’)+max(s, s’)+1)	Max(s, s’)

p=min(38,39)=38,s=17

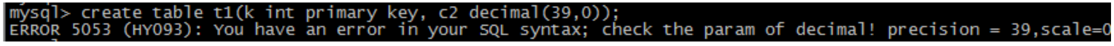
实际结果是小数位 16，整数位 22。

（文档已经解释）

6. 报错信息不规范：

drop table if exists t1;

create table t1(k int primary key, c2 decimal(39,0));



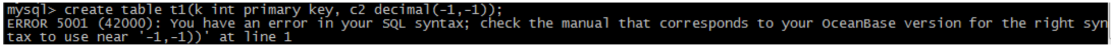
```
mysql> create table t1(k int primary key, c2 decimal(39,0));
ERROR 5053 (HY093): You have an error in your SQL syntax; check the param of decimal! precision = 39,scale=0
```

注意：首先，逗号后面应有空格，最后一个等号两边也应有空格；并且应该提示 39 大于目前 decimal 可支持的最大精度 38。

（报错不规范问题）

drop table if exists t1;

create table t1(k int primary key, c2 decimal(-1,-1));



```
mysql> create table t1(k int primary key, c2 decimal(-1,-1));
ERROR 5001 (42000): You have an error in your SQL syntax; check the manual that corresponds to your OceanBase version for the right syn
tax to use near '-1,-1))' at line 1
```

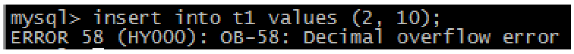
与上面的报错形式不一致。

（负数报错在不同的位置）

drop table if exists t1;

create table t1(k int primary key, c2 decimal(1,0));

insert into t1 values (2, 10);



```
mysql> insert into t1 values (2, 10);
ERROR 58 (HY000): OB-58: Decimal overflow error
```

应该报-57。-58 指十进制数不合法，而-57 指十进制数溢出

（-58 是 decimal 溢出问题）

7. 当属性定义为 decimal(p, 0)时，若插入的数字中仅含有小数部分或为 0，则会导致插入的数据有误，并且都为最大值（38 个 9）。

drop table if exists t1;

create table t1(k int primary key, c2 decimal(5,0));

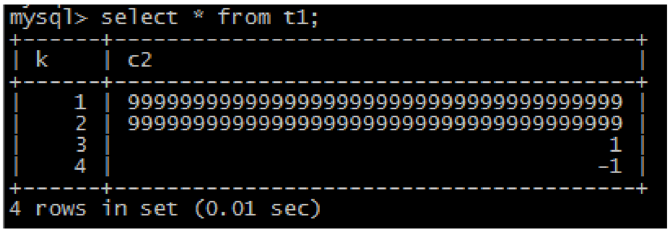
insert into t1 values (1, 0);

insert into t1 values (2, 0.1);

insert into t1 values (3, 1.1);

insert into t1 values (4, -1);

select * from t1;



```
mysql> select * from t1;
+-------+----------------------------------------+
| k     | c2                                     |
+-------+----------------------------------------+
|     1 | 99999999999999999999999999999999999999 |
|     2 | 99999999999999999999999999999999999999 |
|     3 |                                      1 |
|     4 |                                     -1 |
+-------+----------------------------------------+
4 rows in set (0.01 sec)
```

（已经修改）

8. decimal 加法运算正确性有误，示例如下：

Cedar：

drop table if exists t1;

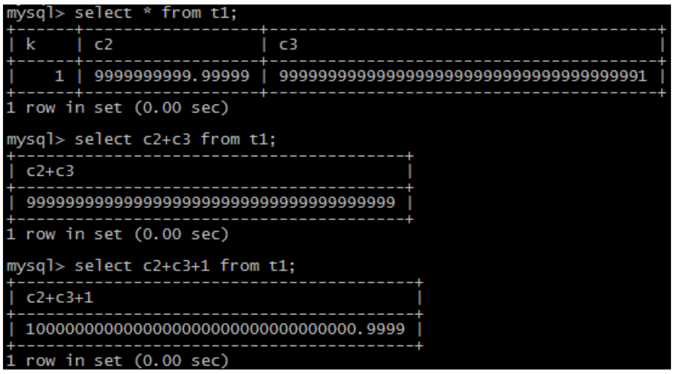
create table t1(k int primary key, c2 decimal(15,5), c3 decimal(38,0));

insert into t1 values (1, 9999999999.99999, 99999999999999999999999999999999999991.99);

select * from t1;

select c2+c3 from t1;

select c2+c3+1 from t1;



```
mysql> select * from t1;
+-------+------------------+----------------------------------------+
| k     | c2               | c3                                     |
+-------+------------------+----------------------------------------+
|     1 | 9999999999.99999 | 99999999999999999999999999999999999991 |
+-------+------------------+----------------------------------------+
1 row in set (0.00 sec)

mysql> select c2+c3 from t1;
+----------------------------------------+
| c2+c3                                  |
+----------------------------------------+
| 99999999999999999999999999999999999999 |
+----------------------------------------+
1 row in set (0.00 sec)

mysql> select c2+c3+1 from t1;
+-----------------------------------------+
| c2+c3+1                                 |
+-----------------------------------------+
| 1000000000000000000000000000000000.9999 |
+-----------------------------------------+
1 row in set (0.00 sec)
```

MySQL：

drop table if exists t1;

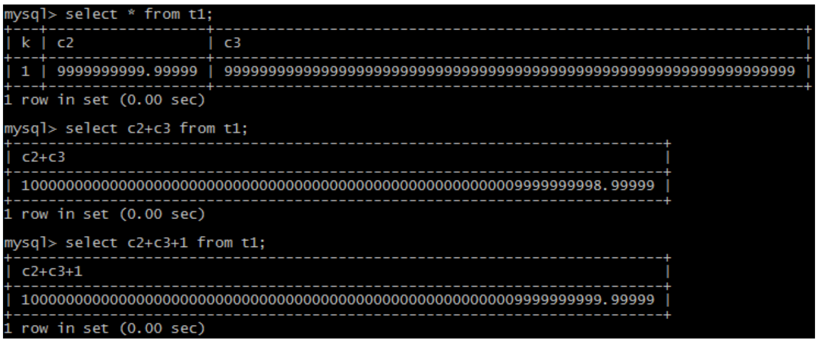
create table t1(k int primary key, c2 decimal(15,5), c3 decimal(65,0));

insert into t1 values (1, 9999999999.99999, 99999999999999999999999999999999999999999999999999999999999999999);

select * from t1;

select c2+c3 from t1;

select c2+c3+1 from t1;



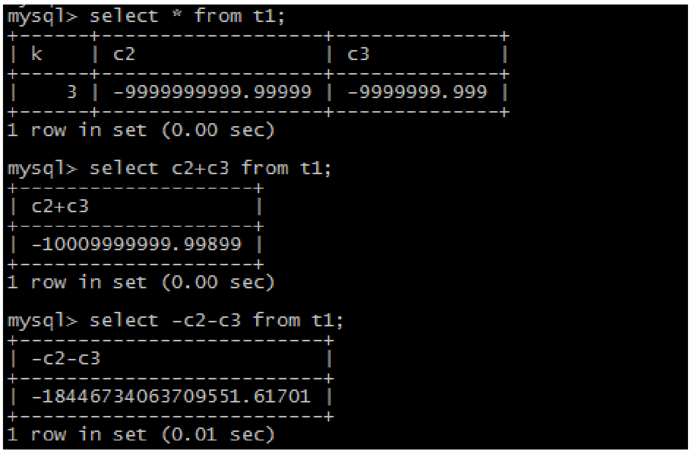
```
mysql> select * from t1;
+---+------------------+-------------------------------------------------------------------+
| k | c2               | c3                                                                |
+---+------------------+-------------------------------------------------------------------+
| 1 | 9999999999.99999 | 99999999999999999999999999999999999999999999999999999999999999999 |
+---+------------------+-------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> select c2+c3 from t1;
+-------------------------------------------------------------------------+
| c2+c3                                                                   |
+-------------------------------------------------------------------------+
| 1000000000000000000000000000000000000000000000000000000009999999998.99999 |
+-------------------------------------------------------------------------+
1 row in set (0.00 sec)

mysql> select c2+c3+1 from t1;
+-------------------------------------------------------------------------+
| c2+c3+1                                                                 |
+-------------------------------------------------------------------------+
| 1000000000000000000000000000000000000000000000000000000009999999999.99999 |
+-------------------------------------------------------------------------+
1 row in set (0.00 sec)
```

注意：MySQL 对于运算结果没有最大值的概念，但是 Cedar 有，并且在运算过程中如果出现最大值，计算结果就会由于不正确的截取策略而出错。建议遇此情况，给出适当提示信息。

（文档已经说明）

9. decimal 减法运算正确性有误，示例如下：

```
drop table if exists t1;
create table t1(k int primary key, c2 decimal(15,5), c3 decimal(10,3));
insert into t1 values (3, -9999999999.99999, -9999999.999);
select * from t1;
select c2+c3 from t1;
select -c2-c3 from t1;
```



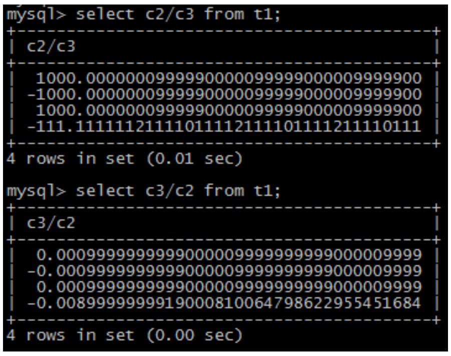
```
mysql> select * from t1;
+------+-------------------+--------------+
| k    | c2                | c3           |
+------+-------------------+--------------+
|    3 | -9999999999.99999 | -9999999.999 |
+------+-------------------+--------------+
1 row in set (0.00 sec)

mysql> select c2+c3 from t1;
+--------------------+
| c2+c3              |
+--------------------+
| -10009999999.99899 |
+--------------------+
1 row in set (0.00 sec)

mysql> select -c2-c3 from t1;
+--------------------------+
| -c2-c3                   |
+--------------------------+
| -18446734063709551.61701 |
+--------------------------+
1 row in set (0.01 sec)
```

注意：其中加法的计算结果是正确的，减法的结算结果是错误的。

（已解决）

10. 除法的计算结果精度控制与文档中描述不符，示例如下：

```
drop table if exists t1;
create table t1(k int primary key, c2 decimal(15,5), c3 decimal(10,3));
insert into t1 values (1, 9999999999.99999, 9999999.999);
insert into t1 values (2, -9999999999.99999, 9999999.999);
insert into t1 values (3, -9999999999.99999, -9999999.999);
replace into t1 values (4, -1111111111.99999, 9999999.999);
select c2/c3 from t1;
select c3/c2 from t1;
```



```
mysql> select c2/c3 from t1;
+--------------------------------------------+
| c2/c3                                      |
+--------------------------------------------+
|  1000.000000099999900000099999900000099999900 |
| -1000.000000099999900000099999900000099999900 |
|  1000.000000099999900000099999900000099999900 |
| -111.1111112111110111112111110111112111110111 |
+--------------------------------------------+
4 rows in set (0.01 sec)

mysql> select c3/c2 from t1;
+--------------------------------------------+
| c3/c2                                      |
+--------------------------------------------+
|  0.000999999999999000000999999999999000009999 |
| -0.000999999999999000000999999999999000009999 |
|  0.000999999999999000000999999999999000009999 |
| -0.00899999999999919000810064798622955451684 |
+--------------------------------------------+
4 rows in set (0.00 sec)
```

DIV 精度控制规则：

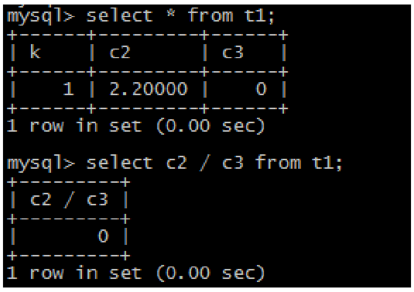
DIV	38	38-p+s-s’

根据规则：38-p+s-s’=38-15+5-3=25。可是上述计算结果小数位数明显不是 25 位。

（精度控制问题,文档说明）

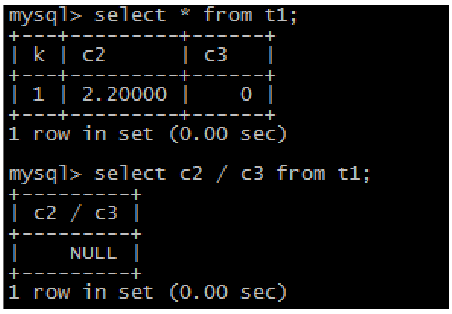
11. 一个 decimal 类型数据与一个 bool 类型数据作除法运算时，如果被除数类型为 bool 且值为 false，正确结果应为 Null，但实际结果为 0。示例如下：

drop table if exists t1;
create table t1(k int primary key, c2 decimal(15,5), c3 bool);
insert into t1 values (1, 2.2, False);
select c2 / c3 from t1;



```
mysql> select * from t1;
+------+---------+------+
| k    | c2      | c3   |
+------+---------+------+
|    1 | 2.20000 |    0 |
+------+---------+------+
1 row in set (0.00 sec)

mysql> select c2 / c3 from t1;
+---------+
| c2 / c3 |
+---------+
|       0 |
+---------+
1 row in set (0.00 sec)
```

相应的 MySQL 结果为：



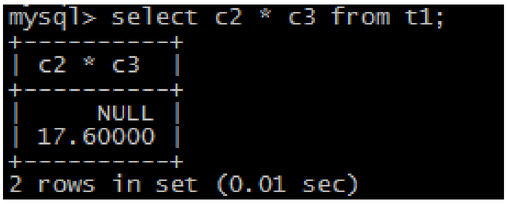
```
mysql> select * from t1;
+---+---------+------+
| k | c2      | c3   |
+---+---------+------+
| 1 | 2.20000 |    0 |
+---+---------+------+
1 row in set (0.00 sec)

mysql> select c2 / c3 from t1;
+---------+
| c2 / c3 |
+---------+
|    NULL |
+---------+
1 row in set (0.00 sec)
```

（NULL 和 0 的冲突）

12. 当 decimal 与一个字符串进行运算时，会先将字符串转化成数字类型再进行运算。在 Cedar 中，若转化不成功，则结果为 Null，但在 MySQL 中，则使用

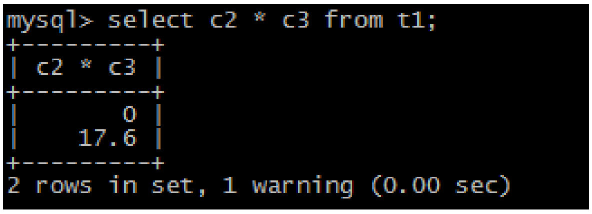
0 代替该字符串继续进行运算。Cedar 中其他数据类型相应计算规则与 MySQL 一致。

drop table if exists t1;
create table t1(k int primary key, c2 decimal(15,5), c3 varchar(100));
insert into t1 values (1, 2.2, 'zsdsa8');
insert into t1 values (2, 2.2, '8');
select c2 * c3 from t1;



```
mysql> select c2 * c3 from t1;
+-----------+
| c2 * c3   |
+-----------+
|      NULL |
| 17.60000 |
+-----------+
2 rows in set (0.01 sec)
```

相应的 MySQL 结果为：

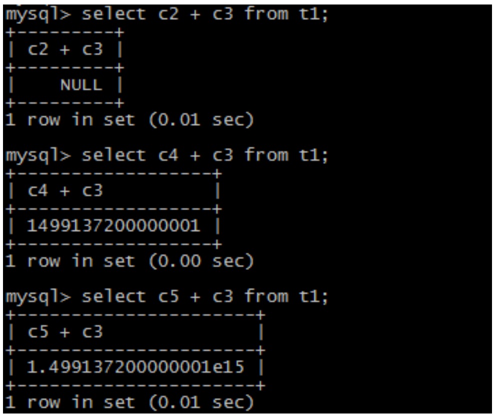


```
mysql> select c2 * c3 from t1;
+---------+
| c2 * c3 |
+---------+
|       0 |
|    17.6 |
+---------+
2 rows in set, 1 warning (0.00 sec)
```

（NULL 和 0 的冲突）

13. 在 Cedar 中，日期类型数据参与计算时，会先将其转化成相应的数字类型再进行运算。但是 decimal 未作相应的处理。示例如下：

drop table if exists t1;
create table t1(k int primary key, c2 decimal(15,5), c3 datetime, c4 int, c5 double);
insert into t1 values (1, 2.2, '2017-07-04 11:00:00', 1, 1);
select c2 + c3 from t1;
select c4 + c3 from t1;
select c5 + c3 from t1;

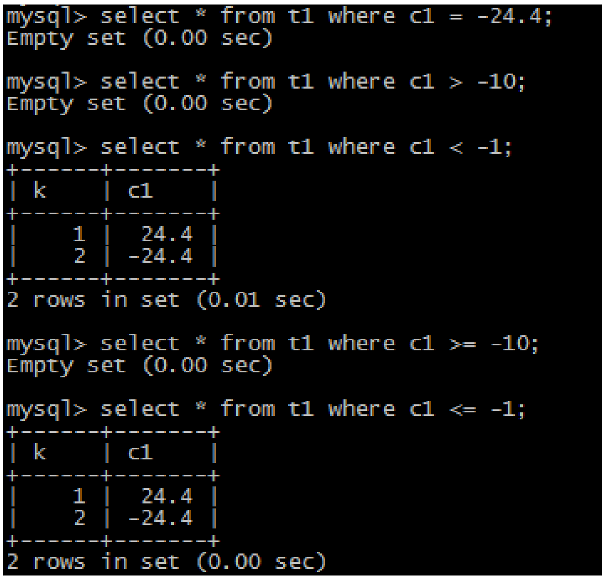


```
mysql> select c2 + c3 from t1;
+---------+
| c2 + c3 |
+---------+
|    NULL |
+---------+
1 row in set (0.01 sec)

mysql> select c4 + c3 from t1;
+-----------------+
| c4 + c3         |
+-----------------+
| 1499137200000001 |
+-----------------+
1 row in set (0.00 sec)

mysql> select c5 + c3 from t1;
+---------------------+
| c5 + c3             |
+---------------------+
| 1.499137200000001e15 |
+---------------------+
1 row in set (0.01 sec)
```

（已解决）

14. decimal 针对比较运算符=、>=、>、<=、<等，当右边比较项为负数时，运算结果会出错。示例如下：

drop table if exists t1;
create table t1(k int primary key, c1 decimal(3,1));
insert into t1 values (1, 24.4);
insert into t1 values (2, -24.4);
select * from t1 where c1 = -24.4;
select * from t1 where c1 > -10;
select * from t1 where c1 < -1;
select * from t1 where c1 >= -10;
select * from t1 where c1 <= -1;

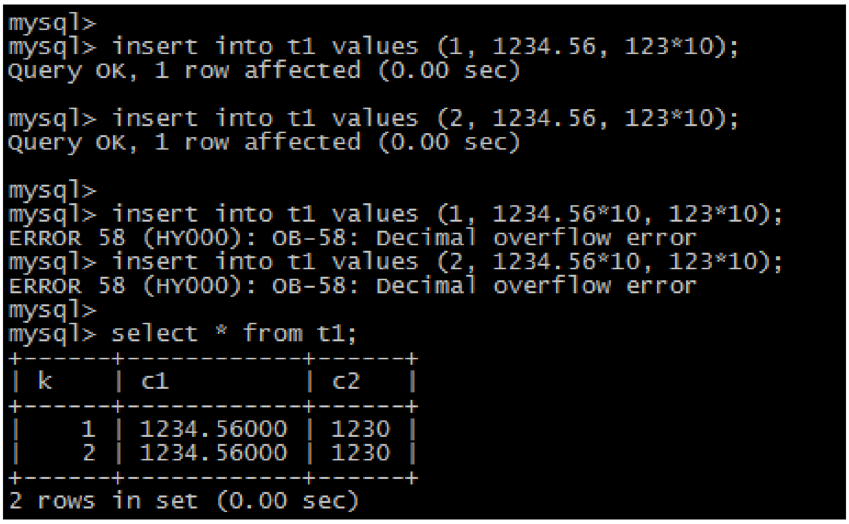


```
mysql> select * from t1 where c1 = -24.4;
Empty set (0.00 sec)

mysql> select * from t1 where c1 > -10;
Empty set (0.00 sec)

mysql> select * from t1 where c1 < -1;
+------+-------+
| k    | c1    |
+------+-------+
|    1 |  24.4 |
|    2 | -24.4 |
+------+-------+
2 rows in set (0.01 sec)

mysql> select * from t1 where c1 >= -10;
Empty set (0.00 sec)

mysql> select * from t1 where c1 <= -1;
+------+-------+
| k    | c1    |
+------+-------+
|    1 |  24.4 |
|    2 | -24.4 |
+------+-------+
2 rows in set (0.00 sec)
```

已经解决（int->decimal 类型转换函数）

15. 在 insert、replace 等语句中，decimal 不支持乘法计算（+、-、/ 是正常的），其他数据类型可以正常计算。

drop table if exists t1;
create table t1(k int primary key, c1 decimal(10,5),c2 int);
insert into t1 values (1, 1234.56, 123*10);
insert into t1 values (2, 1234.56, 123*10);
insert into t1 values (3, 1234.56*10, 123*10);
insert into t1 values (4, 1234.56*10, 123*10);
select * from t1;



```
mysql>
mysql> insert into t1 values (1, 1234.56, 123*10);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (2, 1234.56, 123*10);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> insert into t1 values (1, 1234.56*10, 123*10);
ERROR 58 (HY000): OB-58: Decimal overflow error
mysql> insert into t1 values (2, 1234.56*10, 123*10);
ERROR 58 (HY000): OB-58: Decimal overflow error
mysql>
mysql> select * from t1;
+------+------------+------+
| k    | c1         | c2   |
+------+------------+------+
|    1 | 1234.56000 | 1230 |
|    2 | 1234.56000 | 1230 |
+------+------------+------+
2 rows in set (0.00 sec)
```

(已经解决，修改 ob_expr_obj.cpp 中的 mul 函数，添加 varchar 空间)

16. 针对系统函数 cast()，强制转化 double，varchar（含字母），datatime 类型数据成 decimal 时会出错。

drop table if exists t1;

create table t1(k int primary key, c1 decimal(10,5), c2 int, c3 double, c4 bool, c5 varchar(100), c6 datetime);

insert into t1 values (1, 61234.56728, 12, 12.77, True, '258.9', '2017-07-04 10:00:00');

insert into t1 values (2, 61234.56728, 12, 12.77, False, '258a', '2017-07-04 10:00:00');

select cast(c2 as decimal) from t1;

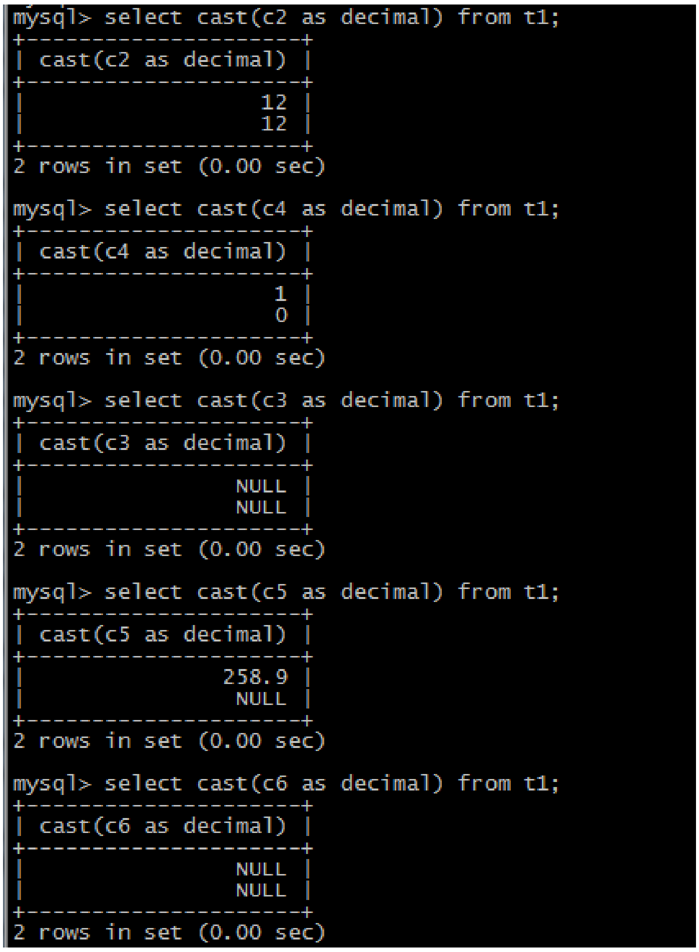
select cast(c4 as decimal) from t1;

select cast(c3 as decimal) from t1;

select cast(c5 as decimal) from t1;

select cast(c6 as decimal) from t1;

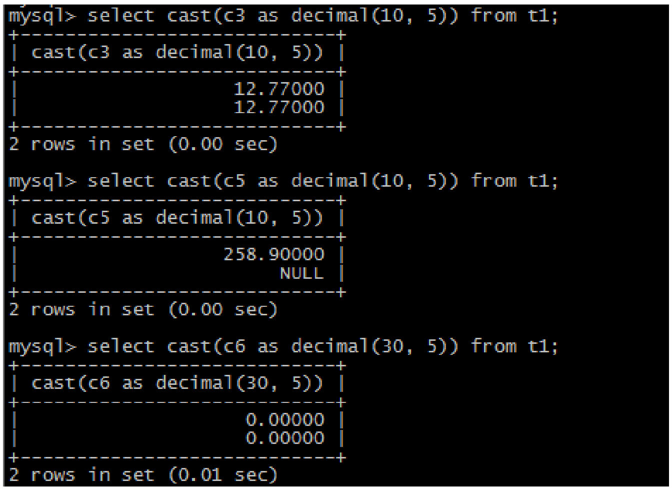
（暂时不支持这个写法）但是没有报语法错误！



```
mysql> select cast(c2 as decimal) from t1;
+---------------------+
| cast(c2 as decimal) |
+---------------------+
|                  12 |
|                  12 |
+---------------------+
2 rows in set (0.00 sec)

mysql> select cast(c4 as decimal) from t1;
+---------------------+
| cast(c4 as decimal) |
+---------------------+
|                   1 |
|                   0 |
+---------------------+
2 rows in set (0.00 sec)

mysql> select cast(c3 as decimal) from t1;
+---------------------+
| cast(c3 as decimal) |
+---------------------+
|                NULL |
|                NULL |
+---------------------+
2 rows in set (0.00 sec)

mysql> select cast(c5 as decimal) from t1;
+---------------------+
| cast(c5 as decimal) |
+---------------------+
|               258.9 |
|                NULL |
+---------------------+
2 rows in set (0.00 sec)

mysql> select cast(c6 as decimal) from t1;
+---------------------+
| cast(c6 as decimal) |
+---------------------+
|                NULL |
|                NULL |
+---------------------+
2 rows in set (0.00 sec)
```

注意：int、bool 类型数据的处理正确，double，varchar（含字母），datatime 类型数据处理有误，上述是在不指定精度情况下测试的，**需在使用文档中说明默认精度的选择（若不支持该种写法，应该报语法错误）。**

select cast(c3 as decimal(10, 5)) from t1;
select cast(c5 as decimal(10, 5)) from t1;
select cast(c6 as decimal(30, 5)) from t1;



```
mysql> select cast(c3 as decimal(10, 5)) from t1;
+----------------------------+
| cast(c3 as decimal(10, 5)) |
+----------------------------+
|                   12.77000 |
|                   12.77000 |
+----------------------------+
2 rows in set (0.00 sec)

mysql> select cast(c5 as decimal(10, 5)) from t1;
+----------------------------+
| cast(c5 as decimal(10, 5)) |
+----------------------------+
|                  258.90000 |
|                       NULL |
+----------------------------+
2 rows in set (0.00 sec)

mysql> select cast(c6 as decimal(30, 5)) from t1;
+----------------------------+
| cast(c6 as decimal(30, 5)) |
+----------------------------+
|                    0.00000 |
|                    0.00000 |
+----------------------------+
2 rows in set (0.01 sec)
```

在指定精度的情况下，double 类型数据处理正确。varchar（含字母）和 datatime

类型数据处理同样有误。

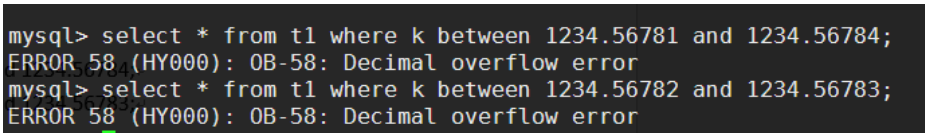
（varchar 类型的字母时无法转换成 decimal 类型的，其他问题已经解决，问题在与需要设置 len）

17. 当 decimal 作为主键时，在 select 过滤条件中通过 between…and…过滤主键时会出错。

drop table if exists t1;
create table t1(k decimal(10,5) primary key, c1 decimal(10,5));

insert into t1 values (1234.56781, 1234.4321);
insert into t1 values (1234.56782, 1234.4322);
insert into t1 values (1234.56783, 1234.4323);
insert into t1 values (1234.56784, 1234.4324);

select * from t1 where k between 1234.56781 and 1234.56784;
select * from t1 where k between 1234.56782 and 1234.56783;

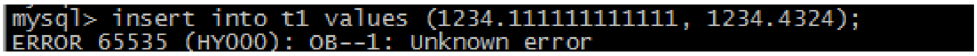


```
mysql> select * from t1 where k between 1234.56781 and 1234.56784;
ERROR 58 (HY000): OB-58: Decimal overflow error
mysql> select * from t1 where k between 1234.56782 and 1234.56783;
ERROR 58 (HY000): OB-58: Decimal overflow error
```

（ob_sql_read_strategy.cpp:522）（**已经解决**）

18. <u>当 decimal 作为主键时，如插入的数据小数部分溢出，按照使用手册中的溢出机制应该直接截取，但是直接执行时报 OB--1: Unknown error。示例如下：</u>

drop table if exists t1;
create table t1(k decimal(10,5) primary key, c1 decimal(10,5));
insert into t1 values (1234.111111111111, 1234.4324);



```
mysql> insert into t1 values (1234.111111111111, 1234.4324);
ERROR 65535 (HY000): OB--1: Unknown error
```

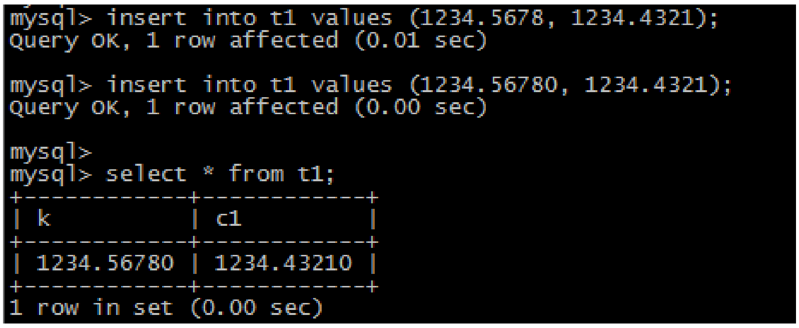
（可能是主键有效数字太多，但是代码没有进行截取，问题在于表达式的构建，主键要构建 IN 表达式在 rpc_scan 中，错误更新在 fix_varchar_and_decimal 函数中，缺少对 len 的处理。已经解决）

19. 当 decimal 作为主键时，两次插入的数据实际值相同，但是未报主键冲突错误。示例如下：

drop table if exists t1;
create table t1(k decimal(10,5) primary key, c1 decimal(10,5));
insert into t1 values (1234.5678, 1234.4321);
insert into t1 values (1234.56780, 1234.4321);

select * from t1;

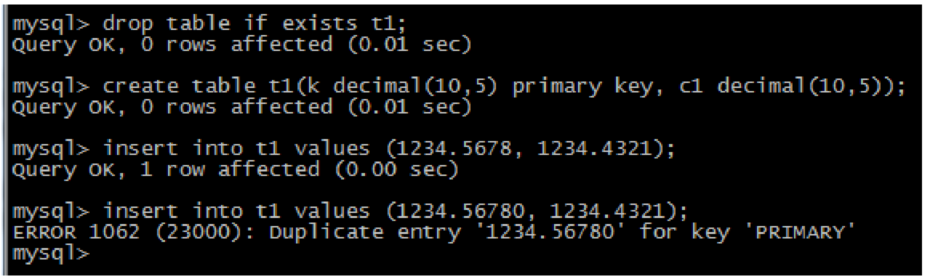
(已经解决，多了一个 0 与主键插入类似问题，对于多出来的 0 如何处理，已解)



```
mysql> insert into t1 values (1234.5678, 1234.4321);
Query OK, 1 row affected (0.01 sec)

mysql> insert into t1 values (1234.56780, 1234.4321);
Query OK, 1 row affected (0.00 sec)

mysql>
mysql> select * from t1;
+------------+------------+
| k          | c1         |
+------------+------------+
| 1234.56780 | 1234.43210 |
+------------+------------+
1 row in set (0.00 sec)
```

注意：实际都执行成功了，但是最终只插入了一条记录，第二次插入应该报主键冲突

相应 MySQL 的执行结果：



```
mysql> drop table if exists t1;
Query OK, 0 rows affected (0.01 sec)

mysql> create table t1(k decimal(10,5) primary key, c1 decimal(10,5));
Query OK, 0 rows affected (0.01 sec)

mysql> insert into t1 values (1234.5678, 1234.4321);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (1234.56780, 1234.4321);
ERROR 1062 (23000): Duplicate entry '1234.56780' for key 'PRIMARY'
mysql>
```

20. 当 decimal 作为主键时，在正常的更新操作（update）下，会出现**数据丢失问题**。示例如下：

drop table if exists t1;

create table t1(k decimal(10,5) primary key, c1 decimal(10,5));

insert into t1 values (1234.56789, 1234.4321);

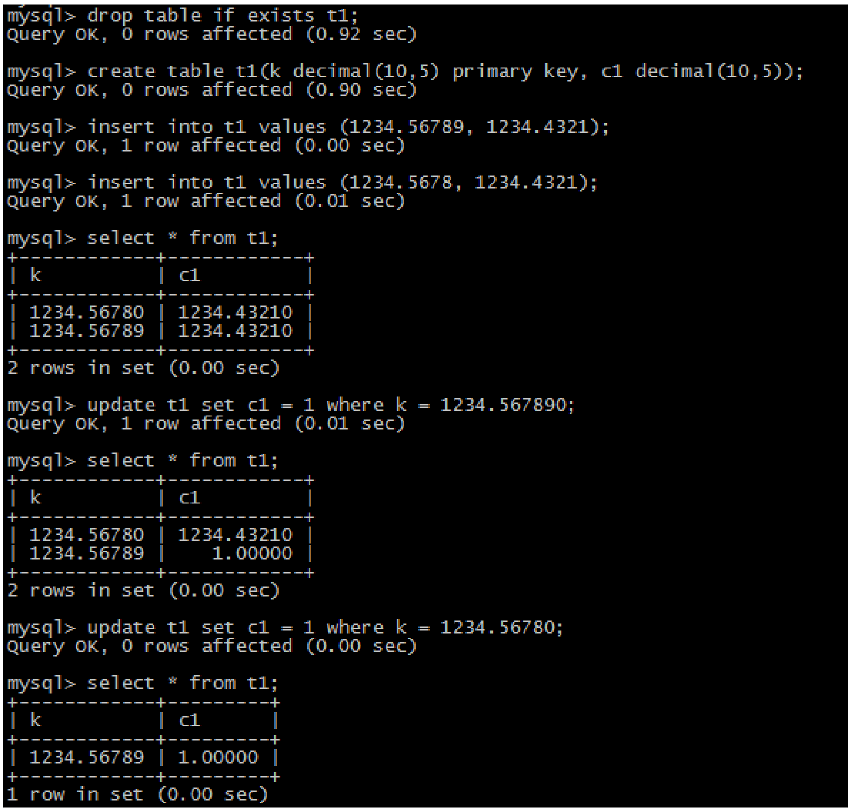
insert into t1 values (1234.5678, 1234.4321);

update t1 set c1 = 1 where k = 1234.567890;

update t1 set c1 = 1 where k = 1234.56780;

select * from t1;

(已经解决，多了一个 0 与主键插入类似问题，对于多出来的 0 如何处理，已解)

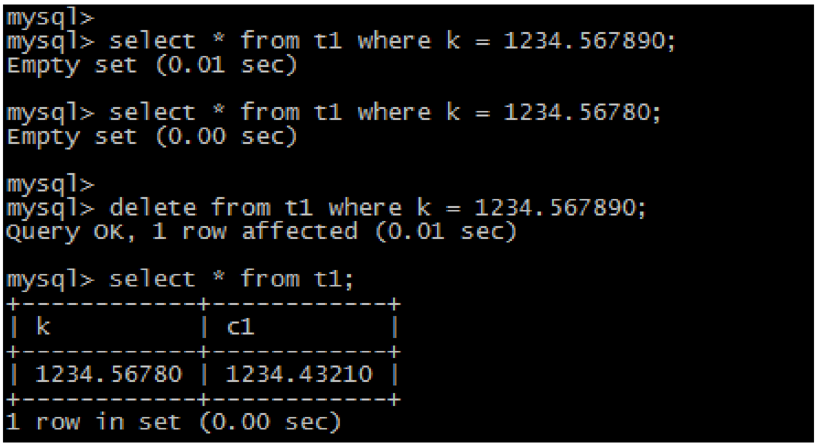


```
mysql> drop table if exists t1;
Query OK, 0 rows affected (0.92 sec)

mysql> create table t1(k decimal(10,5) primary key, c1 decimal(10,5));
Query OK, 0 rows affected (0.90 sec)

mysql> insert into t1 values (1234.56789, 1234.4321);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (1234.5678, 1234.4321);
Query OK, 1 row affected (0.01 sec)

mysql> select * from t1;
+------------+------------+
| k          | c1         |
+------------+------------+
| 1234.56780 | 1234.43210 |
| 1234.56789 | 1234.43210 |
+------------+------------+
2 rows in set (0.00 sec)

mysql> update t1 set c1 = 1 where k = 1234.567890;
Query OK, 1 row affected (0.01 sec)

mysql> select * from t1;
+------------+------------+
| k          | c1         |
+------------+------------+
| 1234.56780 | 1234.43210 |
| 1234.56789 |    1.00000 |
+------------+------------+
2 rows in set (0.00 sec)

mysql> update t1 set c1 = 1 where k = 1234.56780;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from t1;
+------------+---------+
| k          | c1      |
+------------+---------+
| 1234.56789 | 1.00000 |
+------------+---------+
1 row in set (0.00 sec)
```

21. 当 decimal 作为主键时，此时在这个 decimal 类型主键上做过滤，select 返回结果会出错。示例如下：

```
drop table if exists t1;
create table t1(k decimal(10,5) primary key, c1 decimal(10,5));
insert into t1 values (1234.56789, 1234.43212);
insert into t1 values (1234.5678, 1234.4321);
select * from t1 where k = 1234.567890;
select * from t1 where k = 1234.56780;
```

(多了一个 0 与主键插入类似问题，对于多出来的 0 如何处理,已解)

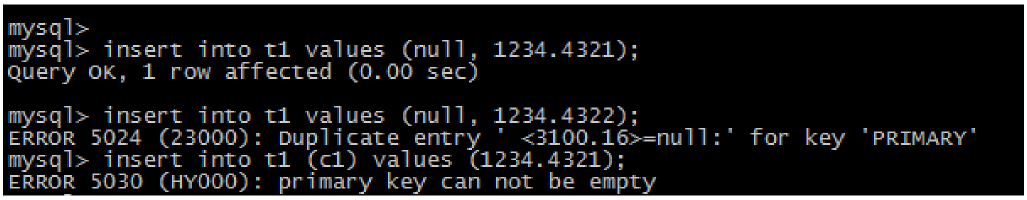


```
mysql>
mysql> select * from t1 where k = 1234.567890;
Empty set (0.01 sec)

mysql> select * from t1 where k = 1234.56780;
Empty set (0.00 sec)

mysql>
mysql> delete from t1 where k = 1234.567890;
Query OK, 1 row affected (0.01 sec)

mysql> select * from t1;
+------------+------------+
| k          | c1         |
+------------+------------+
| 1234.56780 | 1234.43210 |
+------------+------------+
1 row in set (0.00 sec)
```

注意：select 会出错，但是 delete 正常。
（已解决）

22. 在 decimal 作为主键时，主键自增功能有误。示例：

drop table if exists t1;
create table t1(k decimal(10,5) auto_increment primary key, c1 decimal(10,5));
insert into t1 values (null, 1234.4321);
insert into t1 values (null, 1234.4322);
insert into t1 (c1) values (1234.4321);



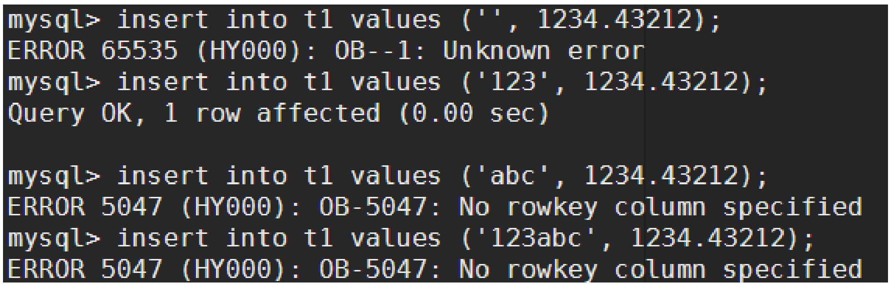
```
mysql>
mysql> insert into t1 values (null, 1234.4321);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values (null, 1234.4322);
ERROR 5024 (23000): Duplicate entry ' <3100.16>=null:' for key 'PRIMARY'
mysql> insert into t1 (c1) values (1234.4321);
ERROR 5030 (HY000): primary key can not be empty
```

（已解决，但是插 NULL 值仍存在）

23. Decimal 作为主键时插入 varchar 类型时报错信息不规范：

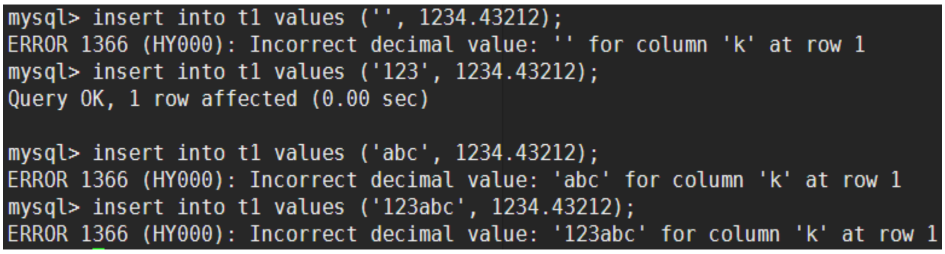
drop table if exists t1;
create table t1(k decimal(10,5) primary key, c1 decimal(10,5));
insert into t1 values ('', 1234.43212);

CEDAR 中报错信息：



```
mysql> insert into t1 values ('', 1234.43212);
ERROR 65535 (HY000): OB--1: Unknown error
mysql> insert into t1 values ('123', 1234.43212);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values ('abc', 1234.43212);
ERROR 5047 (HY000): OB-5047: No rowkey column specified
mysql> insert into t1 values ('123abc', 1234.43212);
ERROR 5047 (HY000): OB-5047: No rowkey column specified
```

对应的 MySQL 中报错信息：

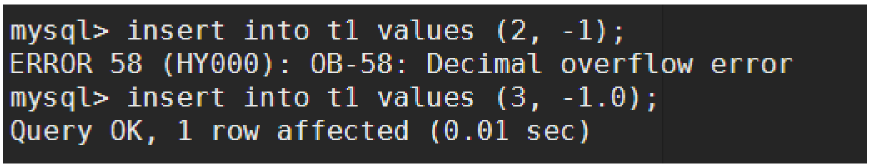


```
mysql> insert into t1 values ('', 1234.43212);
ERROR 1366 (HY000): Incorrect decimal value: '' for column 'k' at row 1
mysql> insert into t1 values ('123', 1234.43212);
Query OK, 1 row affected (0.00 sec)

mysql> insert into t1 values ('abc', 1234.43212);
ERROR 1366 (HY000): Incorrect decimal value: 'abc' for column 'k' at row 1
mysql> insert into t1 values ('123abc', 1234.43212);
ERROR 1366 (HY000): Incorrect decimal value: '123abc' for column 'k' at row 1
```

（已解决）

24. 当属性定义为 decimal(p, 0)时，若插入的数字中有负整数时，会导致插入的数据有误，报数据溢出错误。示例如下：

```
drop table if exists t1;
create table t1(k int primary key, c2 decimal(5,0));
insert into t1 values (1, 1);
insert into t1 values (2, -1);
insert into t1 values (3, -1.0);
```

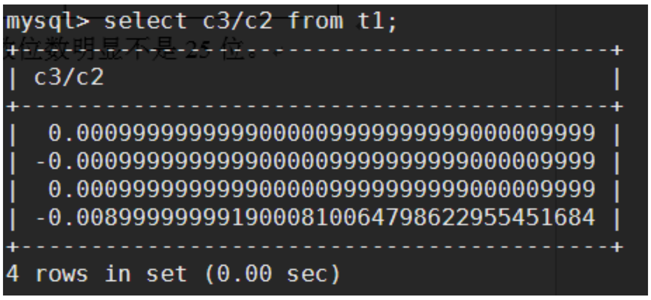


```
mysql> insert into t1 values (2, -1);
ERROR 58 (HY000): OB-58: Decimal overflow error
mysql> insert into t1 values (3, -1.0);
Query OK, 1 row affected (0.01 sec)
```

（已解决）

25. 除法的计算结果当整数部分为 0 时精度控制仍然与文档中描述不符，示例如下：

```
drop table if exists t1;
create table t1(k int primary key, c2 decimal(15,5), c3 decimal(10,3));
insert into t1 values (1, 9999999999.99999, 9999999.999);
insert into t1 values (2, -9999999999.99999, 9999999.999);
insert into t1 values (3, -9999999999.99999, -9999999.999);
replace into t1 values (4, -1111111111.99999, 9999999.999);
select c3/c2 from t1;
```



```
mysql> select c3/c2 from t1;
+-----------------------------------------+
| c3/c2                                   |
+-----------------------------------------+
|  0.0009999999999000009999999999000009999 |
| -0.0009999999999000009999999999000009999 |
|  0.0009999999999000009999999999000009999 |
| -0.0089999999991900081006479862295545168 |
+-----------------------------------------+
4 rows in set (0.00 sec)
```

DIV 精度控制规则：

int_len：运算结果整数有效位

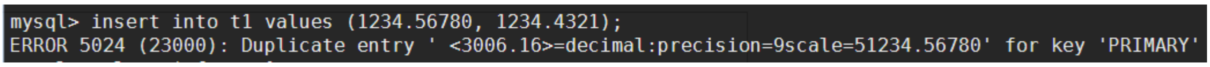
DIV	38	38-int_len

根据规则：上图中 int_len=0，所以 s=38-0=38。可是上述计算结果小数位数只有37 位。

（文档说明）

26. 输出信息不规范：输出信息中 p 和 s 信息混在一起了，中间应该加上空格。而且引号左右两边不对称。

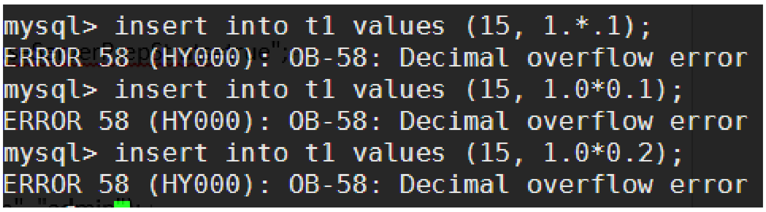
```
drop table if exists t1;
create table t1(k decimal(10,5) primary key, c1 decimal(10,5));
insert into t1 values (1234.5678, 1234.4321);
insert into t1 values (1234.56780, 1234.4321);
select * from t1;
```



```
mysql> insert into t1 values (1234.56780, 1234.4321);
ERROR 5024 (23000): Duplicate entry ' <3006.16>=decimal:precision=9scale=51234.56780' for key 'PRIMARY'
```

27. 在 decimal 测试中，插入语句中会出现以下错误：

```
drop table if exists t1;
create table t1(k int primary key, c1 decimal(10,5));
insert into t1 values (15, 1.*.1);
insert into t1 values (15, 1.0*0.1);
insert into t1 values (15, 1.0*0.2);
```



```
mysql> insert into t1 values (15, 1.*.1);
ERROR 58 (HY000): OB-58: Decimal overflow error
mysql> insert into t1 values (15, 1.0*0.1);
ERROR 58 (HY000): OB-58: Decimal overflow error
mysql> insert into t1 values (15, 1.0*0.2);
ERROR 58 (HY000): OB-58: Decimal overflow error
```

（已解决）

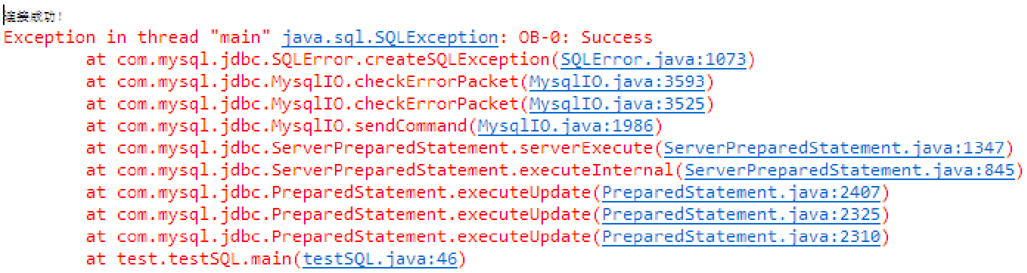
28. JDBC 接口针对 decimal 数据类型的支持性不是很好。prepareStatement 代码插入过程失败，报 OB-0: Success 错误；在查询数据库中数据时，prepareStatement 查询到的结果精度不正确！示例如下：

```
String sql=null;
 String driver = "com.mysql.jdbc.Driver";
```

```
  String url = "jdbc:mysql://10.11.1.194:14880/mysql?useServerPrepStmts=true";
  Connection conn = null;
  try {
        Class.forName(driver);
        conn = DriverManager.getConnection(url, "admin", "admin");
        conn.createStatement().executeQuery("set
@@session.ob_query_timeout=9000000000;");
  } catch (Exception e) {
  }
  if(conn!=null)
  {
        System.out.println("连接成功！");
  }
  Statement stmt=null;
  PreparedStatement pstmt=null;
  stmt = conn.createStatement();
  sql="drop table if exists t1";
  stmt.executeUpdate(sql);
  sql="create table t1(k int primary key, c1 decimal(10,5))";
  stmt.executeUpdate(sql);
  sql="insert into t1 values(1, 1111.111)";
  stmt.executeUpdate(sql);
//sql="insert into t1 values(?,?)";
//pstmt=conn.prepareStatement(sql);
//pstmt.setInt(1, 2);
//pstmt.setBigDecimal(2, new BigDecimal(123.456));
//pstmt.executeUpdate();
  ResultSet rs=stmt.executeQuery("select * from t1");
  while(rs.next())
  {
        System.out.println(rs.getInt(1)+"    "+rs.getBigDecimal(2));
  }
  PreparedStatement pstmt2=conn.prepareStatement("select * from t1");
  ResultSet rs2=pstmt2.executeQuery();
  while(rs2.next())
  {
        System.out.println(rs2.getInt(1)+"    "+rs2.getBigDecimal(2));
  }
```

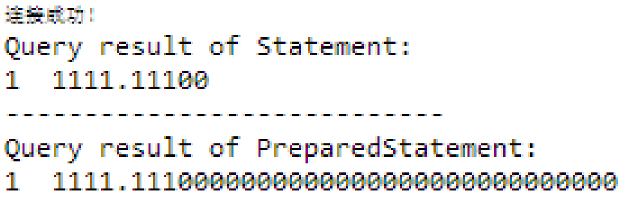
执行结果：

prepareStatement 代码插入：



```
连接成功!
Exception in thread "main" java.sql.SQLException: OB-0: Success
	at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:1073)
	at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3593)
	at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3525)
	at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:1986)
	at com.mysql.jdbc.ServerPreparedStatement.serverExecute(ServerPreparedStatement.java:1347)
	at com.mysql.jdbc.ServerPreparedStatement.executeInternal(ServerPreparedStatement.java:845)
	at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:2407)
	at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:2325)
	at com.mysql.jdbc.PreparedStatement.executeUpdate(PreparedStatement.java:2310)
	at test.testSQL.main(testSQL.java:46)
```

prepareStatement 查询结果:

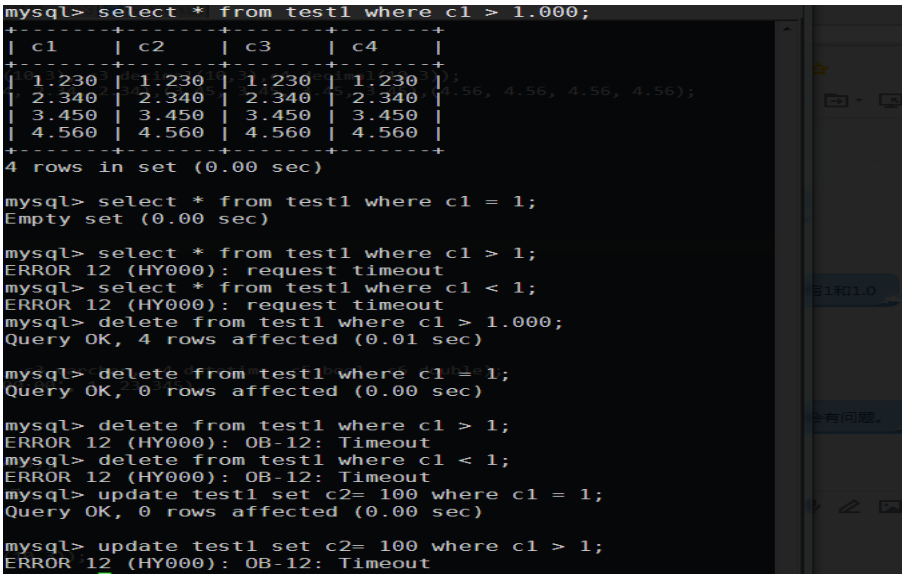


```
连接成功!
Query result of Statement:
1  1111.11100
----------------------------
Query result of PreparedStatement:
1  1111.111000000000000000000000000000
```

（已解决 PrepareStatement 插入数据问题，但是输出精度仍然不对（输出函数的问题））

29. Select 和 delete，update 语句在 decimal 做主键时，会产生以下类型转换错误：

drop table if exists test1,test2;
create table test1(c1 decimal(10,3) primary key, c2 decimal(10,3), c3 decimal(10,3),c4 decimal(10,3));
insert into test1 values(1.23, 1.23, 1.23, 1.23),(2.34, 2.34, 2.34, 2.34),(3.45, 3.45, 3.45, 3.45),(4.56, 4.56, 4.56, 4.56);
select * from test1 where c1 > 1.000;
select * from test1 where c1 = 1;
select * from test1 where c1 > 1;
select * from test1 where c1 < 1;
delete from test1 where c1 > 1.000;
delete from test1 where c1 = 1;
delete from test1 where c1 > 1;
delete from test1 where c1 < 1;
update test1 set c2= 100 where c1 = 1;
update test1 set c2= 100 where c1 > 1;
update test1 set c2= 100 where c1 < 1;
insert into test1 values (100, 1.23, 1.23, 1.23);

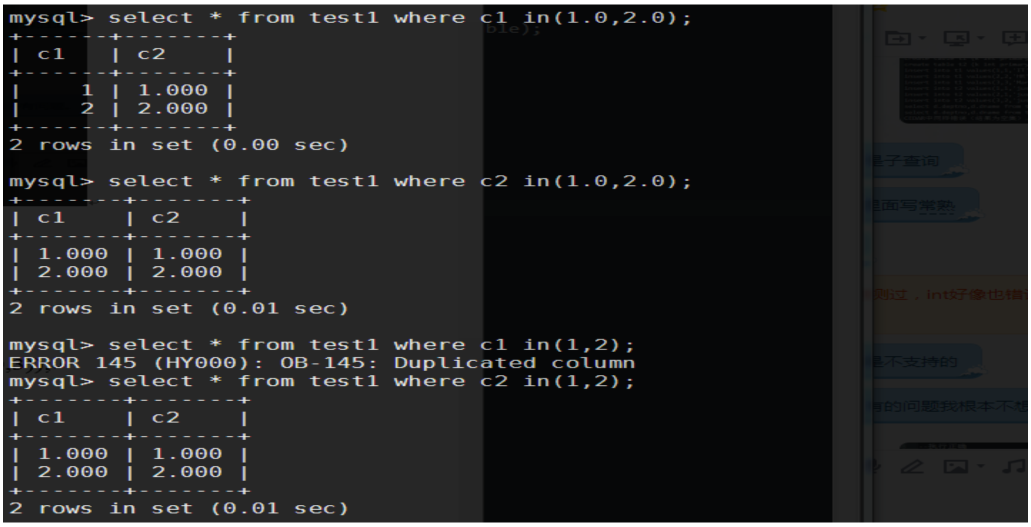


```
mysql> select * from test1 where c1 > 1.000;
+--------+--------+--------+--------+
| c1     | c2     | c3     | c4     |
+--------+--------+--------+--------+
|  1.230 |  1.230 |  1.230 |  1.230 |
|  2.340 |  2.340 |  2.340 |  2.340 |
|  3.450 |  3.450 |  3.450 |  3.450 |
|  4.560 |  4.560 |  4.560 |  4.560 |
+--------+--------+--------+--------+
4 rows in set (0.00 sec)

mysql> select * from test1 where c1 = 1;
Empty set (0.00 sec)

mysql> select * from test1 where c1 > 1;
ERROR 12 (HY000): request timeout
mysql> select * from test1 where c1 < 1;
ERROR 12 (HY000): request timeout
mysql> delete from test1 where c1 > 1.000;
Query OK, 4 rows affected (0.01 sec)

mysql> delete from test1 where c1 = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> delete from test1 where c1 > 1;
ERROR 12 (HY000): OB-12: Timeout
mysql> delete from test1 where c1 < 1;
ERROR 12 (HY000): OB-12: Timeout
mysql> update test1 set c2= 100 where c1 = 1;
Query OK, 0 rows affected (0.00 sec)

mysql> update test1 set c2= 100 where c1 > 1;
ERROR 12 (HY000): OB-12: Timeout
```

（已解决）

30. in 表达式中如果是主键，会产生如下错误：

drop table if exists test1,test2;
create table test1(c1 decimal(10,3) primary key, c2 decimal(10,3));
insert into test1 values(1,1);
insert into test1 values(2.0,2.0);
select * from test1 where c1 in(1.0,2.0);
select * from test1 where c2 in(1.0,2.0);
select * from test1 where c1 in(1,2);
select * from test1 where c2 in(1,2);



```
mysql> select * from test1 where c1 in(1.0,2.0);
+-------+--------+
| c1    | c2     |
+-------+--------+
|     1 |  1.000 |
|     2 |  2.000 |
+-------+--------+
2 rows in set (0.00 sec)

mysql> select * from test1 where c2 in(1.0,2.0);
+--------+--------+
| c1     | c2     |
+--------+--------+
|  1.000 |  1.000 |
|  2.000 |  2.000 |
+--------+--------+
2 rows in set (0.01 sec)

mysql> select * from test1 where c1 in(1,2);
ERROR 145 (HY000): OB-145: Duplicated column
mysql> select * from test1 where c2 in(1,2);
+--------+--------+
| c1     | c2     |
+--------+--------+
|  1.000 |  1.000 |
|  2.000 |  2.000 |
+--------+--------+
2 rows in set (0.01 sec)
```

（已解决）

性能测试情况：

1. 单集群测试 decimal 写入性能。

配置 194: RS+UPS+MS+CS；

测试目的：比较 decimal 数据类型和 double 数据类型的写入性能；

测试方法：分别建立以下表结构，分别向表中写入 100W 条数据记录，比较写入结果：

drop table if exists test1;

Create table test1(c1 int primary key, c2 int,c3 float,c4 double,c5 decimal(30,10),c6 decimal(30,10),c7 decimal(30,10),c8 decimal(30,10),c9 decimal(30,10),c10 decimal(30,10));

drop table if exists test2;

Create table test2(k int primary key, c1 int,c2 float,c3 double, c4 double, c5 double, c6 double, c7 double ,c8 double ,c9 double, c10 double);

然后分别向表中插入 100w，500w，1000w 条记录数据（通过开启 100 个线程）

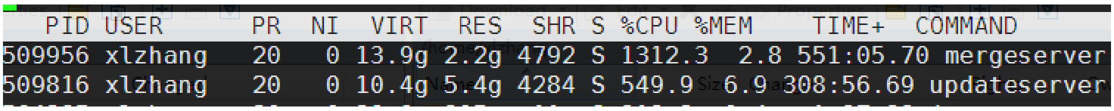
set @@session.ob_query_timeout=9000000000;

运行 jar 包

测试结果如下：

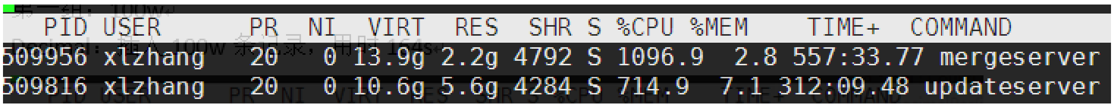
第一组：100w

Decimal：插入 100w 条记录，用时 54s（mysql 40s）



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
509956 xlzhang   20   0 13.9g 2.2g 4792 S 1312.3  2.8 551:05.70 mergeserver
509816 xlzhang   20   0 10.4g 5.4g 4284 S 549.9  6.9 308:56.69 updateserver
```

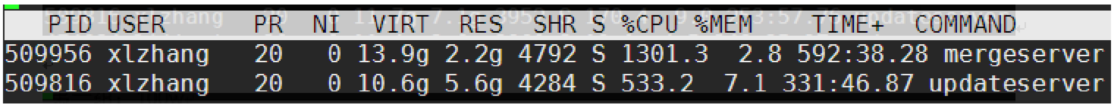
Double：插入 100w 条记录，用时 45s（mysql 36s）



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
509956 xlzhang   20   0 13.9g 2.2g 4792 S 1096.9  2.8 557:33.77 mergeserver
509816 xlzhang   20   0 10.6g 5.6g 4284 S 714.9  7.1 312:09.48 updateserver
```

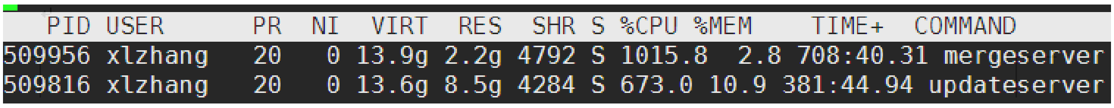
第二组：500w

Decimal：插入 500w 条记录，用时 270s



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
509956 xlzhang   20   0 13.9g 2.2g 4792 S 1301.3  2.8 592:38.28 mergeserver
509816 xlzhang   20   0 10.6g 5.6g 4284 S 533.2  7.1 331:46.87 updateserver
```

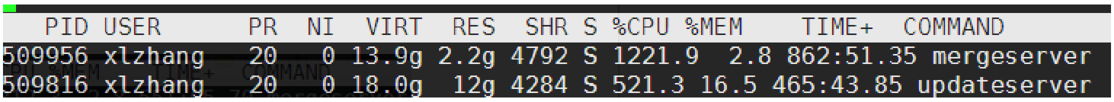
Double：插入 500w 条记录，用时 239s



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
509956 xlzhang   20   0 13.9g 2.2g 4792 S 1015.8  2.8 708:40.31 mergeserver
509816 xlzhang   20   0 13.6g 8.5g 4284 S 673.0 10.9 381:44.94 updateserver
```

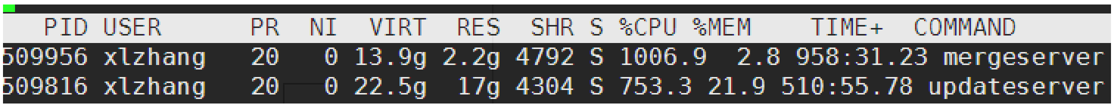
第三组：1000w

Decimal：插入 1000w 条记录，用时 541s



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
509956 xlzhang   20   0 13.9g 2.2g 4792 S 1221.9  2.8 862:51.35 mergeserver
509816 xlzhang   20   0 18.0g  12g 4284 S 521.3 16.5 465:43.85 updateserver
```

Double：插入 1000w 条记录，用时 403s

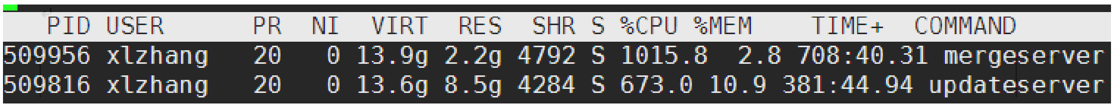


```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
509956 xlzhang   20   0 13.9g 2.2g 4792 S 1006.9  2.8 958:31.23 mergeserver
509816 xlzhang   20   0 22.5g  17g 4304 S 753.3 21.9 510:55.78 updateserver
```

结论：decimal 数据类型的插入比 double 类型的数据插入速度要慢，但是在 decimal 插入过程中，MS 所占 CPU 利用率高于 double 类型，UPS 所占 CPU 利用率低于 double 类型。

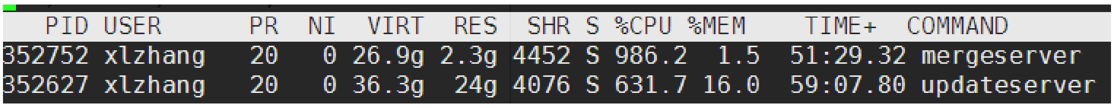
2. 测试 decimal 数据类型的引入对 double 数据类型插入速度的影响
配置：194：RS+UPS+MS+CS（含 decimal 的版本）
　　　195：RS+UPS+MS+CS（不含 decimal 的版本）
测试目的：测试在引入 decimal 数据类型后对 double 数据类型插入速度的影响
测试方法：分别在两个版本上建立如下数据表结构，向其中插入 500w 条记录（100 线程）
drop table if exists test2;
Create table test2(k int primary key, c1 int,c2 float,c3 double, c4 double, c5 double, c6 double, c7 double ,c8 double ,c9 double, c10 double);
最后比较测试结果。

测试结果如下：
Double（含 decimal 版本，分支 Decimal_optimize）：插入 500w 记录，用时 239s



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
509956 xlzhang   20   0 13.9g 2.2g 4792 S 1015.8  2.8 708:40.31 mergeserver
509816 xlzhang   20   0 13.6g 8.5g 4284 S 673.0 10.9 381:44.94 updateserver
```

Double（不含 decimal 版本，分支 Dev）：插入 500w 记录，用时 217s



```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
352752 xlzhang   20   0 26.9g 2.3g 4452 S 986.2  1.5  51:29.32 mergeserver
352627 xlzhang   20   0 36.3g  24g 4076 S 631.7 16.0  59:07.80 updateserver
```

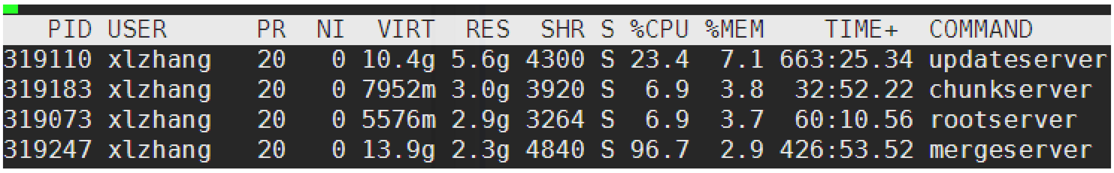
3. 测试 decimal 的引入有无导致内存泄漏
配置：194：RS+UPS+MS+CS
测试目的：测试 decimal 的引入会不会造成内存的泄漏

测试方法：通过在 decimal 的计算查询过程中监控系统的资源情况来判断。
drop table if exists test1;
Create table test1(c1 int primary key, c2 int,c3 float,c4 double,c5 decimal(30,10));

计算表达式：(((c2+c5)*(c3-c5)/c2-c5)*c5+c4)/c5

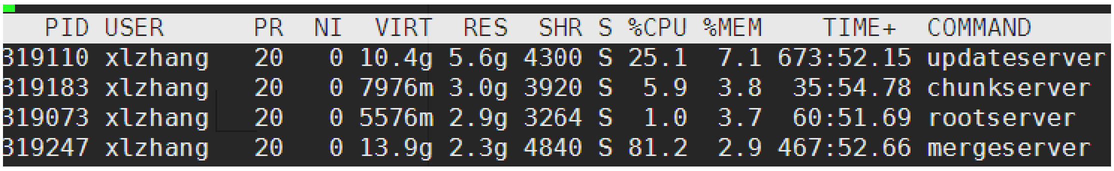
向表中插入 100w 行数据，然后循环执行以下操作：
set @@session.ob_query_timeout=9000000000;
select (((c2+c5)*(c3-c5)/c2-c5)*c5+c4)/c5 from test1;

执行 1 次时内存资源占用情况：



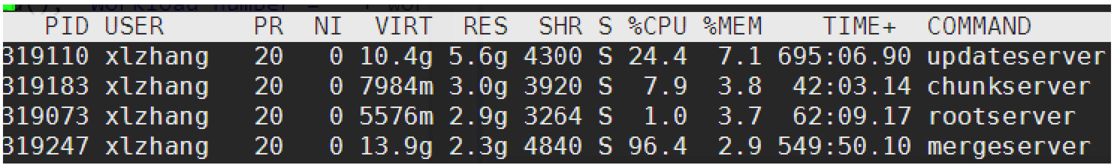
```
   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
319110 xlzhang   20   0 10.4g 5.6g 4300 S 23.4  7.1 663:25.34 updateserver
319183 xlzhang   20   0 7952m 3.0g 3920 S  6.9  3.8  32:52.22 chunkserver
319073 xlzhang   20   0 5576m 2.9g 3264 S  6.9  3.7  60:10.56 rootserver
319247 xlzhang   20   0 13.9g 2.3g 4840 S 96.7  2.9 426:53.52 mergeserver
```

执行 100 次时内存资源占用情况：



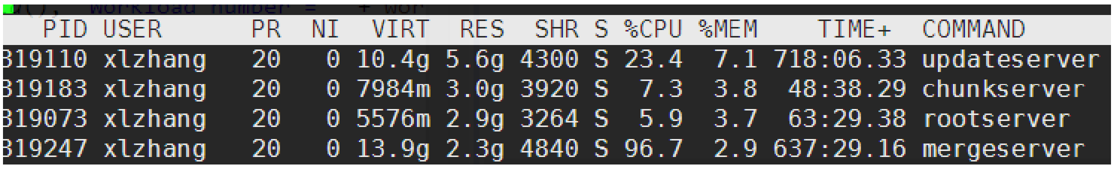
```
   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
319110 xlzhang   20   0 10.4g 5.6g 4300 S 25.1  7.1 673:52.15 updateserver
319183 xlzhang   20   0 7976m 3.0g 3920 S  5.9  3.8  35:54.78 chunkserver
319073 xlzhang   20   0 5576m 2.9g 3264 S  1.0  3.7  60:51.69 rootserver
319247 xlzhang   20   0 13.9g 2.3g 4840 S 81.2  2.9 467:52.66 mergeserver
```

执行 300 次时内存资源占用情况：



```
   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
319110 xlzhang   20   0 10.4g 5.6g 4300 S 24.4  7.1 695:06.90 updateserver
319183 xlzhang   20   0 7984m 3.0g 3920 S  7.9  3.8  42:03.14 chunkserver
319073 xlzhang   20   0 5576m 2.9g 3264 S  1.0  3.7  62:09.17 rootserver
319247 xlzhang   20   0 13.9g 2.3g 4840 S 96.4  2.9 549:50.10 mergeserver
```

执行 500 次时内存资源占用情况：

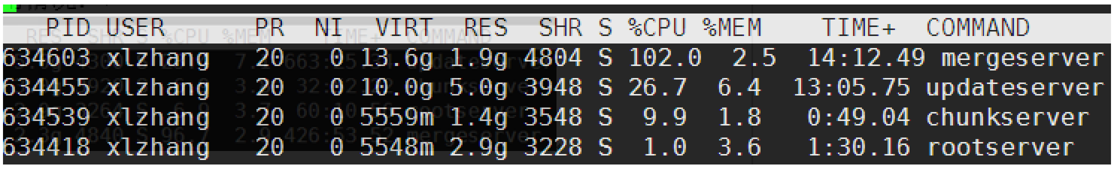


```
   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
319110 xlzhang   20   0 10.4g 5.6g 4300 S 23.4  7.1 718:06.33 updateserver
319183 xlzhang   20   0 7984m 3.0g 3920 S  7.3  3.8  48:38.29 chunkserver
319073 xlzhang   20   0 5576m 2.9g 3264 S  5.9  3.7  63:29.38 rootserver
319247 xlzhang   20   0 13.9g 2.3g 4840 S 96.7  2.9 637:29.16 mergeserver
```

drop table if exists test1;
Create table test1(c1 int primary key, c2 int,c3 float,c4 double,c5 decimal(30,10),c6 decimal(30,10),c7 decimal(30,10),c8 decimal(30,10),c9 decimal(30,10),c10 decimal(30,10));

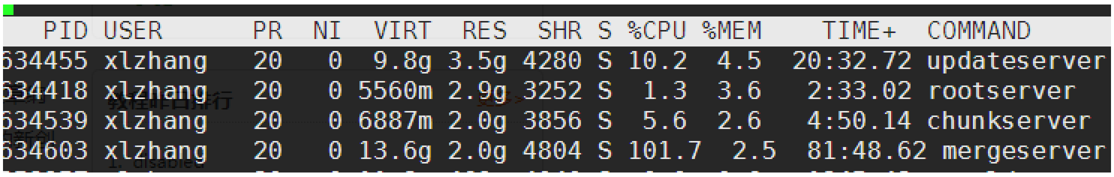
set @@session.ob_query_timeout=9000000000;
向表中插入 100w 行数据，然后循环执行以下操作：
select (((c2+c5)*(c3-c6)/c2-c7)*c8+c9)/c10 from test1;

执行 1 次时（42s）内存资源占用情况：



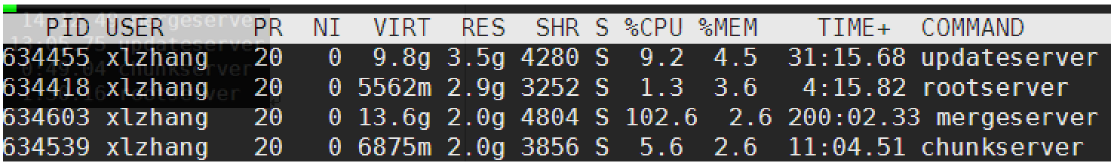
```
   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
634603 xlzhang   20   0 13.6g 1.9g 4804 S 102.0  2.5  14:12.49 mergeserver
634455 xlzhang   20   0 10.0g 5.0g 3948 S 26.7  6.4  13:05.75 updateserver
634539 xlzhang   20   0 5559m 1.4g 3548 S  9.9  1.8   0:49.04 chunkserver
634418 xlzhang   20   0 5548m 2.9g 3228 S  1.0  3.6   1:30.16 rootserver
```

执行 100 次时（4174s）内存资源占用情况：



```
   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
634455 xlzhang   20   0  9.8g 3.5g 4280 S 10.2  4.5  20:32.72 updateserver
634418 xlzhang   20   0 5560m 2.9g 3252 S  1.3  3.6   2:33.02 rootserver
634539 xlzhang   20   0 6887m 2.0g 3856 S  5.6  2.6   4:50.14 chunkserver
634603 xlzhang   20   0 13.6g 2.0g 4804 S 101.7  2.5  81:48.62 mergeserver
```

执行 300 次时（12597s）内存资源占用情况：



```
   PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
634455 xlzhang   20   0  9.8g 3.5g 4280 S  9.2  4.5  31:15.68 updateserver
634418 xlzhang   20   0 5562m 2.9g 3252 S  1.3  3.6   4:15.82 rootserver
634603 xlzhang   20   0 13.6g 2.0g 4804 S 102.6  2.6 200:02.33 mergeserver
634539 xlzhang   20   0 6875m 2.0g 3856 S  5.6  2.6  11:04.51 chunkserver
```

结论：无内存泄漏问题。

4. 测试 decimal 与其他数据类型的运算速度和代价

配置：单集群 194：RS+MS+CS+UPS

测试目的：测试 decimal 与其他数据类型的运算速度和运算代价

测试方法：

首先创建下面两张表：

drop table if exists test1;

Create table test1(c1 int primary key, c2 int,c3 float,c4 double,c5 decimal(30,10),c6 decimal(30,10),c7 decimal(30,10),c8 decimal(30,10),c9 decimal(30,10),c10 decimal(30,10));

drop table if exists test2;

Create table test2(k int primary key, c1 int,c2 float,c3 double, c4 double, c5 double, c6 double, c7 double ,c8 double ,c9 double, c10 double);

然后先向表中插入 100w 条数据：

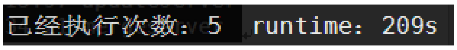
接着执行下面两条 select 语句，比较执行速度：

select (((c2+c5)*(c3-c6)/c2-c7)*c8+c9)/c10 from test1;

select (((c2+c5)*(c3-c6)/c2-c7)*c8+c9)/c10 from test2;

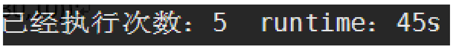
执行 5 次的时间分别为：

Decimal：



```
已经执行次数: 5  runtime: 209s
```

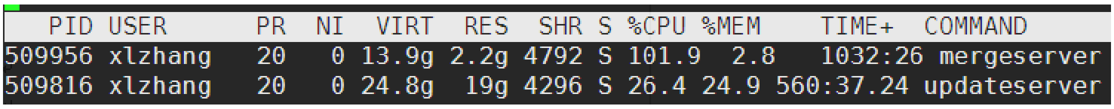
Double：



```
已经执行次数：5  runtime：45s
```

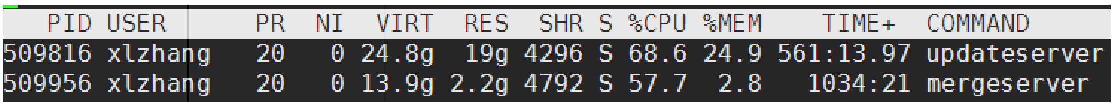
硬件资源使用情况：

Decimal：



```
   PID USER      PR  NI   VIRT   RES   SHR S %CPU %MEM     TIME+  COMMAND
509956 xlzhang   20   0 13.9g 2.2g 4792 S 101.9  2.8   1032:26 mergeserver
509816 xlzhang   20   0 24.8g  19g 4296 S 26.4 24.9 560:37.24 updateserver
```

Double：

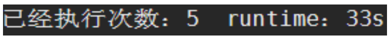


```
   PID USER      PR  NI   VIRT   RES   SHR S %CPU %MEM     TIME+  COMMAND
509816 xlzhang   20   0 24.8g  19g 4296 S 68.6 24.9 561:13.97 updateserver
509956 xlzhang   20   0 13.9g 2.2g 4792 S 57.7  2.8   1034:21 mergeserver
```

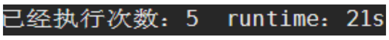
只是进行以下选择（无计算）操作，分别执行五次时间分别为 33s，21s

select c10 from test1;

select c10 from test2;

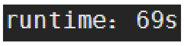


```
已经执行次数：5  runtime：33s
```



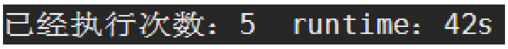
```
已经执行次数：5  runtime：21s
```

与旧版 decimal 测试对比情况：

五次的平均执行时间为： runtime：69s，而新 decimal 版本的 5 次平均执行时间为（209/5）=41s，所以相对旧版 decimal 版本执行时间提升了 69/41=1.68 倍。

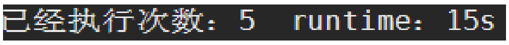
相同条件下 MYSQL 中执行 5 次选择（计算）的时间分别为

Decimal：



```
已经执行次数：5  runtime：42s
```

Double：



```
已经执行次数：5  runtime：15s
```

Mysql 中只是进行选择（无计算）操作执行五次的时间分别为 7s，11s；

所以实际计算时间的比例为：

OB：decimal 计算花费时间为(209-33)=176s，double 计算花费时间为(45-21)=24s

MYSQL：decimal 计算花费时间为(42-7)=36s，double 计算花费时间为(15-11)=4s

所以 OB 中 decimal 的计算代价是 MYSQL 中的 5 倍左右，OB 中的 double 是 MYSQL 中 double 为 6 倍左右。（待参考）

进行合并之后测试情况为：

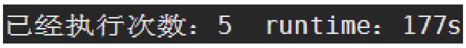
合并之后：

New:
Select * from test1 : 30.51s
Select c10 from test1 : 5.16s
select (((c2+c5)*(c3-c6)/c2-c7)*c8+c9)/c10 from test1; : 35.89s
合并之后 decimal 的计算花费为 : 30.73s

select * from test2 : 13.01s
select c1 from test2 : 2.15s
select (((c2+c5)*(c3-c6)/c2-c7)*c8+c9)/c10 from test2; : 4.48s
合并之后 double 的计算花费为 : 2.33s

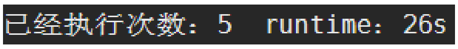
Old:
Select c10 from test1 : 4.05s
select (((c2+c5)*(c3-c6)/c2-c7)*c8+c9)/c10 from test1; : 79.56s
calc : 75.51s

另补：
选择（含计算）：



```
已经执行次数: 5  runtime: 177s
```

选择（无计算）：



```
已经执行次数: 5  runtime: 26s
```

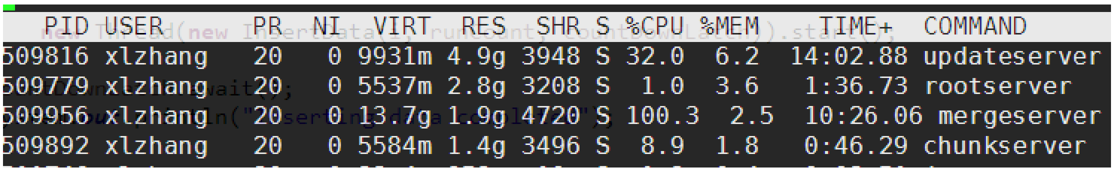
稳定性：

配置：194：RS+UPS+MS+CS
测试目的：长时间运行时会不会崩溃
测试方法：通过在 decimal 的计算查询过程中监控系统的资源情况来判断。
drop table if exists test1;
Create table test1(c1 int primary key, c2 int,c3 float,c4 double,c5 decimal(30,10));

计算表达式：(((c2+c5)*(c3-c5)/c2-c5)*c5+c4)/c5

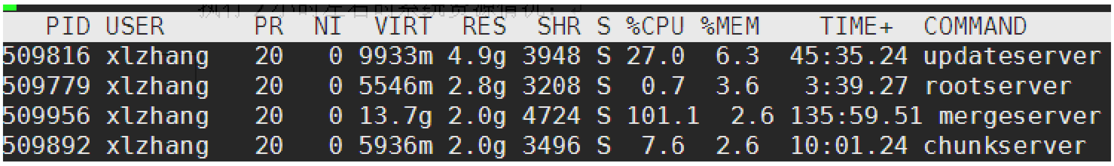
向表中插入 100w 行数据，然后循环执行以下操作：
set @@session.ob_query_timeout=9000000000;
select (((c2+c5)*(c3-c5)/c2-c5)*c5+c4)/c5 from test1;

刚开始时（1 分钟左右）系统资源情况：



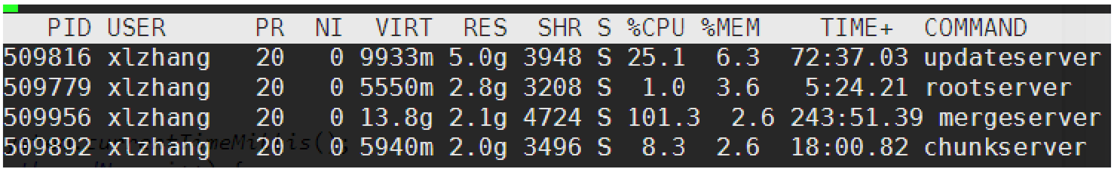
```
  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
509816 xlzhang   20   0 9931m 4.9g 3948 S 32.0  6.2  14:02.88 updateserver
509779 xlzhang   20   0 5537m 2.8g 3208 S  1.0  3.6   1:36.73 rootserver
509956 xlzhang   20   0 13.7g 1.9g 4720 S 100.3  2.5  10:26.06 mergeserver
509892 xlzhang   20   0 5584m 1.4g 3496 S  8.9  1.8   0:46.29 chunkserver
```

执行 2 小时左右时系统资源情况：



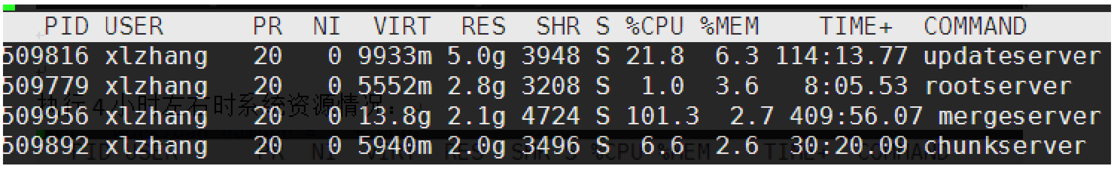
```
   PID USER        PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
509816 xlzhang     20   0 9933m 4.9g 3948 S 27.0  6.3  45:35.24 updateserver
509779 xlzhang     20   0 5546m 2.8g 3208 S  0.7  3.6   3:39.27 rootserver
509956 xlzhang     20   0 13.7g 2.0g 4724 S 101.1  2.6 135:59.51 mergeserver
509892 xlzhang     20   0 5936m 2.0g 3496 S  7.6  2.6  10:01.24 chunkserver
```

执行 4 小时左右时系统资源情况：



```
   PID USER        PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
509816 xlzhang     20   0 9933m 5.0g 3948 S 25.1  6.3  72:37.03 updateserver
509779 xlzhang     20   0 5550m 2.8g 3208 S  1.0  3.6   5:24.21 rootserver
509956 xlzhang     20   0 13.8g 2.1g 4724 S 101.3  2.6 243:51.39 mergeserver
509892 xlzhang     20   0 5940m 2.0g 3496 S  8.3  2.6  18:00.82 chunkserver
```

执行 8 小时左右时系统资源情况：



```
   PID USER        PR  NI  VIRT  RES  SHR S %CPU %MEM     TIME+  COMMAND
509816 xlzhang     20   0 9933m 5.0g 3948 S 21.8  6.3 114:13.77 updateserver
509779 xlzhang     20   0 5552m 2.8g 3208 S  1.0  3.6   8:05.53 rootserver
509956 xlzhang     20   0 13.8g 2.1g 4724 S 101.3  2.7 409:56.07 mergeserver
509892 xlzhang     20   0 5940m 2.0g 3496 S  6.6  2.6  30:20.09 chunkserver
```