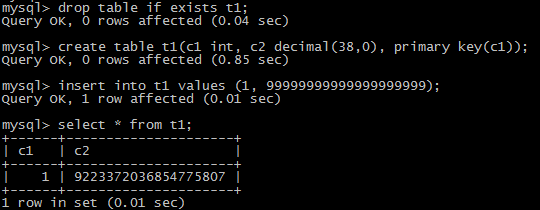
1. 针对一个decimal类型属性，在插入数据时（未超精度范围），插入结果会出错。示例如下：

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;



结果错误（**已知错误，其使用手册中有说明**）

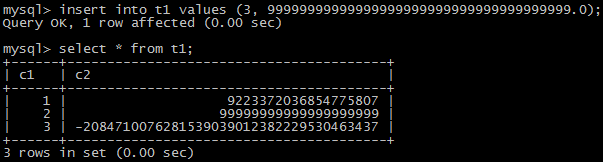
按照其使用手册的使用方法，在上述数据后面加上小数，就可正常插入，但是实际执行结果依然错误。

insert into t1 values (2, 99999999999999999999.0);

select \* from t1;

insert into t1 values (3, 99999999999999999999999999999999999999.0);

select \* from t1;



该错误已解决

1. 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));

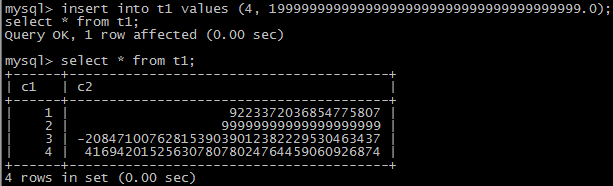
insert into t1 values (1, 99999999999999999999);

insert into t1 values (2, 99999999999999999999.0);

insert into t1 values (3, 99999999999999999999999999999999999999.0);

insert into t1 values (4, 199999999999999999999999999999999999999.0);

select \* from t1;



该错误已解决

1. 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）



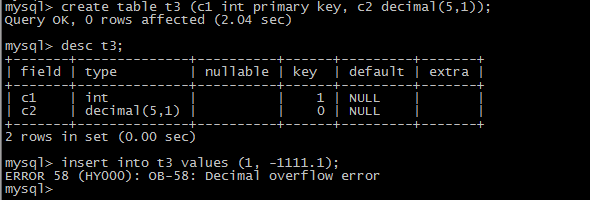
该错误已解决

1. 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);



该错误已解决

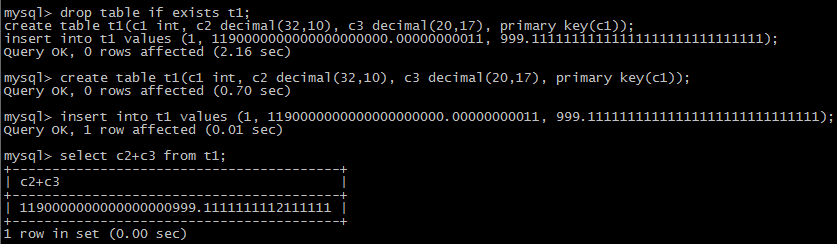
1. 两个decimal类型属性做加法运算时，精度控制与文档描述不符。示例如下：

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, 999.11111111111111111111111111111);

select c2+c3 from t1;





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）

1. 报错信息不规范：

drop table if exists t1;

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



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));



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);



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）

1. 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));

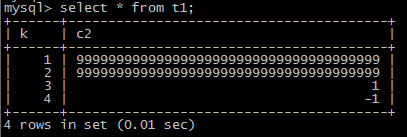
insert into t1 values (1, 0);

insert into t1 values (2, 0.1);

insert into t1 values (3, 1.1);

insert into t1 values (4, -1);

select \* from t1;



（已经修改）

1. decimal加法运算正确性有误，示例如下：

Cedar：

drop table if exists t1;

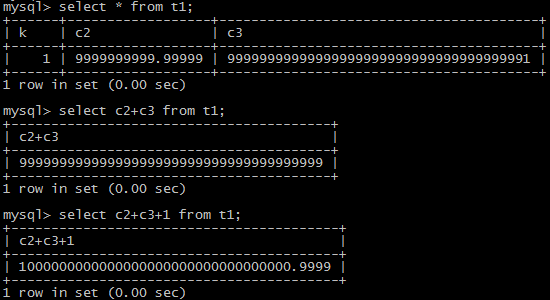
create table t1(k int primary key, c2 decimal(15,5), c3 decimal(38,0));

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

select \* from t1;

select c2+c3 from t1;

select c2+c3+1 from t1;



MySQL：

drop table if exists t1;

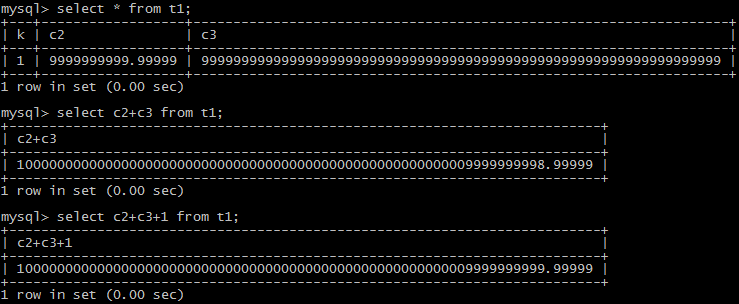
create table t1(k int primary key, c2 decimal(15,5), c3 decimal(65,0));

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

select \* from t1;

select c2+c3 from t1;

select c2+c3+1 from t1;



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）

1. decimal减法运算正确性有误，示例如下：

drop table if exists t1;

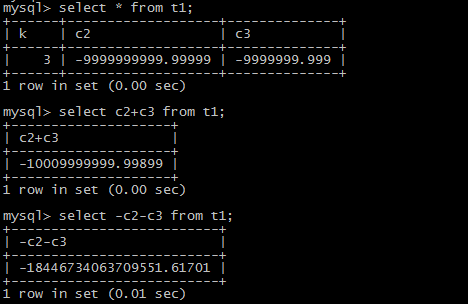
create table t1(k int primary key, c2 decimal(15,5), c3 decimal(10,3));

insert into t1 values (3, -9999999999.99999, -9999999.999);

select \* from t1;

select c2+c3 from t1;

select -c2-c3 from t1;



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）**

1. 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);

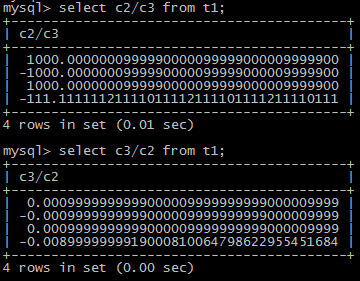
insert into t1 values (2, -9999999999.99999, 9999999.999);

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;



DIV精度控制规则：



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）

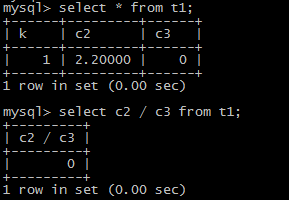
1. 一个decimal类型数据与一个bool类型数据作除法运算时，如果被除数类型为bool且值为false，正确结果应为Null，但实际结果为0。示例如下：

drop table if exists t1;

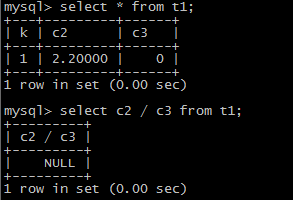
create table t1(k int primary key, c2 decimal(15,5), c3 bool);

insert into t1 values (1, 2.2, False);

select c2 / c3 from t1;



相应的MySQL结果为：



（NULL 和 0 的冲突）

1. 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。

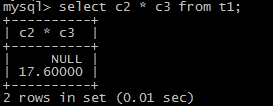
drop table if exists t1;

create table t1(k int primary key, c2 decimal(15,5), c3 varchar(100));

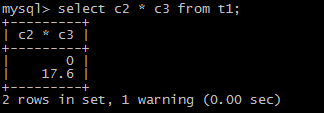
insert into t1 values (1, 2.2, 'zsdsa8');

insert into t1 values (2, 2.2, '8');

select c2 \* c3 from t1;



相应的MySQL结果为：



（NULL 和 0 的冲突）

1. 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;

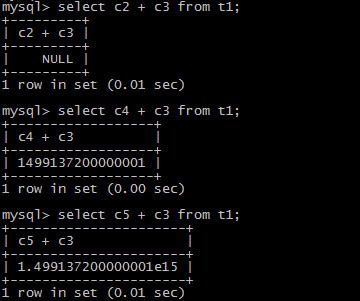
create table t1(k int primary key, c2 decimal(15,5), c3 datetime, c4 int, c5 double);

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;



**（已解决）**

1. 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);

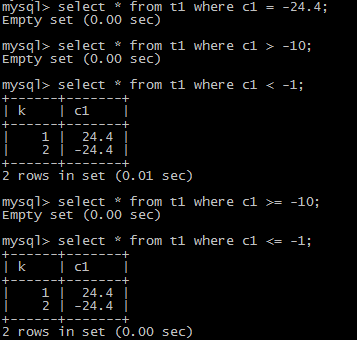
select \* from t1 where c1 = -24.4;

select \* from t1 where c1 > -10;

select \* from t1 where c1 < -1;

select \* from t1 where c1 >= -10;

select \* from t1 where c1 <= -1;



已经解决（int->decimal类型转换函数）

1. 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);

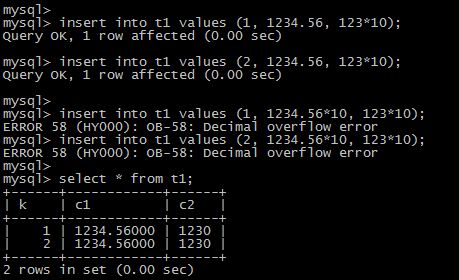
insert into t1 values (1, 1234.56, 123\*10);

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;



(已经解决，修改ob\_expr\_obj.cpp中的mul函数，添加varchar空间)

1. 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;

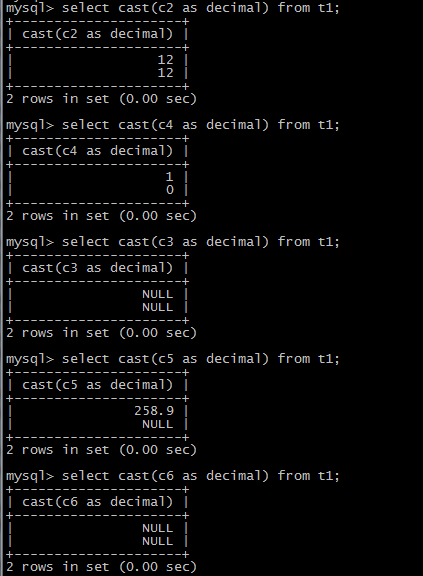
select cast(c4 as decimal) from t1;

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！

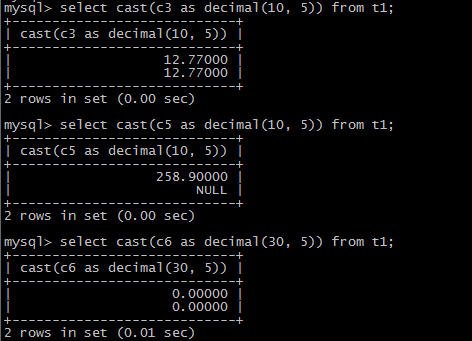


注意：int、bool类型数据的处理正确，double，varchar（含字母），datatime类型数据处理有误，上述是在不指定精度情况下测试的，**需在使用文档中说明默认精度的选择（若不支持该种写法，应该报语法错误）。**

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;



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）

1. 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);

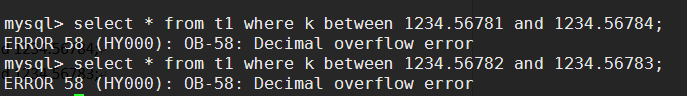
insert into t1 values (1234.56782, 1234.4322);

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;



（ob\_sql\_read\_strategy.cpp:522）（**已经解决**）

1. **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：**

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);



（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）

1. 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;

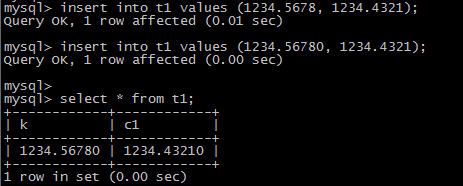
create table t1(k decimal(10,5) primary key, c1 decimal(10,5));

insert into t1 values (1234.5678, 1234.4321);

insert into t1 values (1234.56780, 1234.4321);

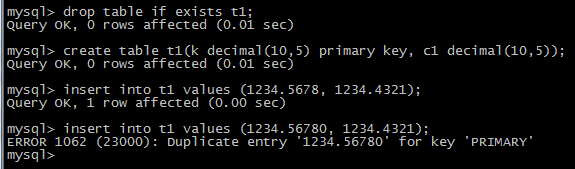
select \* from t1;

(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)



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：



1. 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);

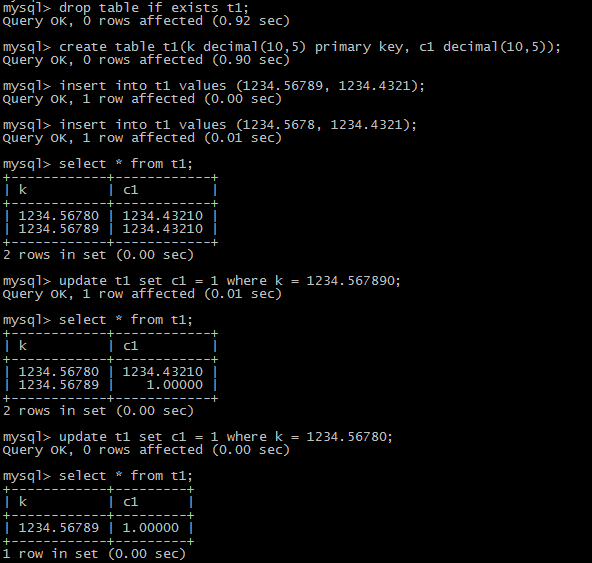
insert into t1 values (1234.5678, 1234.4321);

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)



1. 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));

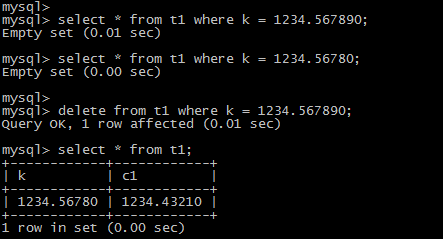
insert into t1 values (1234.56789, 1234.43212);

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)



注意：select会出错，但是delete正常。

（已解决）

1. 在decimal作为主键时，主键自增功能有误。示例：

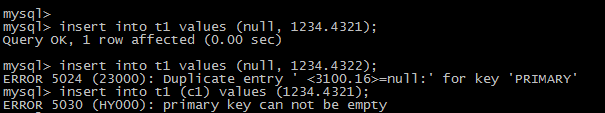
drop table if exists t1;

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);



（已解决，但是插NULL值仍存在）

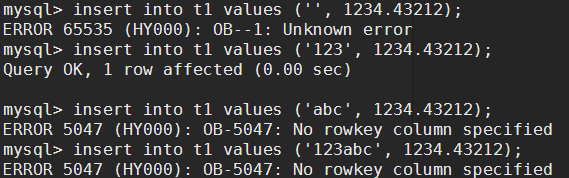
1. Decimal作为主键时插入varchar类型时报错信息不规范：

drop table if exists t1;

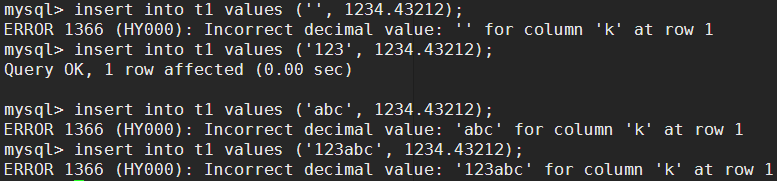
create table t1(k decimal(10,5) primary key, c1 decimal(10,5));

insert into t1 values ('', 1234.43212);

CEDAR中报错信息：



对应的MySQL中报错信息：



（已解决）

1. 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：

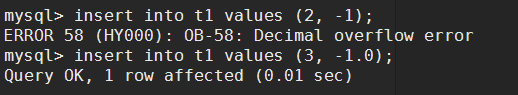
drop table if exists t1;

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);



（已解决）

1. 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));

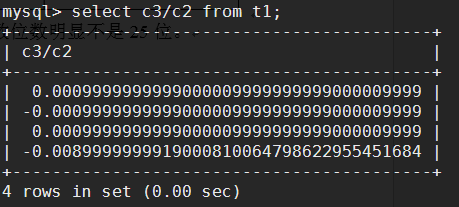
insert into t1 values (1, 9999999999.99999, 9999999.999);

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;



DIV精度控制规则：

int\_len：运算结果整数有效位



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）

1. 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;



1. 在decimal测试中，插入语句中会出现以下错误：

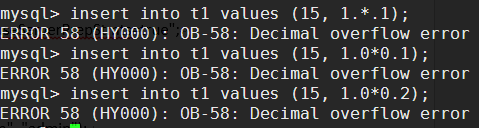
drop table if exists t1;

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);



（已解决）

1. 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())

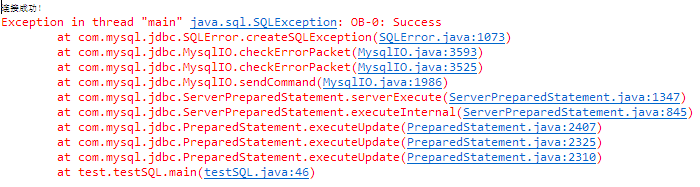
{

System.out.println(rs2.getInt(1)+" "+rs2.getBigDecimal(2));

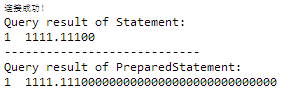
}

执行结果：

prepareStatement代码插入：



prepareStatement查询结果:



（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））

1. 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;

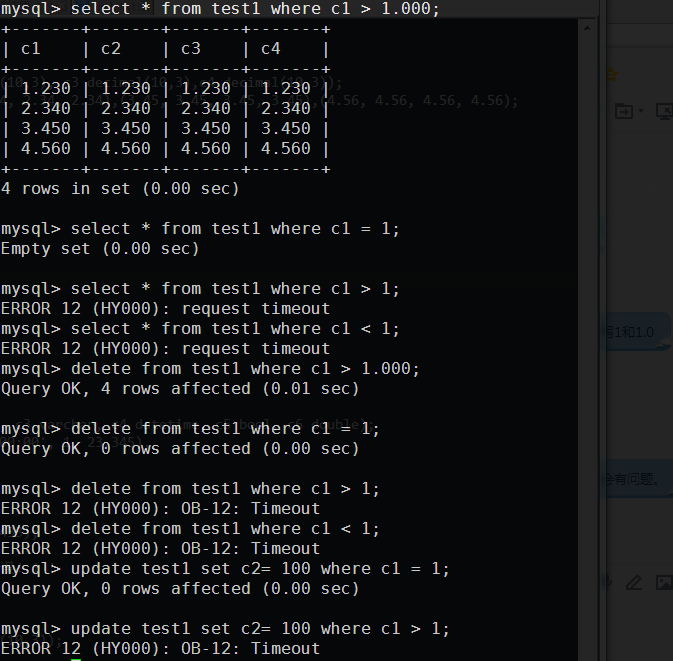
delete from test1 where c1 < 1;

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);



（已解决）

1. 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);

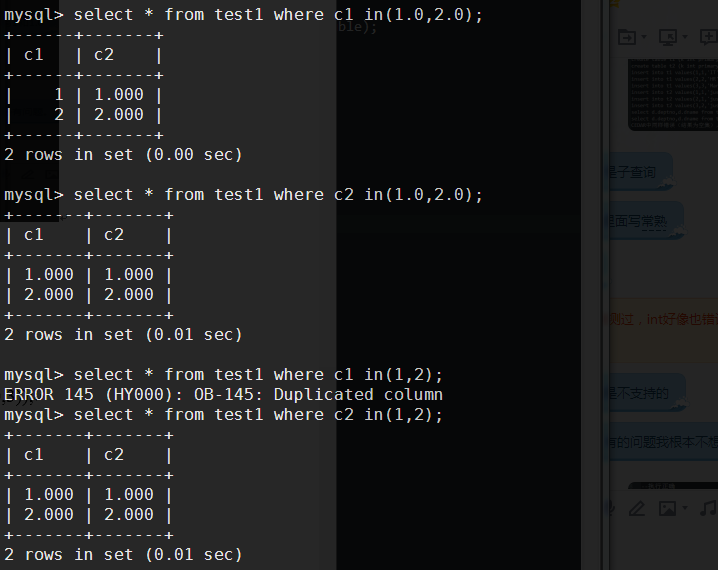
insert into test1 values(2.0,2.0);

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);



（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）

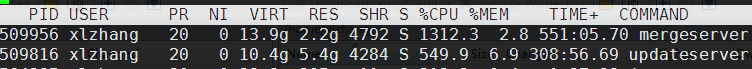
set @@session.ob\_query\_timeout=9000000000;

运行jar包

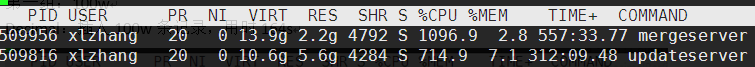
测试结果如下：

第一组：100w

Decimal：插入100w条记录，用时54s（mysql 40s）

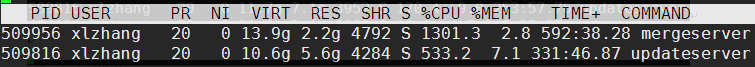


Double：插入100w条记录，用时45s（mysql 36s）

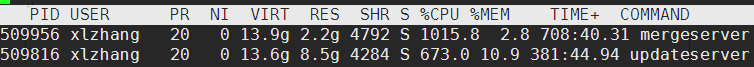


第二组：500w

Decimal：插入500w条记录，用时270s

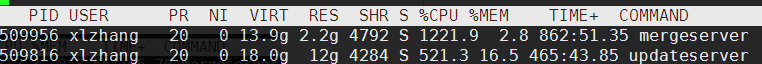


Double：插入500w条记录，用时239s

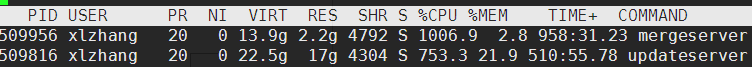


第三组：1000w

Decimal：插入1000w条记录，用时541s



Double：插入1000w条记录，用时403s



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。

1. 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）

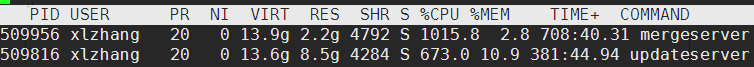
drop table if exists test2;

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);

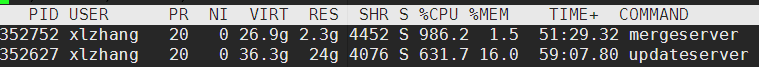
最后比较测试结果。

测试结果如下：

Double（含decimal版本，分支Decimal\_optimize）：插入500w记录，用时239s



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



1. 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));

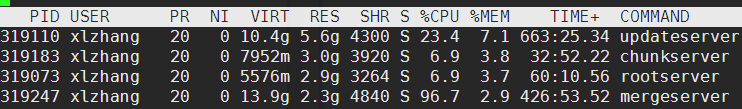
计算表达式：(((c2+c5)\*(c3-c5)/c2-c5)\*c5+c4)/c5

向表中插入100w行数据，然后循环执行以下操作：

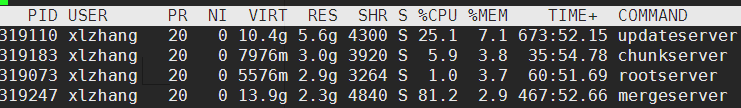
set @@session.ob\_query\_timeout=9000000000;

select (((c2+c5)\*(c3-c5)/c2-c5)\*c5+c4)/c5 from test1;

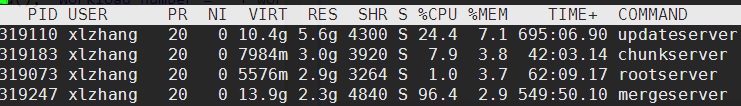
执行1次时内存资源占用情况：



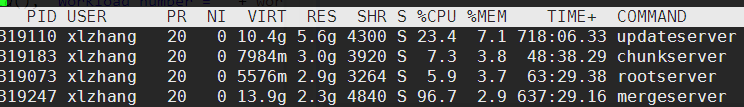
执行100次时内存资源占用情况：



执行300次时内存资源占用情况：



执行500次时内存资源占用情况：



drop table if exists test1;

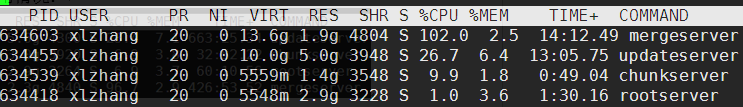
Create table test1(c1 int primary key, c2 int,c3 float,c4 double,c5 decimal(30,10),c6 decimal(30,10),c7 decimal(30,10),c8 decimal(30,10),c9 decimal(30,10),c10 decimal(30,10));

set @@session.ob\_query\_timeout=9000000000;

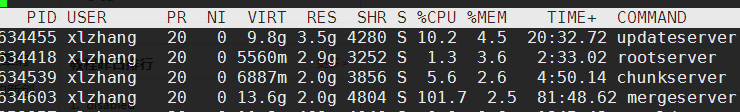
向表中插入100w行数据，然后循环执行以下操作：

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

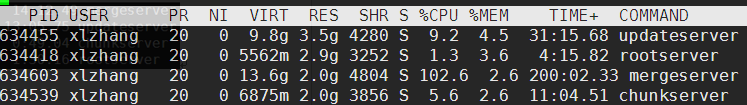
执行1次时（42s）内存资源占用情况：



执行100次时（4174s）内存资源占用情况：



执行300次时（12597s）内存资源占用情况：



结论：无内存泄漏问题。

1. 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：

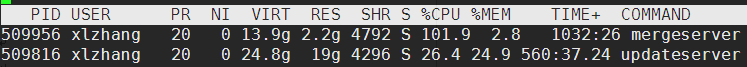


Double：

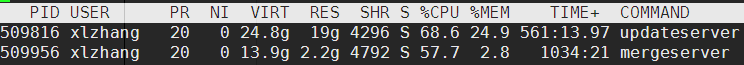


硬件资源使用情况：

Decimal：



Double：



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;





与旧版decimal测试对比情况：

五次的平均执行时间为：，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：



Double：



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）：



选择（无计算）：



**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));

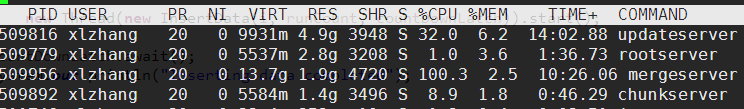
计算表达式：(((c2+c5)\*(c3-c5)/c2-c5)\*c5+c4)/c5

向表中插入100w行数据，然后循环执行以下操作：

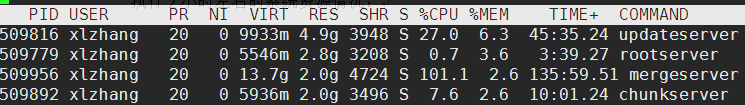
set @@session.ob\_query\_timeout=9000000000;

select (((c2+c5)\*(c3-c5)/c2-c5)\*c5+c4)/c5 from test1;

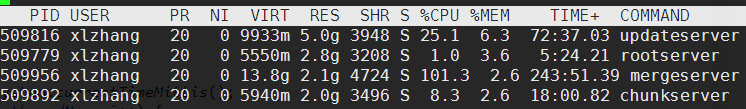
刚开始时（1分钟左右）系统资源情况：



执行2小时左右时系统资源情况：



执行4小时左右时系统资源情况：



执行8小时左右时系统资源情况：

