
Subject: [BUG] MySQL error 2014 while reading output from stored procedure.
Posted by [Klugier](#) on Fri, 15 Feb 2013 16:20:14 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hello,

First of all, I have created following stored procedure on MySQL server:

```
DELIMITER //

CREATE PROCEDURE selectCurrentUserInfo ()
LANGUAGE SQL
DETERMINISTIC
BEGIN
  DECLARE userID INT;

  SET userID = (SELECT `ID` FROM `LoginInfo`);

  IF !ISNULL (userID) THEN
    SELECT `Rights`, `FirstName`, `Surname` FROM Users WHERE `ID`=(SELECT userID);
  END IF;
END;

//
DELIMITER ;
```

Next I have created method that can read the select output(C++):

```
User DatabaseUsers::getCurrentUserInfo () {
  User user;
  String query = "CALL selectCurrentUserInfo()";

  if (SQL.Execute (query)) {
    while (SQL.Fetch () && SQL.GetColumns () == 3) {
      /* READ THE SELECT TABLE - Works! */
    }
  }

  return User;
}
```

The method works fine, but after executing query program lost synchronization with MySQL server. (MySQL 2014 error - Commands out of sync).

How to solve the problem? Let's look more closely to the following upp method (MySQL -> MySQL.cpp):

```
bool MySqlConnection::Execute() {
    String query;
    int pi = 0;
    const char *s = statement;
    while(s < statement.End())
        if(*s == "\"" || *s == "\'")
            s = MySqlReadString(s, query);
        else {
            if(*s == '?')
                query.Cat(param[pi++]);
            else
                query.Cat(*s);
            s++;
        }
    Cancel();
    if(!MysqlQuery(query))
        return false;
    result = mysql_store_result(mysql);
    rows = (int)mysql_affected_rows(mysql);
    if(result) {
        int fields = mysql_num_fields(result);
        info.SetCount(fields);
        convert.Alloc(fields, false);
        for(int i = 0; i < fields; i++) {
            MYSQL_FIELD *field = mysql_fetch_field_direct(result, i);
            SqlColumnInfo& f = info[i];
            f.name = field->name;
            switch(field->type) {
                case FIELD_TYPE_TINY:
                case FIELD_TYPE_SHORT:
                case FIELD_TYPE_LONG:
                case FIELD_TYPE_INT24:
                    f.type = INT_V;
                    break;
                case FIELD_TYPE_LONGLONG:
                case FIELD_TYPE_DECIMAL:
                case FIELD_TYPE_FLOAT:
                case FIELD_TYPE_DOUBLE:
                    f.type = DOUBLE_V;
                    break;
                case FIELD_TYPE_DATE:
                    f.type = DATE_V;
                    break;
                case FIELD_TYPE_DATETIME:
```

```

case FIELD_TYPE_TIMESTAMP:
    f.type = TIME_V;
    break;
case FIELD_TYPE_VAR_STRING:
case FIELD_TYPE_STRING:
    convert[i] = true;
default:
    f.type = STRING_V;
    break;
}
f.width = field->length;
f.scale = f.precision = 0;
}
}
else {
    lastid = (int)mysql_insert_id(mysql);
    if(lastid) {
        SqlColumnInfo& f = info.Add();
        f.width = f.scale = f.precision = 0;
        f.binary = false;
        f.type = DOUBLE_V;
        f.name = "LAST_INSERT_ID";
        rows = 1;
    }
}
return true;
}

```

What's wrong with this method? It can stored only one result, but my procedure produces more than one result. This is what i have done to fix this issue:

```

bool MySqlConnection::Execute() {
    String query;
    int pi = 0;
    const char *s = statement;
    while(s < statement.End())
        if(*s == "\"" || *s == "\'")
            s = MySqlReadString(s, query);
        else {
            if(*s == '?')
                query.Cat(param[pi++]);
            else
                query.Cat(*s);
            s++;
        }
    Cancel();
}

```

```

if(!MysqlQuery(query))
    return false;
result = mysql_store_result(mysql);

/* FIX - we can read all the results from query */
while (mysql_more_results (mysql)) {
    int tempResult = mysql_next_result (mysql); // <- Temporary solution: we are losing rest of
results
}

rows = (int)mysql_affected_rows(mysql);
if(result) {
    int fields = mysql_num_fields(result);
    info.SetCount(fields);
    convert.Alloc(fields, false);
    for(int i = 0; i < fields; i++) {
        MYSQL_FIELD *field = mysql_fetch_field_direct(result, i);
        SqlColumnInfo& f = info[i];
        f.name = field->name;
        switch(field->type) {
            case FIELD_TYPE_TINY:
            case FIELD_TYPE_SHORT:
            case FIELD_TYPE_LONG:
            case FIELD_TYPE_INT24:
                f.type = INT_V;
                break;
            case FIELD_TYPE_LONGLONG:
            case FIELD_TYPE_DECIMAL:
            case FIELD_TYPE_FLOAT:
            case FIELD_TYPE_DOUBLE:
                f.type = DOUBLE_V;
                break;
            case FIELD_TYPE_DATE:
                f.type = DATE_V;
                break;
            case FIELD_TYPE_DATETIME:
            case FIELD_TYPE_TIMESTAMP:
                f.type = TIME_V;
                break;
            case FIELD_TYPE_VAR_STRING:
            case FIELD_TYPE_STRING:
                convert[i] = true;
            default:
                f.type = STRING_V;
                break;
        }
        f.width = field->length;
        f.scale = f.precision = 0;
    }
}

```

```
}  
}  
else {  
    lastid = (int)mysql_insert_id(mysql);  
    if(lastid) {  
        SqlColumnInfo& f = info.Add();  
        f.width = f.scale = f.precision = 0;  
        f.binary = false;  
        f.type = DOUBLE_V;  
        f.name = "LAST_INSERT_ID";  
        rows = 1;  
    }  
}  
return true;  
}
```

This is only temporary solution (We can read only first select output from procedure, but we don't lose synchronization with MySQL server).
