Subject: ODBC Assertion failed
Posted by Giorgio on Thu, 17 Dec 2020 10:01:38 GMT
View Forum Message <> Reply to Message

Hi there,
I have an application that has to insert some data on a Microsoft SQL Server db.

When the query runs without error the application is ok, the problem is when the query fails for some reason: in that case the application crashes.

This is what I found in the log:

ERROR [Microsoft][ODBC SQL Server Driver][SQL Server]Violation of PRIMARY KEY constraint 'PK_RIG'. Cannot insert duplicate key in object 'dbo.DOROW'. The duplicate key value is [...]
********** ASSERT FAILED: Assertion failed in C:\Users\GIRU1\ToolsBox\upp-2020.1\uppsrc\ODBC\ODBC.cpp, line 215
tlevel >= 0

This is the code that crashes:

```
bool InsertDoument(ValueMap * vm){

 Sql query(my_mssql_db);
 query.ClearError();

 query.Begin();

 try{
  query * Insert(My_mssql_table)(*vm);
 } catch(SqlExc) {
  ErrorOK(query.GetLastError());
 }

 if(query.WasError()){
  query.Rollback();
  return false;
 }

 query.Commit();
    return true;
}
```

Thanks for any hint!
gio

Subject: Re: ODBC Assertion failed
Posted by JeyCi on Thu, 17 Dec 2020 13:09:30 GMT
View Forum Message <> Reply to Message

Giorgio wrote on Thu, 17 December 2020 11:01if (query.WasError())
I'm not sure that you should use 2 error checks - both "try_catch" & "if (query.WasError())"... I think WasError can be enough & you'd better try to do the whole necessary stuff here in this scope (including the one that you're trying to do in catch)... then "try_catch" seems unneeded for me...
I'm not sure but it seems to be excessive error checks when using both... because in catch-scope you are not correctly closing your connection to process the error that arrised, but still continue to utilize it (connection) for error checks again... imho

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Thu, 17 Dec 2020 13:22:13 GMT
View Forum Message <> Reply to Message

Uhm, I tried with the try/catch only, with .WasError() only, with both of them, with none of them: always same result. :(

Subject: Re: ODBC Assertion failed
Posted by JeyCi on Thu, 17 Dec 2020 14:06:50 GMT
View Forum Message <> Reply to Message

Giorgio wrote on Thu, 17 December 2020 11:01Violation of PRIMARY KEY
In any case it is the problem that should be solved at Server-side & SQL-statement better... before insertion check for Uniqueness in some Join or take Distinct Group_By - to assure that rows to be inserted are really unique - as PKs ought to be...
If I had a chance to meet some duplication of PKs - I would first make Query for them (with a help of Self-Join), to understand what rows are really problematic & how to correct input data, then make necessary corrections either manually or in code - to have correct data, filtered from duplications, and then try again to insert into DB...
SQL language has pretty enough stuff to deal with such situations - e.g. Triggers...
An attempt to insert 2 identical PKs into DB should be filtered somehow in the Query itself, not in application language... though of course the last is also possible - to check duplications in C++ before Insert Into DB - but it is not very convenient...

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Thu, 17 Dec 2020 15:27:11 GMT
View Forum Message <> Reply to Message

I do not agree, checking the uniqueness beforehand could be a useful workaround, but an application cannot crash because someone is trying to enter wrong data.

Subject: Re: ODBC Assertion failed
Posted by JeyCi on Thu, 17 Dec 2020 16:44:04 GMT
View Forum Message <> Reply to Message

well, your disagreement also seems reasonable...
Giorgio wrote on Thu, 17 December 2020 11:01upp-2020.1\uppsrc
... additional check needed from upp-team who use this version... I cannot help with any other suggestions because am using older version & cannot reproduce the situation...
with sqlite3 WasError() catches PK-duplication error & after PromptOK-notification within it (that I use to see GetErrorStatement) normal return from WasError()-scope & from function without crash of app...

Subject: Re: ODBC Assertion failed
Posted by JeyCi on Fri, 18 Dec 2020 05:26:23 GMT
View Forum Message <> Reply to Message

BTW, I can just suggest to make a couple of experiments:
1) try to log your queries with something like (but about your ms sql server)

```
sqlite3.Open(db)
#ifdef _DEBUG
   sqlite3.SetTrace();  //logging queries
#endif
```

& without Commit & Rollback... AFAIK, such command is being self-rollbacked automatically in the case of error or commited automatically if no errors - in ms sql server - but I have no chance to check it in ms sql server

```
// initialize transaction
BEGIN TRAN;
// transaction
INSERT INTO sales
VALUES ('7896', 'JR3435', 'Oct 28 1997', 25, 'Net 60', 'BU7832');
```

2) I'm not sure, that having duplicated PKs the query starts anyway, therefore I'm not sure that you have the right to do Rollback when this error arrised...
therefore you can try just SQL-statement to .Execute, expressing your sql-command in sql-string, not u++ method Insert... something like this - that is doing rollback ONLY IF @@TRANCOUNT > 0

```
BEGIN TRY
   BEGIN TRANSACTION
```

```
        exec( @sqlHeader)
    COMMIT
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK
END CATCH
```

or check this count in U++ somehow, before rollback - though it seems meaningless in the case of error... If you'd had several queries - you'd had something to rollback if previous queries already done... but in a single Insert failed - nothing to Rollback... imho

---

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Fri, 18 Dec 2020 07:26:46 GMT
View Forum Message <> Reply to Message

I've found out that the problem arises only in debug mode. The problem is the ASSERT at line 200 and 215 in ODBC/ODC.cpp. I can live without having the capability to debug that part of code, although it's very handy when there are issues (a problem with that part of the code was the reason I turned on the debugger in the first place). Honestly my C++ isn't good enough to fix that, so I hope that someone of the UPP's developer can notice this thread and have a look.

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Fri, 18 Dec 2020 07:32:22 GMT
View Forum Message <> Reply to Message

Honestly I think that you are approaching the problem in the wrong way. Found some kind of workaround can be a quick & dirty solution, but is not a long term solution. I've ton of sql queries towards a PostgreSQL rdbms in my application: never had a glitch. That it's clearly some kind of bug in the ODBC code.

Subject: Re: ODBC Assertion failed
Posted by mirek on Fri, 18 Dec 2020 09:22:33 GMT
View Forum Message <> Reply to Message

Giorgio wrote on Fri, 18 December 2020 08:32Honestly I think that you are approaching the problem in the wrong way. Found some kind of workaround can be a quick & dirty solution, but is not a long term solution. I've ton of sql queries towards a PostgreSQL rdbms in my application: never had a glitch. That it's clearly some kind of bug in the ODBC code.

I absolutely agree. Will investigate soon (I have not seen this kind of behaviour the last time I used

ODBC).

Mirek

Subject: Re: ODBC Assertion failed
Posted by mr_ped on Fri, 18 Dec 2020 13:02:48 GMT
View Forum Message <> Reply to Message

Giorgio wrote on Fri, 18 December 2020 08:26I've found out that the problem arises only in debug mode. The problem is the ASSERT at line 200 and 215 in ODBC/ODC.cpp. I can live without having the capability to debug that part of code, although it's very handy when there are issues (a problem with that part of the code was the reason I turned on the debugger in the first place).

Usually ASSERT compiles to nothing in release, and to debug check under debug.

So "arises only in debug mode" means that you see the warning in debug mode, but in release the same condition, which would trigger assert, does nothing and continues further.

ASSERT usually ensures conditions which "must be true" (and there is no need to check it at runtime) and the code after them usually is written in such a way that it may often crash hard if the condition is false.

So you get assert warning only in debug mode, but most likely the real problem is happening in release too, it's just not catched early by ASSERT, but causes havoc further down the line. In some situations you may be lucky that the code after is robust enough to survive even when condition is false, and the assert is rather just information that something is seriously broken in the code, way beyond the author expectations.

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Mon, 21 Dec 2020 08:54:34 GMT
View Forum Message <> Reply to Message

Hi,
I reviewed my code, but I actually cannot see anything terrible. I have innumerable queries in PostgreSQL that is the rdbms that I currently use, previously I used MySql so I wrote a lot queries with that also, have also a few in sqlite... never a problem. The structure of the code is in my first post: it seems to me very simple and basic. I also tried different combinations of try/catch and .WasErrors() but the behaviour is also the same. Maybe it's the way I open the connection to the db?

(mymssql_db is a private memeber of the class and its type is MSSQLSession).

bool OpenMSSqlDB(){

```
 String username = "sa";
 String pwd = "mypassword";
 String dbname = "MYDBNAME";
 String server = "servername\\instancename";
 String cs = "Driver={SQL Server};Server=" + server +
     ";UID=" + username + ";PWD=" + pwd + ";Database=" + dbname + ";";
 if(!mymssql_db.Connect(cs))
 {
  string err="Unable to connect to MSSql database due to the following error: " +
mymssql_db.GetLastError().ToStd();
  this->err_messages->push_back(err);
  return false;
 }

 mymssql_db.ThrowOnError();
 mymssql_db.LogErrors();
 mymssql_db.SetTrace();

 return true;
}
```

So if you could give me some hints about some tests to perform or what to look for in my code I could really use it.

Thanks,
gio

Subject: Re: ODBC Assertion failed
Posted by JeyCi on Mon, 21 Dec 2020 14:33:11 GMT
View Forum Message <> Reply to Message

was this correct in your whole code?
Quote:SQL = myDB;
Sql query(myDB);
the insertion below is working for me in Sqlite3 - try to adopt the code to your MS_SQL_Server_connection (your Commit, Rollback & WasError() are included in this code, I just can not check for ms_sql_server & therefore cannot provide adopted example)... but such is my view on insertion

```
#include <Core/Core.h>
#include <plugin/sqlite3/Sqlite3.h>

using namespace Upp;
```

```
CONSOLE_APP_MAIN
{
 Cout() << "Start" << EOL;

 Sqlite3Session myDB; // <<<<<<<<<<<<<< ADOPT TO YOUR MS_Sql_Server.....
 if(!myDB.Open(ConfigFile("DATA.db"))) {
  LOG("Can't create or open database file\n");
  return;
 }
#ifdef _DEBUG
 myDB.SetTrace();
#endif

 SQL = myDB;
 // ======================= initial data
 String s = "[{\"A\":\"valA1\", \"B\":\"valB1\"}, {\"A\":\"valA2\", \"B\":\"valB2\"}]";
 ValueMap js=ParseJSON(s);   // frankly speaking ValueArray should be perhaps, but it works for
me...
 int jsz=js.GetCount();

 // ======================= create tbl
 SQL.Execute("drop table IF EXISTS tbl;");
 SQL.Execute("create table IF NOT EXISTS tbl (A STRING , B STRING, PRIMARY KEY (A));");

 // ======================= insert into tbl
 Sql query(myDB);  // (my_db)
 query.SetStatement("INSERT INTO tbl ( A, B) VALUES ( ?, ?) ");
 // initialize transaction
 query.Begin();
        // loop to insert rows
 for(size_t i = 0; i < jsz; i++) {
     query.SetParam(0,~js[i]["A"]);
     query.SetParam(1,~js[i]["B"]);
  query.Execute();
 }
 // error behavior
 if(query.WasError()){
  Cout() << query.GetLastError();
  query.Rollback();
  return;  // false;
 }
 //Save data & unlock db
 query.Commit();
 Cout() << "Done" << EOL;

 // ======================= select to console_view
 if (query.Execute("SELECT * FROM tbl"))
 {
```

```
  Cout() << "Call succeeded" << EOL;

  for (int i = 0; i < query.GetColumnCount(); ++i) {
   Cout() << Format("%d: %s", i, query.GetColumnInfo(i).name) << EOL;
  }

  while (query.Fetch()) {
   Cout() << query[0] << " ; " << query[1] << EOL;
  }
 }
 else
 {
  Cout() << "Call failed" << EOL;
  Cout() << SQL.GetLastError() << EOL;
 }
}
```

if you put "valA1" to both A-key values in json-string -- you will catch
Quote:SQL logic error
p.s.
I don't know whether SqlMassInsert will be quicker, but using Transactions gives the possibility for Rollback - I am not sure about this possibility in SqlMassInsert, perhaps it is being done automatically for this class - in LOG should be seen, - but I didn't check SqlMassInsert& UseTransaction(bool b = true) == testing needed to compare the speed of 2 approaches (my & SqlMassInsert)
Wish you good luck if my code could be helpful for you... or somebody can notice some mistakes with a fresh_eye :blush:

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Tue, 22 Dec 2020 10:44:01 GMT
View Forum Message <> Reply to Message

mr_ped wrote on Fri, 18 December 2020 14:02Giorgio wrote on Fri, 18 December 2020 08:26I've found out that the problem arises only in debug mode. The problem is the ASSERT at line 200 and 215 in ODBC/ODC.cpp. I can live without having the capability to debug that part of code, although it's very handy when there are issues (a problem with that part of the code was the reason I turned on the debugger in the first place).

Usually ASSERT compiles to nothing in release, and to debug check under debug.

So "arises only in debug mode" means that you see the warning in debug mode, but in release the same condition, which would trigger assert, does nothing and continues further.

ASSERT usually ensures conditions which "must be true" (and there is no need to check it at runtime) and the code after them usually is written in such a way that it may often crash hard if the condition is false.

So you get assert warning only in debug mode, but most likely the real problem is happening in

release too, it's just not catched early by ASSERT, but causes havoc further down the line. In some situations you may be lucky that the code after is robust enough to survive even when condition is false, and the assert is rather just information that something is seriously broken in the code, way beyond the author expectations.

I think I found the problem. In the ODBC there is a member called tmode of type "TransactionMode". TransactionMode can be NORMAL or IMPLICIT. There is also the related method SetTransactionMode to modify tmode. For some reason it was set to NORMAL and that caused the issue. Setting it to IMPLICT fixed the problem.

Unfortunately I did not found any reference to SetTransactionMode in the forum so the only way to understand the problem was inspecting the ODBC.h and ODBC.cpp code. The problem was specifically related to MSSQLSession, so nothing in the query itself was wrong and changing it would not have helped. I do not think that my code was "seriously broken in the code, way beyond the author expectations" and I do think that who wrote that piece of code could have easily given some hint about the origin of the problem.

Subject: Re: ODBC Assertion failed
Posted by mirek on Tue, 22 Dec 2020 17:22:49 GMT
View Forum Message <> Reply to Message

Giorgio wrote on Tue, 22 December 2020 11:44mr_ped wrote on Fri, 18 December 2020 14:02Giorgio wrote on Fri, 18 December 2020 08:26I've found out that the problem arises only in debug mode. The problem is the ASSERT at line 200 and 215 in ODBC/ODC.cpp. I can live without having the capability to debug that part of code, although it's very handy when there are issues (a problem with that part of the code was the reason I turned on the debugger in the first place).

Usually ASSERT compiles to nothing in release, and to debug check under debug.

So "arises only in debug mode" means that you see the warning in debug mode, but in release the same condition, which would trigger assert, does nothing and continues further.

ASSERT usually ensures conditions which "must be true" (and there is no need to check it at runtime) and the code after them usually is written in such a way that it may often crash hard if the condition is false.

So you get assert warning only in debug mode, but most likely the real problem is happening in release too, it's just not catched early by ASSERT, but causes havoc further down the line. In some situations you may be lucky that the code after is robust enough to survive even when condition is false, and the assert is rather just information that something is seriously broken in the code, way beyond the author expectations.

I think I found the problem. In the ODBC there is a member called tmode of type "TransactionMode". TransactionMode can be NORMAL or IMPLICIT. There is also the related method SetTransactionMode to modify tmode. For some reason it was set to NORMAL and that

caused the issue. Setting it to IMPLICT fixed the problem.

Unfortunately I did not found any reference to SetTransactionMode in the forum so the only way to understand the problem was inspecting the ODBC.h and ODBC.cpp code. The problem was specifically related to MSSQLSession, so nothing in the query itself was wrong and changing it would not have helped. I do not think that my code was "seriously broken in the code, way beyond the author expectations" and I do think that who wrote that piece of code could have easily given some hint about the origin of the problem.

Sorry for the late reply, only now I have checked all details.

The ASSERT should basically mean that there is mismatch between Begin and Commit/Rollback pairs.

NORMAL transaction mode means you are using Begin/Commit/Rollback. Without Begin, there is commit after each statement. Transactions can be nested (it that case, only top-level transaction is relevant).

In IMPLICIT mode, each commit or rollback starts another transaction. Begin is the same as commit, mismatch between Begin/Commit is ignored. That is why you are not seeing the problem with IMPLICIT mode...

That said, I cannot construct a scenario which would lead to failure you describe. Would it be possible to insert DDUMP(tlevel) before both ASSERTs? (and post here the result...)

Mirek

Subject: Re: ODBC Assertion failed
Posted by mirek on Tue, 22 Dec 2020 17:49:14 GMT
View Forum Message <> Reply to Message

One little issue: You are supposed to use Begin/End/Commit with your session, not query - that is deprecated. But in fact, the result should be the same (it just calls GetSession().Begin().. etc...)

For what is worth, I have altered SQL_MSSQL reference example to check the issue:

```
#include "app.h"

#include <Sql/sch_schema.h>
#include <Sql/sch_source.h>

using namespace Upp;

CONSOLE_APP_MAIN
{
```

```
 MSSQLSession mssql;
 if(!mssql.Connect("Driver={SQL Server Native Client
11.0};Server=localhost;Database=master;Trusted_Connection=Yes;")) {
  Cout() << "Connect failed: " << mssql.GetLastError() << '\n';
  return;
 }

 SQL = mssql;

 SqlSchema sch(MSSQL);
 StdStatementExecutor se(SQL.GetSession());
 All_Tables(sch);
 ODBCPerformScript(sch.Upgrade(), se);
 ODBCPerformScript(sch.Attributes(), se);

#ifdef _DEBUG
 mssql.SetTrace();
 mssql.LogErrors();
 mssql.ThrowOnError();
#endif


 SQL.Begin();
 try {
  for(int i = 0; i < 10; i++)
   SQL * Insert(TEST)(ID, i)(TEXT, String('A' + i, 1));
 }
 catch(SqlExc e)
 {
  DDUMP(e);
 }

 if(SQL.WasError()) {
  DLOG("There was ERROR!");
  SQL.Rollback();
 }
 else
  SQL.Commit();

 S_TEST tst;
 Sql sql;
 sql * Select(tst).From(TEST);
 while(sql.Fetch(tst))
  Cout() << tst.ID << ", " << tst.TEXT << '\n';
}
```

and everything works as expected...

Mirek

Subject: Re: ODBC Assertion failed
Posted by JeyCi on Wed, 23 Dec 2020 05:08:00 GMT
View Forum Message <> Reply to Message

mirek wrote on Tue, 22 December 2020 18:49One little issue: You are supposed to use Begin/End/Commit with your session, not query - that is deprecated.
thanks for corrections of my view!
then we can use "try-catch" for query/transaction & WasError()-catching for the session, as I can see... will remember :)
& thanks to Giorgio for such useful question

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Wed, 27 Jan 2021 15:44:21 GMT
View Forum Message <> Reply to Message

Hello,
I made a lot of tests in the last weeks and I still cannot wrap my head around this error.

To begn with, I had some issues with DDUMP: if I put that before the assert and I try to compile in release mode I got "error C2018: unknown character '0x40'". So I used RDUMP in release. In debug mode DDUMP is ok.

Anyway, I noticed in the log "tlevel = 0" and just after that "tlevel = -1" so inspecting closely the code I found this in the destructor of the class:

```
 if(my_mssql_db.IsOpen()){
  my_mssql_db.Commit();
  my_mssql_db.Close();
 }
```

Actually, that extra .Commit() was wrong (there was no matching between the .Begin and the .Commit/.Rollback) and after I removed it the "tlevel = -1" disappeared... but the application crashed anyway.

So I tried something different. The code I posted initially is actually a simplified version of my code. What I'm doing is inserting a header/body document, so I have a query that insert the header in a table and after that a for cycle for the body in a different table. Initially I used just one Sql object both for the header and the body, so I tried to use two different Sql objects, one for the body and one for the header:

```
bool InsertDocument(ValueMap * header, vector<vm*> * rows){

 Sql query_header(my_mssql_db);
 query_header.ClearError();

 Sql query_body(my_mssql_db);
 query_body.ClearError();

 query_header.Begin();
 query_body.Begin();

   try{
    query_heder * Insert(My_mssql_header_table)(*header);
   } catch(SqlExc) {
    ErrorOK(query_header.GetLastError());
   }

 for(vector<vm*>::iterator it = rows->begin(); it != rows->end(); ++it) {
     try{
      query_body * Insert(My_mssql_body_table)(*it);
     } catch(SqlExc) {
      ErrorOK(query_body.GetLastError());
     }
    }

 if(query_header.WasError() || query_body.WasError(){
  query_body.Rollback();
  query_header.Rollback();
  return false;
 }

 query_header.Commit();
 query_body.Commit();
    return true;
}
```

After a couple of tests it seemed to work, but in the end also this approach led to crashes. At the end I decided to scrap the Commit altogether, so each insert it's by itself. After a couple of week of usage I got no crashes. This approach can lead to some inconsistencies (e.g. an header without a body), but luckily enough this is a quite isolated part of the application, so I can manage manually this kind of situation.

What baffles me is that - AFAIK - using a transaction to bundle together a bunch of insert is perfectly fine (it's actually the best solution because in case of problem you rollback the entire set of instructions - both the insert in the header and those in the body), so I cannot understand why using "isolated" insert is working while Commit leads to crashes.

Regards,
gio

Subject: Re: ODBC Assertion failed
Posted by mirek on Wed, 27 Jan 2021 16:08:43 GMT
View Forum Message <> Reply to Message

Giorgio wrote on Wed, 27 January 2021 16:44Hello,
I made a lot of tests in the last weeks and I still cannot wrap my head around this error.

To begn with, I had some issues with DDUMP: if I put that before the assert and I try to compile in release mode I got "error C2018: unknown character '0x40'". So I used RDUMP in release. In debug mode DDUMP is ok.

That is a correct behaviour. See, the reason for DDUMP is that ppl often forget debugging logs in the code. So if it does not compile in release, you are forced to remove them...

```
bool InsertDocument(ValueMap * header, vector<vm*> * rows){

 Sql query_header(my_mssql_db);
 query_header.ClearError();

 Sql query_body(my_mssql_db);
 query_body.ClearError();

 query_header.Begin();
 query_body.Begin();
```

This is really quite ugly code: Transaction is associated with SqlSession, so the second begin starts the second level.

Mirek

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Thu, 28 Jan 2021 08:26:32 GMT
View Forum Message <> Reply to Message

mirek wrote on Wed, 27 January 2021 17:08Giorgio wrote on Wed, 27 January 2021 16:44Hello,

```
bool InsertDocument(ValueMap * header, vector<vm*> * rows){

 Sql query_header(my_mssql_db);
 query_header.ClearError();

 Sql query_body(my_mssql_db);
 query_body.ClearError();

 query_header.Begin();
 query_body.Begin();
```

This is really quite ugly code: Transaction is associated with SqlSession, so the second begin starts the second level.

I agree :roll: it was one desperate attempt, at the end I removed the "double" session...

Subject: Re: ODBC Assertion failed
Posted by mirek on Thu, 28 Jan 2021 08:38:38 GMT
View Forum Message <> Reply to Message

Any chance for a testcase? I would not want to let this go...

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Fri, 12 Feb 2021 17:15:57 GMT
View Forum Message <> Reply to Message

I will prepare indeed a testcase, I have just to find some spare time in this crazy time of the pandemic.

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Fri, 21 May 2021 09:04:01 GMT
View Forum Message <> Reply to Message

Hello,
I'm preparing the testcase, it's harder than I thought as I have to strip down a quite big software, I'll upload the testcase next week.
Regards,
gio

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Mon, 24 May 2021 12:54:26 GMT
View Forum Message <> Reply to Message

Hi there,
so, here's my test case.

The basic functioning of the test case is the following: an object of the class Order is created (there is a in the test case class that creates a fake order, in my application the order is entered by the user through a UI), the order is then converted in a ValueMap and finally is inserted in the DB using SqlIds and other facilities provided by U++. I use this database connection to integrate my application with an accounting software. This accounting software provides a special table of tye database to allow other software to insert data in it (i.e. I'm using the table of the accounting software for their designed purpose).

I'll try now to go through each class to explain the goal of each one.

The class Order (alongside with the class OrderRow) represents a customer order: it has the classic header/body structure.

The file TesDbSql.h contains definitions of the SqlIds used by Upp's Sql class (for insert etc.).

The class TestDAO contains the actual functions that save data into the database.

The class TestDocumentMapping transform an Order object into a ValueMap, so it can be easily processed by the DAO.

The CreateOrder class creates a fake order.

The util class contains some functions that I use to convert the Order object into a ValueMap.

Finally the Db.sql file contains the script used to create the test database.

The crash happens only when the application is compiled, not in debug mode. With the real ms sql database the behaviour is really "consistent": I can insert one order without problems, but, if I try to insert a second one immediately after I inserted the first, the application crashes. With the database included in my test case the behaviour is much more erratic. I noticed however that there are some patterns: if I enter some orders with the application, exits the application, delete the content of the tables in the db and then launch the application and insert an order again, the application usually crashes. It usually crashes even if I insert some orders, exit the application, launch it again and insert some new orders.

If you have additional question let me know.
Regards,
gio

File Attachments

1) mssqltestcase.zip, downloaded 217 times

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Tue, 25 May 2021 15:16:26 GMT
View Forum Message <> Reply to Message

Hi,
I created the test case here.
Regards,
gio

Subject: Re: ODBC Assertion failed
Posted by mirek on Tue, 08 Jun 2021 08:08:31 GMT
View Forum Message <> Reply to Message

Giorgio wrote on Tue, 25 May 2021 17:16Hi,
I created the test case here.
Regards,
gio

You seem to set

test_db.ThrowOnError();

but nowwhere I can see any catches for Sql exception. As provided, it tends to err on duplicate keys, which without catches to Sql errors leads to aborting the app. Can that be a reason?

After commenting out above line, all works as expected, no crashes....

(OK, I think there will be more, but this where I got for now testing your testcase...)

Mirek

Subject: Re: ODBC Assertion failed
Posted by Giorgio on Thu, 10 Jun 2021 15:00:33 GMT
View Forum Message <> Reply to Message

Hi there,
I finally found the culprit: that was a very tricky one.

So, in my code, I had the following in the connection string:

"Driver={SQL Server}"

I used that after checking the driver available in my system using ODBC Data sources:

For some reason that driver is not working (AFAIK because it's a 32 bit and I compile my application in 64 bit). Using the driver {ODBC Driver 17 for SQL Server} everything works fine. Anyway the application was able to insert the data into the DB before crashing, so it was really hard to tell that the problem was the driver. I used Upp with Mysql, Postgres and SqlLite and never had such a problem, so I blame Microsoft.

Thanks for the support!
gio
