Subject: Oracle RefCursor and Stored Procedure
Posted by aksdb on Thu, 17 Apr 2014 16:59:19 GMT
View Forum Message <> Reply to Message

I'm currently evaluating U++, which at a first glance seems pretty awesome. I usually detest C/C++, but U++ showed me that it can indeed look fantastic

Anyway, our Oracle (11) database at work is handled strictly through stored procedures. For each table there are automatically generated SPs to insert, update and select as well as manually crafted SPs for special tasks. Each function returns a custom struct (containing error code and message), while the actual "results" of the query are in a refcursor as out-parameter.

From what I can see, OCI8Connection provides a .SetParam(int, Sql&) which seems to be intended for exactly such ref-cursors. However I wasn't able to find out how I can make use of it, since the SQL class doesn't give me any way to access that overloaded function.

So: any chance to work in this environment with current U++ SQL?

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by mirek on Tue, 22 Apr 2014 06:14:27 GMT
View Forum Message <> Reply to Message

I cannot say I have used it ever, but from what I see in the code, the trick is to use "raw" Execute and pass a pointer to Sql using RawValue.

Something like (sorry for the lack of knowledge of Oracle syntax):

```
Sql out;
SQL.Execute("call x(?)", RawToValue(&out));
```

Mirek

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by aksdb on Tue, 22 Apr 2014 08:03:15 GMT
View Forum Message <> Reply to Message

Close, it actually works without the RawToValue I wasn't aware that a typed pointer would be wrapped into Value and correctly identified later on.
So at least I got the call now. I just need to figure out, why Oracle doesn't like the refcursor/result set and throws me an
PLS-00382: expression is of wrong type

It's probably because we use "custom" ref cursors

```
TYPE ref_cursor_type IS
    REF CURSOR RETURN ref_cursor_rec;
```

(where ref_cursor_rec is a custom structure containing the fields to be returned).

I didn't expect it to be that much of a hassle. But I'll hopefully figure this out. Thanks for that pointer (literally)!

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by aksdb on Tue, 22 Apr 2014 09:26:29 GMT
View Forum Message <> Reply to Message

I was wrong: RawToValue is necessary. However something is wrong there.

I added some debug output to "void OCI8Connection::SetParam(int i, const Value& q)":
Cout() << Format("Type is 0x%X (%s)", (int)q.GetType(), q.GetTypeName()) << EOL;

This results in:
Type is 0x8000000 (N3Upp11RawValueRepIPNS_3SqlEEE)

Obviously, this means that type is != UNKNOWN_V, so we run into the assertion "NEVER();".

Just for the sake of testing it, I moved the relevant check (if(IsTypeRaw<Sql *>(q)) ...) into the default case. This will at least make the Oracle call succeed and I also get a result set I can iterate on, however when the application closes, I get a report about a heap corruption:
Heap is corrupted --memory-breakpoint__ 1189

Something's very wrong here, I'm afraid.

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by mirek on Tue, 22 Apr 2014 12:28:01 GMT
View Forum Message <> Reply to Message

aksdb wrote on Tue, 22 April 2014 09:26I was wrong: RawToValue is necessary. However something is wrong there.

I added some debug output to "void OCI8Connection::SetParam(int i, const Value& q)":
Cout() << Format("Type is 0x%X (%s)", (int)q.GetType(), q.GetTypeName()) << EOL;

This results in:
Type is 0x8000000 (N3Upp11RawValueRepIPNS_3SqlEEE)

Obviously, this means that type is != UNKNOWN_V, so we run into the assertion "NEVER();".

Just for the sake of testing it, I moved the relevant check (if(IsTypeRaw<Sql *>(q)) ...) into the default case. This will at least make the Oracle call succeed and I also get a result set I can iterate on, however when the application closes, I get a report about a heap corruption:
Heap is corrupted --memory-breakpoint__ 1189

Something's very wrong here, I'm afraid.

Removing case UNKNOWN_V is absolutely correct; it looks like nobody ever used this code snippet for a very long time.
Personally, I am only using Oci7 part of Oracle package. Change is now commited.

It is also possible that the heap corruption was there since the day one, only that it now gets detected:)

However, is not it possible that the heap corruption is somehow caused by your app? If it is simple test, can you post it here?

Mirek

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by mirek on Tue, 22 Apr 2014 12:31:45 GMT
View Forum Message <> Reply to Message

PS:

--memory-breakpoint__ 1189

If "1189" is stable between runs, it could be easy to find the error. Just put the whole thing into commandline; U++ will crash at the allocation of block that will get corrupted later. Catch in debugger and examine the backtrace...

Mirek

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by aksdb on Tue, 22 Apr 2014 12:53:33 GMT
View Forum Message <> Reply to Message

It's already a quite minimal example. I anonymized it to show you what I currently do:
#include <Core/Core.h>
#include <Sql/Sql.h>
#include <Oracle/Oracle8.h>

using namespace Upp;

void runQuery()

```
{
 StringBuffer sb;

 sb << "DECLARE \n";
 sb << "  res SOME_CUSTOM_TYPE; \n";
 sb << "BEGIN \n";
 sb << "  res := SCHEMA.PACKAGE.FUNCTION(OutCursor=>?); \n";
 sb << "END; \n";

 Sql ref;
 if (SQL.Execute(sb, RawToValue(&ref)))
 {
  Cout() << "Call succeeded" << EOL;

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

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

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

 String sUser("USER");
 String sPassword("PASS");
 String sDB("DB");

 String sConnStr;
 sConnStr.Cat() << sUser << "/" << sPassword << "@" << sDB;

 Oracle8 oraSession;
 if (!oraSession.Open(sConnStr, false)) {
  Cout() << "Cannot open DB connection." << EOL;
  Cout() << oraSession.GetLastError() << EOL;
  return;
 }

 SQL = oraSession;
```

```
 runQuery();

 Cout() << "Done" << EOL;
}
```
I also attach the backtrace (I'm already looking into it, but at least to me it's not obvious (yet) what goes wrong here.):
```
#0  0x000000000048f308 in Upp::Panic (
    msg=0x7fffffffe0d0 "Heap is corrupted --memory-breakpoint__ 1168 ")
    at /home/aksdb/upp/uppsrc/Core/Util.cpp:119
#1  0x000000000046b2e7 in Upp::HeapPanic (
    text=0x7fffffffe0d0 "Heap is corrupted --memory-breakpoint__ 1168 ",
    pos=0x7ffff7ff6060, size=-1)
    at /home/aksdb/upp/uppsrc/Core/heaputil.cpp:202
#2  0x000000000046d87c in Upp::DbgHeapPanic (
    text=0x59290a "Heap is corrupted ", p=0x7ffff7ff6040)
    at /home/aksdb/upp/uppsrc/Core/heapdbg.cpp:53
#3  0x000000000046da96 in Upp::MemoryFree (ptr=0x7ffff7ff6060)
    at /home/aksdb/upp/uppsrc/Core/heapdbg.cpp:130
#4  0x0000000000445b5c in operator delete[] (ptr=0x7ffff7ff6060)
    at /home/aksdb/upp/uppsrc/Core/Core.h:255
#5  0x00000000004b2927 in Upp::OCI8Connection::Item::Clear (
    this=0x7ffff7fdd1c0) at /home/aksdb/upp/uppsrc/Oracle/Oci8.cpp:186
#6  0x00000000004b2a1f in Upp::OCI8Connection::Item::~Item (
    this=0x7ffff7fdd1c0, __in_chrg=<optimized out>)
    at /home/aksdb/upp/uppsrc/Oracle/Oci8.cpp:208
#7  0x00000000004bdebb in Upp::Array<Upp::OCI8Connection::Item>::Free (
    this=0x7ffff7fdf0e0) at /home/aksdb/upp/uppsrc/Core/Vcont.hpp:388
#8  0x00000000004bd6c9 in Upp::Array<Upp::OCI8Connection::Item>::~Array (
    this=0x7ffff7fdf0e0, __in_chrg=<optimized out>)
    at /home/aksdb/upp/uppsrc/Core/Vcont.h:263
#9  0x00000000004b8cdc in Upp::OCI8Connection::~OCI8Connection (
    this=0x7ffff7fdf060, __in_chrg=<optimized out>)
    at /home/aksdb/upp/uppsrc/Oracle/Oci8.cpp:1214
#10 0x00000000004b8dbe in Upp::OCI8Connection::~OCI8Connection (
    this=0x7ffff7fdf060, __in_chrg=<optimized out>)
    at /home/aksdb/upp/uppsrc/Oracle/Oci8.cpp:1217
#11 0x0000000000588827 in Upp::Sql::Detach (this=0x7ffff7ff8160)
    at /home/aksdb/upp/uppsrc/Sql/Sql.cpp:676
#12 0x0000000000588887 in Upp::Sql::~Sql (this=0x7ffff7ff8160,
    __in_chrg=<optimized out>) at /home/aksdb/upp/uppsrc/Sql/Sql.cpp:688
#13 0x00000000004f4eb4 in Upp::One<Upp::Sql>::Free (this=0x7fffffffe598)
    at /home/aksdb/upp/uppsrc/Core/Other.h:26
#14 0x00000000004f44aa in Upp::One<Upp::Sql>::Clear (this=0x7fffffffe598)
    at /home/aksdb/upp/uppsrc/Core/Other.h:35
#15 0x00000000004ea432 in Upp::SqlSession::SessionClose (this=0x7fffffffe4d0)
    at /home/aksdb/upp/uppsrc/Sql/Session.cpp:90
#16 0x00000000004ba3d0 in Upp::Oracle8::Logoff (this=0x7fffffffe4d0)
```

```
    at /home/aksdb/upp/uppsrc/Oracle/Oci8.cpp:1343
#17 0x00000000004baf36 in Upp::Oracle8::~Oracle8 (this=0x7fffffffe4d0,
    __in_chrg=<optimized out>)
    at /home/aksdb/upp/uppsrc/Oracle/Oci8.cpp:1455
#18 0x0000000000404680 in ConsoleMainFn_ ()
    at /home/aksdb/MyApps/SQLTest/SQLTest.cpp:59
#19 0x0000000000485957 in Upp::AppExecute__ (app=0x404353 <ConsoleMainFn_()>)
    at /home/aksdb/upp/uppsrc/Core/App.cpp:322
#20 0x0000000000404347 in main (argc=1, argv=0x7fffffffe7f8,
    envptr=0x7fffffffe808) at /home/aksdb/MyApps/SQLTest/SQLTest.cpp:37
```

I'll debug this further in hopes to find out, what reference is corrupt exactly (I presume some of the special refcursor handling isn't up-to-date with other SQL backend changes ... but we'll see)

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by aksdb on Tue, 22 Apr 2014 14:27:03 GMT
View Forum Message <> Reply to Message

Ok, one huge problem apparently are these comparisons:total_len > sizeof(buffer)
Since total_len is an int (and therefore signed), the behaviour is rather strange. I actually would have expected the compiler to prefer the type of the left side, but apparently it doesn't. So now we (sometimes) compare -1 > 8u which will then return true (since -1 will be cast into uint).

Replacing these four occurances with
total_len > (int)sizeof(buffer)
solves at least this particular problem. There is still some more memory corruption (since the next query fails at GetColumnInfo which, if executed alone, works fine). So I keep hunting ...

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by mirek on Wed, 23 Apr 2014 06:27:46 GMT
View Forum Message <> Reply to Message

aksdb wrote on Tue, 22 April 2014 14:27Ok, one huge problem apparently are these comparisons:total_len > sizeof(buffer)
Since total_len is an int (and therefore signed), the behaviour is rather strange. I actually would have expected the compiler to prefer the type of the left side, but apparently it doesn't. So now we (sometimes) compare -1 > 8u which will then return true (since -1 will be cast into uint).

Replacing these four occurances with
total_len > (int)sizeof(buffer)
solves at least this particular problem. There is still some more memory corruption (since the next query fails at GetColumnInfo which, if executed alone, works fine). So I keep hunting ...

Thanks! It is quite apparent, now knowing it

Hopefully I will make my Oracle working this evening, so I will be able to fix remaining issues in Oci8.

I apologize for problems, it really is half-forgoten piece of code. I guess most of us prefer open-source databases these days...

Mirek

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by mirek on Wed, 23 Apr 2014 06:32:33 GMT
View Forum Message <> Reply to Message

I have spotted and fixed two more possible instances of this problem in Oci8 - it is now in the trunk. Perhaps it will help...

Mirek

Subject: Re: Oracle RefCursor and Stored Procedure
Posted by aksdb on Wed, 23 Apr 2014 08:22:22 GMT
View Forum Message <> Reply to Message

There is absolutely no need to appologize. I would not use Oracle if I had a choice, but I intend to develop an administration tool for our existing system, and that uses Oracle so I have to live with it

I also found a workaround for the second crash.

Part of the backtrace was Execute -> GetColumnInfo -> Vector<SqlColumnInfo>::Clear -> SqlColumnInfo::Free.
So I tried using separate Sql objects for each query. (Before I used the global "SQL" for everything)
Now that crash doesn't happen.

I consider it fixed Thanks!

(If I have time, I'll also try to find why using the global SQL object causes a problem in this case.)
