

RE: DBI-1.48 bind_param_inout produces ORA-01461 colliding with a CLOB

Source: <http://coding.derkeiler.com/Archive/Perl/perl.dbi.users/2005-07/msg00098.html>

- *From:* Ron.Reidy@xxxxxxxxxxxxxxxxxxxxxx (Ron Reidy)
 - *Date:* Wed, 13 Jul 2005 10:52:33 -0600
-

Steve,

See the DBD::Oracle docs, section "Handling LOBs". Basically, you need to insert a `EMPTY_CLOB()`, return the locator, and then call `ora_write_lob()` to push the data into the locator.

Ron Reidy
Lead DBA
Array BioPharma, Inc.

-----Original Message-----

From: Steven Lembark [<mailto:lembark@xxxxxxxxxxxxx>]
Sent: Wednesday, July 13, 2005 11:00 AM
To: dbi-users@xxxxxxxx
Subject: DBI-1.48 bind_param_inout produces ORA-01461 colliding with a CLOB

Summary

=====

perl-5.8.7, DBI-1.48, Oracle-9.2.0.4.

This is probably an Oracle question in the long run: How to frame a query so that `bind_param_inout` can pass strings as `varchar's` and have them inserted into a clob. It'd be nicer if there were a way to handle this in Perl however...

In theory I could write something to query the data dictionary for LOB fields and adjust the binding type used with `bind_param_inout`; in reality I have a short time to finish this and need a quicker fix.

The default handling of `varchar` bumping into a CLOB w/in Oracle is probably what is killing me...

RE: DBI-1.48 bind_param_inout produces ORA-01461 colliding with a CLOB

RE: DBI-1.48 bind_param_inout produces ORA-01461 colliding with a CLOB

This code has worked for any number of tables,
so the CLOB issue is probably what's biting me.

Any help would be appreciated.

=====
Detail
=====

Google on qw(perl DBI bind_param_inout ORA-01461)
in various combinations gets me nowhere. The POD
for bind_param_inout references bind_parm, which
notes that:

Perl only has string and number scalar data types. All
database types that aren't numbers are bound as
strings and must be in a format the database will
understand except where the bind_param() TYPE
attribute specifies a type that implies a particular
format. For example, given:

```
$sth->bind_param(1, $value, SQL_DATETIME);
```

the driver should expect \$value to be in the ODBC
standard SQL_DATETIME format, which is 'YYYY-MM-DD
HH:MM:SS'. Similarly for SQL_DATE, SQL_TIME etc.

As an alternative to specifying the data type in the
"bind_param" call, you can let the driver pass the
value as the default type ("VARCHAR"). You can then
use an SQL function to convert the type within the
statement. For example:

```
INSERT INTO price(code, price) VALUES (?, CONVERT(MONEY,?))
```

The "CONVERT" function used here is just an example.
The actual function and sy