

Re: Standard DBI Proposal

Source: <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2007-10/msg01276.html>

- *From:* tunity5@xxxxxxxxxx
 - *Date:* Wed, 31 Oct 2007 05:31:08 -0700
-

Kevin Kenny wrote:

I don't quite understand what you're asking for here.

If you're asserting that there is no ODBC on non-Windows platforms, unixODBC (<http://www.unixodbc.org/>) appears to be both popular and well-maintained. There was a release shipped just a couple of weeks ago. I can't comment on how well recent versions work. I haven't tried it in several years.

I think you are changing the subject a little bit. Can I load unixODBC, iodbc or other drivers on Linux/Unix from Tcl? Is there a Tcl command I can use to connect, etc.? I don't think so. (I would be glad to hear I am mistaken.) This is what tclodbc does and I think everybody on this thread understands its value.

If you're asserting that Tcl entirely lacks an interface to ODBC on non-Windows platforms, I know that tclodbc has been built on Unix hosts. Again, I haven't tried it there in quite some time. (I tend to regard it as a Windows library.) When I have used it, I've found Roy Nurmi, its manager, to be quite responsive to bug reports. It's mature enough on Windows that I haven't needed to report any bugs in several years. For what I use it for, it just works. If the Unix implementation is less stable, then you've a legitimate concern -- but please take it up in a separate thread, because it's a bit off-topic for this one.

He replied to you perhaps because he knows you? Some from our company tried to contact him several years ago. There was no response from him, which is a more common experience based on similar posts on this forum.

As I said in my email, I did try to build it on non-Windows systems. Again, it was possible a few years ago on Linux. My experience with it says it is quite an accomplishment if you can get it to compile

successfully today, if at all.

I owe you an explanation of why I haven't tried ODBC on Unix for quite some time. I have always found ODBC, even on its native Windows, to be slow, and often have stability issues with it as well. I use it to connect to Jet databases and SQLServer, for which no really good alternatives are available, but I don't use them much. My two workhorse databases are Oracle (for when only Big Iron will do) – which has very well supported native drivers – and SQLite. I use the latter because of its "zero configuration" properties. It runs in-process without a separate server, manages its databases simply as files in the file system, and is simply "drop in and go." Having to deal with configuring ODBC data sources, installing *three* separate ODBC layers on Unix (ODBC itself, an ODBC driver for the database, and a Tcl-ODBC layer) would totally destroy that advantage. ODBC therefore loses out on grounds of both convenience and performance.

I have used tclodbc on Linux with Oracle, DB2, PostgreSQL and a few other databases, and even more databases on Windows. I did not find it to be slow, or have stability issues. If you did, I would guess it was more due to the app architecture than a simple API to talk to the database.

In any case, ODBC comes with Windows. It should be similar on Linux. With modern package managers, it is very simple. And, you only need to do it once.

Most of the discussion in this thread proposes the same for Tcl: a specification (and possible interface glue) so that all the databases present a common-denominator API. That would give you what you ask for: a uniform way to move your Tcl code from one database to another.

I disagree. It is about the availability of the glue that tclodbc provides. It provides a nice interface. The additional discussion has been to build on top of it at a higher level. You may agree or disagree with the goals of such a project; but the underlying foundation was a tool like tclodbc or snodbc.

Re: Standard DBI Proposal

Bear in mind, though, that we do need this discussion. We do have to get the interface right. I already see a number of key ideas in this thread:

- resistance to SQL insertion attacks has to be designed in and can be helped by integrating the SQL query tightly to the Tcl layer. SQLite does quite a good job of this. TclODBC has a more awkward interface, but at least gets the job done with prepared statements. MysqLtcl falls out of bed; the idea of providing a separate "magic quoting" operation to scrub text for insertion into a query is a non-starter.

The only platform that suffers from magic quoting is MySQL (and magnified with its use with PHP). I have worked with literally with more dozen databases and have never had a problem with it after dealing with it once.

- proper handling of NULLs is essential. This idea has a natural tension against Tcl; in Tcl, "everything is a string," but the NULL value is not any string. This has been a problem in the past; TIP 185 proposes one solution (ill-advised in my personal opinion). I'm convinced that using dictionaries to represent rows in result sets is a far better approach. I'll post more on this in a separate discussion, probably in the next few days.

I believe you can set a value of your choosing to to return instead of NULL's. Same thing is available in oratcl too.

- The interface must support result sets that are too large to fit in memory. Some of us do slog through multiple gigabytes of data on the client side. Sometimes there's just no alternative.

Ahh! Now I see why you have mentioned ODBC performance a few times. You are transferring large amounts of data back and forth between the database server and the client. And it would seem that ODBC was not performing because of all the data transfers back and forth.

I do not think so. You will never have enough memory for your data.

Re: Standard DBI Proposal

This is one thing I do not like about Sqlite interface. If you are retrieving plain data rows from the server and doing all the work on the client (joins, merges, etc.), please read up on databases and SQL. Or buy our products :-). Otherwise, you are re-implementing what a database server does.

– It is highly desirable for result sets to support external iterators ("cursors") as well as internal iterators (looping constructs). Once again, some of us maintain large federated databases, and sometimes there's no alternative to doing a merge-join on the client side that involves two or more multi-gigabyte result sets. Life is hard.

Again, this is database server responsibility; and most do provide forward cursors at a minimum. ODBC and tclodbc support prepared statements that handle it quite nicely.

Expect more on this in the coming weeks. I'm rather under a lot of time pressure lately, so my replies are likely to have sporadic and unpredictable delays, but I am continuing to examine these issues.

If it can let me connect to databases on non-Windows systems, I am really looking forward to it. I hope it provides at least the same API that tclodbc does without getting in the way too much.

--

73 de ke9tv/2, Kevin