

Re: Standard DBI Proposal

Source: <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2007-10/msg01246.html>

- *From:* Kevin Kenny <kennykb@xxxxxxx>
 - *Date:* Tue, 30 Oct 2007 10:38:53 -0400
-

tunity5@xxxxxxxxx wrote:

I have followed this thread with a slight interest. As with the original poster, I find the lack of an ODBC driver on non-Windows platforms a bit of a concern. The thread, however, seems to have evolved into something else: an SQL layer over just such an ODBC driver.

Am I missing something? Why not work on a driver first and the simplification layer later? While we are at that, what is wrong with the ODBC interface? (It is not OO is not a valid argument.) ODBC has worked for ages and is universally available. Further, any attempt to hide/objectify/replicate the full SQL standards seems besides the point; especially without also providing the capacity a driver provides to connect to databases.

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.

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.

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

Re: Standard DBI Proposal

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.

Given such constraints, we will be living with multiple database connectivity API's for some time to come. Our colleagues in the Java community have faced the same problem, and arrived at the conclusion that they needed an overarching interface — JDBC — that subsumes multiple database APIs; hence, JDBC can connect to ODBC, to Oracle's native API, to Postgres95's native API, and so on. It winds up being incumbent upon the database vendor, or an aftermarket developer, to supply the bridge that adapts JDBC to whatever API the underlying database presents.

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 will admit that the discussions in this thread have been straying somewhat afield from that goal. I'd ascribe that to posters floating "blue sky" design ideas – attempting to improve on interfaces like ODBC and JDBC by making them "more object-oriented" or whatever. Many of the people who are likely to do the actual work, however, are reading these discussions with some interest, mentally filing the wilder ideas under "interesting idea, might be nice to have in a later release," and thinking more about the job at hand. Those individuals also don't post very much – they're too busy working on the problem.

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,

Re: Standard DBI Proposal

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.

– 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.

– 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.

– 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.

As far as I can tell, none of the existing Tcl database API's gets all of these completely right. But (unlike object orientation or XML-ification or Rails-style web interface generation), these cannot be worked around by adding another layer on top of the database interfaces. In my view, these points, and likely a handful of others, are fundamental to any database interface.

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.

—

73 de ke9tv/2, Kevin

.