

TIP #308: Twylite's concerns

Source: <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2007-11/msg00705.html>

- *From:* Kevin Kenny <kennykb@xxxxxxx>
 - *Date:* Sat, 17 Nov 2007 15:00:39 -0500
-

Twylite wrote:

> So here is a list of specific concerns that arise from this
> exercise:
>
> (1) The DBI is complex. The comparative lengths of my example
> functions illustrate that clearly. If ::tcl::db::execute had a
> mechanism to report the ordered list of columns returned this would
> make "simple" development a lot easier (comparable to SQLite). This
> change alone would also void my comments about error handling (at
> least to some degree).

OK, so we start needing like [`$db execute $sql -columns colsVar etc...`]
That's easy enough to add; I can throw that in for the next round.

> (2) The amount of error handling this DBI requires is significant –
> 4 nested error handlers. Other DBIs require 1 to 3. This level of
> error handling detracts from my task of writing in functionality
> makes quality assurance more difficult.

If, as you say, the simple change of adding a place for [`$db execute`]
to return the column list fixes this, I'm not going to address it
for now. Feel free to follow up if you have further issues.

> (3) Taking variables from the current stack frame is a recipe for
> trouble, and makes writing logic like `extract_report` more complex
> and slower (best case: [`dict with ...`]; worst case: `foreach {name
> value} { set ... }`) I would prefer to see `execute` take a `dict`; then
> it is at least clear what variables you are (and are not) providing.
> Taking variables from the stack without clearly indicating them `_in`
> `Tcl code_` (as opposed to another language like SQL) is a little to
> much of a DWIM for my liking.

The latest edits to TIP #308 allow for both versions. They are
both needed: fixed transactional queries like

```
select balance from accounts where account_number = :acct
```

really benefit from not having to fabricate a dictionary with the

TIP #308: Twylite's concerns

substituents, while, as you observe, ad-hoc queries, often built on the fly, need the extra isolation that a dictionary provides.

- > (4) Returning the columns as a resultset is a real pain. Rather make
- > the columns available as
- > 4.a) a list of tuples; or
- > 4.b) a list of column names, plus a function to get a dict of
- > information about the column (given its name)

I'll think about this one a little bit. You're right that it's most natural to the user to have it as list-of-tuples, and that's easy to provide. It also seems more natural to the database implementor to