

Re: Ada DB bindings and APQ

Source: <http://coding.derkeiler.com/Archive/Ada/comp.lang.ada/2004-12/0307.html>

From: Warren W. Gay VE3WWG (ve3wwg_at_NoSpam.cogeco.ca)

Date: 12/16/04

Date: Wed, 15 Dec 2004 22:10:49 -0500

Brian May wrote:

```
>>>>>>"Brian" == Brian May <bam@snoopy.apana.org.au> writes:
>
>
> Brian> The documentation has an alternative loop structure:
>
> Brian> while not End_Of_Query(Q) loop Fetch(Q); ... end loop;
>
> I realized, despite the documentation, End_Of_Query is not currently
> supported on sequential connections, because it is not known if all
> tuples have been fetched until you try to fetch the next one past the
> end.
```

Actually for PostgreSQL, this works fine. But for MySQL (at least for versions that I worked with), it was "MySQL busted" (this is documented in the APQ 2.2 manual). MySQL's library would first indicate that it is "not at end", and then when you went to fetch a row, it would realize "whoopsie, we are at the end of rows".

Then when I got to Sybase (IIRC), you cannot do this kind of test at all in sequential mode.

```
> This limitation could be overcome if the previous call to the
> "execute" or "fetch" called fetch in advance for the next row, and
> stored the results in a temporary holding point. You don't miss out on
> anything either, as all the rows will eventually have to be fetched
> anyway.
```

Prefetching introduces much ugliness in the Query_Type object, because you change the state of things. This can be especially complicated for Sybase, because the Sybase is very fussy about what happens on the connection and its state(s). Based upon what I've seen so far, I would not recommend any kind of prefetching.

- > *This is better, IMHO, then requiring an exception be the terminating*
- > *condition for a loop.*

I'll agree that I am not completely happy with the way that Fetch raises the exception, but I felt that was better than the possibility of ignoring a failure (the C tendency). That way if you code a SELECT for example that should only return 1 row, but you get zero rows for some reason, a Fetch would raise an exception (it cannot be ignored without going out of your way!)

The other consideration is that normally many rows are fetched, and there is only one "End". In this scenario, I felt that the exception was an acceptable compromise.

- > *An alternative would be to restructure the loop as:*
- >
- > *while true loop*
- > *Fetch(Q);*
- > *exit if No_More_Data(Q);*
- >
- > *...*
- > *end loop;*
- >
- > *I don't particular like this approach though, although it would work.*

I like the neatness of this approach, but the danger is that you might do the "..." part on a row that might not exist. Though I suppose the Query_Type object could maintain enough state such that it can raise an exception if any "Values" were fetched from the row that is not there. Just a quibble: I'd prefer something like "No_Row(Q)" perhaps.

An even neater loop is this, IMHO ;-)

- > *loop*
- > *Fetch(Q);*
- > *exit when No_More_Data(Q);*
- >
- > *...*
- > *end loop;*

--
Warren W. Gay VE3WWG
<http://home.cogeco.ca/~ve3wwg>