

Re: Opinions on approach, please...

Re: Opinions on approach, please...

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2008-05/msg00549.html>

- *From:* Robert <no@xxxxxx>
 - *Date:* Tue, 27 May 2008 08:20:29 -0500
-

On Tue, 27 May 2008 23:29:35 +1200, "Pete Dashwood" <dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote:

"Robert" <no@xxxxxx> wrote in message
<news:dbrm34h6fnvtl296cbteg5d6j76e3cl3a0@xxxxxxxxxx>

On Tue, 27 May 2008 11:45:57 +1200, "Pete Dashwood"
<dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
wrote:

"Robert" <no@xxxxxx> wrote in message
<news:5nv134pdmegh0jld4fnl3t0e80phj5k5an@xxxxxxxxxx>

On Tue, 27 May 2008 01:47:26 +1200,
"Pete Dashwood"
<dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxx>
wrote:

I can detect which key field
is being used, I can detect
the relation, I
can
get the data for the
condition, and I can
probably handle up to 3
connected
conditions, but I still don't
have a clear idea of how I'll
get this into
MOST... :-)

Write the where clause manually, one time.

Re: Opinions on approach, please...

Do you mean in the Application?

In the data layer. Create a data layer program per table, containing all the SQL for that table.

That is exactly what I'm doing. It cannot contain manually written WHERE clauses because these are application dependent. I am looking for it to have a GENERAL WHERE clause structure which it can populate with data from the interface and EXECUTE. It is tricky, but I don't think it is impossible. There may have been some misunderstanding here.

It's not that simple. The statement will say

```
EXECUTE :handle USING :host-variable1, host-variable2, ...
```

The number of host variable pointers, which is fixed at compilation time, must equal the number of bind variable tokens in your dynamic SQL.

That's why I recommended doing it once to see what's involved.

There IS a way to modify the bind variable list at execution time. It is the DESCRIBE verb, which is in guru territory. Your frustration trigger will be pulled when you discover DESCRIBE changes the number of arguments and their names, but not their types. You'll wind up converting everything to and from strings (varchars) or much worse, embedding data in the SQL 'just to get it working'.

The practical solution is a separate entry point (method) for each SQL statement. Usually, the only difference is the hard coded WHERE clause.

Yes, the layer itself has to be written in COBOL, but it will be a COM server. As such it can be activated from ANY language. It would be embedded in application COBOL if I allowed SQL in the application programs.

That's why I don't want SQL in the application code. Just an invocation of the COM component.

That's the classical three layer client/server model.

Don't assume the data layer must execute on the same machine as the application. It is semi-common for the invocation to send a message through middleware.

Re: Opinions on approach, please...

Re: Opinions on approach, please...

The where clause on UPDATEs is simple if you get in the habit of always fetching a ROWID.

It's faster too. Same for IN followed by a subselect.

Is ROWID a feature of the RDBMS or is it a column you have defined? Not familiar with this.

It's a database feature. ROWID is your friend.

Have you ever tried putting host variables into dynamic SQL?

No, I don't think so. I have limited experience with dynamic SQL in general, but I am learning rapidly... :-)

If not, do it once to see what's involved before you go down that road. It's a pain and it is error-prone.

I'll take your word for it. What alternative would you propose?

Hard code WHERE clauses.

Convert the ISAM IO to data layer SQL, then manually tailor where clauses.

If I do that I'd need to tailor manually every single COM server, and change it manually for every application program that uses that particular ISAM file, before inserting invokes into that particular application. There has to be a better way. If the COM server is generalised to handle whatever the application program requires, then I don't need to do that. The only part that is problematic, is the conditions on things like singletons and START. SQL provides the capability to build a query at run time and execute it. I believe the application can pass enough information through the interface to allow the COM server to do that.

If you go with dynamic SQL, you'd better master SQL debugging tools, because you'll be spending a lot of time using them. A handy one that's often overlooked is ODBC tracing.

By all means, get a copy of TOAD.

.

Re: Opinions on approach, please...