

Re: Opinions on approach, please...

Source: <http://coding.derkeiler.com/Archive/Cobol/comp.lang.cobol/2008-05/msg00499.html>

- *From:* "William M. Klein" <wmklein@xxxxxxxxxxxxxxxxxxxxx>
 - *Date:* Sun, 25 May 2008 03:15:14 GMT
-

Pete,

I know little enough about most of what you are asking about to be of almost no use. The one question that I would ask is whether you are "converting" from a COBOL that supports both forward and backwards "browsing" of ISAM, i.e.

START LESS THAN

and

READ PREVIOUS

If so, does the approach you are consider handle this?

The other question is what your converted code will do with "file-sharing" in existing COBOL (that conforms to the old X/Open specification, e.g. Micro Focus). Traditionally, I think this played a significant part in programs that kept multiple indexes "in use" against the same file.

--

Bill Klein

[wmklein <at> ix.netcom.com](mailto:wmklein@ix.netcom.com)

"Pete Dashwood" <dashwood@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx> wrote in message news:69s1u6F3438eiU1@xxxxxxxxxxxxxxxxxxxxxxxxxxxxx

I'm currently looking at automated code conversion for migrating COBOL applications that use Indexed files to RDB.

(Data conversion is pretty easy and has been established for some time now.)

Code conversion is much more tricky.

Conceptually, I want to add a Data Access layer to the existing COBOL, which has no such layer. The code has indexed I-O scattered right through it as was usually the case for COBOL.

I plan to generate from a template, a module (actually a Class...) that will do all of the possible I-O for a given indexed file, except it will do it against a table set on a RDB. (A "table set" in this context is defined as a BASE table with any attached tables. Any COBOL OCCURS clauses in the original record definition cause an attached table to be generated, in line with 2NF, when the data is converted.)

Re: Opinions on approach, please...

The maintenance "module" will be tailored for each table set, and it will be an OO Class. This allows multiple users to access simultaneously, using their own instance of it. I call it MOST (Maintenance Object Server Template). This code is being generated as a COM server because it will be used from a COBOL and (later) from a C# environment (where it runs as unmanaged code).

I have a current Migration Toolset which is being updated to automate the task of replacing the ISAM access in the COBOL source with INVOKES of equivalent MOST object methods.

Converting the existing programs then comes down to mainly replacing indexed access with the invocation of a MOST object method, through an interface.

Here's the current definition of the interface block:

```
001711 01 MOST-interface-block.
001712 12 MOST-reserved. *> Do NOT change these fields
001713 *> once this block has been
001714 *> returned by MOST
001715 15 MOST-SQLSTATE pic x(5).
001716 15 MOST-SQLMsg pic x(512).
001717 15 MOST-file-status pic xx.
001718 15 MOST-access-mode pic x.
001719 88 MOST-sequential value '0'.
001720 88 MOST-random value '1'.
001721 88 MOST-dynamic value '2'.
001722 15 MOST-open-mode pic x.
001723 88 MOST-input value '0'.
001724 88 MOST-output value '1'.
001725 88 MOST-I-O value '2'.
001726 88 MOST-extend value '3'.
001727 15 MOST-action pic x.
001728 88 MOST-read value '0'.
001729 88 MOST-write value '1'.
001730 88 MOST-rewrite value '2'.
001731 88 MOST-start value '3'.
001732 88 MOST-delete value '4'.
001733 15 MOST-start-condition.
001734 88 MOST-relation pic XX.
001735 88 MOST-rel-data pic x(100).
001736 15 MOST-instance-stamp pic x(8). *> Do not pass this
block
001737 *> to any instance of MOST
001738 *> OTHER THAN the one
001739 *> which created it.
001740 15 filler pic x(30). *>
internal states and
001741 *> object references
001742 *> used by MOST
001743 12 MOST-data-buffer pic x(7600). *> total block
cannot
```

Re: Opinions on approach, please...

*>
exceed 8192 for a
*>
COM server.

When the MOST object is instantiated, this block is initialized and returned by the Factory class.

As you can see, the block caters for all of the possible access that can be done (and the MOST object validates that the actions are compliant with COBOL standards... no WRITE when opened INPUT, etc...).

The job of the MOST template is to "translate" these Indexed file actions into equivalent actions on the corresponding table set.
(The MOST template is written in COBOL).

For the most part it is fairly straightforward, and I am writing a C# module that scans the COBOL source using regular expressions to detect the ISAM actions, comments them out, and inserts INVOKEs to the MOST object methods instead. This is a bit like a source preprocessor, as the output from it is then compiled to become a MOST Class designed to handle a specific Indexed file.

All of the singleton type things are pretty easy, but the sequential processing is more difficult. I (think I...) need to declare and open a cursor when a START is encountered, translate the ensuing READs into FETCHes and then CLOSE it when we hit EOF (AT END in the existing code).

Now the reason I am posting...

I'd like to get some other opinions on the stickier bits... :-)

Obviously START is problematic because it has a condition attached to it. I can parse this condition and translate it into an SQL condition to go into a WHERE clause, but I'm not sure whether it might be better to do this with dynamic (PREPARED) SQL (I could almost pick up the COBOL condition verbatim) or to use a static definition for each of the possible relational conditions.
Any thoughts?

In every case MOST will return all fields (SELECT *) so the effect is the same as COBOL record processing. (This is not intended to be an end point; it is a start to get the existing system running against a RDB as datasource, rather than against indexed files, and without changing the existing logic.)

Have I missed an entirely better way of handling skip-sequential processing?
Can I do it without a cursor?

Remember, all of the actions carried out by MOST must be achievable in embedded SQL in COBOL. Currently, I haven't ruled out generating stored procedures for certain actions, but I don't want to do this unless it really makes sense.

Re: Opinions on approach, please...

Re: Opinions on approach, please...

All ideas and suggestions appreciated.

Thanks,

Pete.

—

"I used to write COBOL...now I can do anything."