Programmers' Corner

Conditional compilation and Db2 DCLGEN

Don Leahy | Jun 16, 2022 | Tags: db2 dclgen cobol productivity | 37 Views Like

I first mentioned conditional compilation in a 2017 blog called COBOL 6.2 goodies. I didn't say much about it because frankly, I lacked the imagination to come up with a plausible use case for using this feature. I could think of unlikely cases such as wanting to compile the same code for execution under CICS or IMS, but I didn't see a burning demand for such a feature.

Recently, I was made aware of a situation that turned out to be an ideal candidate for conditional compilation.

The problem

Many of our COBOL programs read flat file unloads of our Db2 tables. The layout of the file is usually identical to the COBOL part of the Db2 DCLGEN member that was generated for the table, so the path of least resistance was to include the DCLGEN member in the COBOL program. Until recently that meant using the SQL INCLUDE statement and the Db2 precompiler to resolve the external reference. You couldn't bring it in via the COBOL COPY statement because the DECLARE TABLE part of the DCLGEN member is not valid COBOL syntax. The precompiler took care of that by commenting out the DECLARE TABLE in its modified COBOL code. [For those of you who don't write code every day, here is an example of a copy member created by the DCLGEN utility][1]:

```
D999       TDBP.C.P.BASE0.CPY(LDASNFHT) - 01.00                Columns 00001 0008
Command ===> _                                                  Scroll ===> CSR
****** ***************************** Top of Data ********************************
000001 ********************************************************************
000002 * DCLGEN TABLE(VDASNFHT)                                           *
000003 *        LIBRARY(DAST.NFHIST.DCLGEN(LDASNFHT))                     *
000004 *        ACTION(REPLACE)                                           *
000005 *        LANGUAGE(COBOL)                                           *
000006 *        QUOTE                                                     *
000007 * ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS     *
000008 ********************************************************************
000009     EXEC SQL DECLARE VDASNFHT TABLE
000010     ( BRANCH_NUM                     DECIMAL(5, 0) NOT NULL,
000011       ACOUNT_NUM                     DECIMAL(7, 0) NOT NULL,
000012       EVENT_D                        DATE NOT NULL,
000013       EVENT_PROC_TS                  TIMESTAMP NOT NULL,
000014       EVENT_ID                       DECIMAL(3, 0) NOT NULL,
000015       EVENT_SEQ_NUM                  DECIMAL(3, 0) NOT NULL,
000016       LOGON_ID                       CHAR(8) NOT NULL,
000017       SUPV_LOGON_ID                  CHAR(8) NOT NULL,
000018       HISTORY_VALUE                  VARCHAR(254) NOT NULL
000019     ) END-EXEC.
000020 ********************************************************************
000021 * COBOL DECLARATION FOR TABLE VDASNFHT                             *
000022 ********************************************************************
000023  01  DCLVDASNFHT.
000024      10 BRANCH-NUM           PIC S9(5)V USAGE COMP-3.
000025      10 ACOUNT-NUM           PIC S9(7)V USAGE COMP-3.
000026      10 EVENT-D              PIC X(10).
000027      10 EVENT-PROC-TS        PIC X(26).
000028      10 EVENT-ID             PIC S9(3)V USAGE COMP-3.
000029      10 EVENT-SEQ-NUM        PIC S9(3)V USAGE COMP-3.
000030      10 LOGON-ID             PIC X(8).
000031      10 SUPV-LOGON-ID        PIC X(8).
000032      10 HISTORY-VALUE.
000033         49 HISTORY-VALUE-LEN  PIC S9(4) USAGE COMP.
000034         49 HISTORY-VALUE-TEXT  PIC X(254).
000035 ********************************************************************
000036 * THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 9         *
000037 ********************************************************************
```

This created a problem because in order to use the Db2 precompiler you had to choose an Endevor processor that was designed for Db2. The Db2 precompiler creates a DBRM when a DCLGEN generated member in INCLUDE'd, even if there are no other SQL statements in the program. Binding such a DBRM into a package was wasted processing, as the package would never be executed. (i.e. DECLARE TABLE is not an executable SQL statement).

The game changed recently when we migrated to COBOL 6.2. For more information, see COBOL 6.2: something else you need to know.

One of the new capabilities enabled by using the SQL Coprocessor was the ability to use the COBOL COPY verb to bring in DCLGEN generated members. If **SQL("APOSTSQL,VERSION(Annnnnn)")** is in effect the compiler accepts DECLARE TABLE as valid syntax instead of rejecting it. However, this did not address the drawback of generating unnecessary and useless DBRMs just because there is a DECLARE TABLE statement in the program..

One possible solution is to perform major surgery on the DCLGEN generated member. The COBOL portion of the member could be segregated into a separate member and the flat file program can bring it in via COPY and the compile can run with SQL(NO), which of course suppresses the creation of a DBRM member.

Conditional compilation

A dive into the COBOL manual suggested another solution: conditional compilation. Specifically, to exploit the existence of a special variable that is set when **SQL("APOSTSQL,VERSION(Annnnnn)")** is used as a compile option. First, the program has to be changed to use COPY in lieu of SQL INCLUDE:

```
000016       WORKING-STORAGE SECTION.
000017
000018           COPY        LDASNFHT.
000019
```

Once this is done, a trivial change to the DCLGEN generated member is all that is required to exploit the special variable:

Similar Entri

DBA IMS p
Blog: Pro
Don Leahy
Updated D
0

New DBD
Blog: Pro
Don Leahy
Updated N
0

New versic
Blog: Pro
Don Leahy
Updated N
1

Abending
Blog: Pro
Don Leahy
Updated O
0

Central Ca
Blog: Pro
Don Leahy
Updated S
3

Similar Ideas

Re: Valuin
Ideation Bl
Marjorie An
Updated Ju
0

Archive

December 2022
November 2022
October 2022
September 2022
August 2022
July 2022
June 2022
May 2022
April 2022
March 2022
February 2022
January 2022
December 2021
November 2021
October 2021
September 2021
August 2021
July 2021
June 2021
May 2021
April 2021

```
0999 ISREDDE2 T.LEAHYD2.CPY(LDASNFHT) - 01.02                Columns 00001 00080
Command ===>                                                  Scroll ===> CSR
****** ********************************* Top of Data ********************************
000001       >>IF SQL
000002      ************************************************************
000003      * DCLGEN TABLE(VDASNFHT)                                    *
000004      *        LIBRARY(DAST.NFHIST.DCLGEN(LDASNFHT))              *
000005      *        ACTION(REPLACE)                                    *
000006      *        LANGUAGE(COBOL)                                    *
000007      *        QUOTE                                              *
000008      * ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS *
000009      ************************************************************
000010           EXEC SQL DECLARE VDASNFHT TABLE
000011           ( BRANCH_NUM                     DECIMAL(5, 0) NOT NULL,
000012             ACOUNT_NUM                     DECIMAL(7, 0) NOT NULL,
- - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 5 Line(s) not Displayed
000018             SUPV_LOGON_ID                  CHAR(8) NOT NULL,
000019             HISTORY_VALUE                  VARCHAR(254) NOT NULL
000020           ) END-EXEC.
000021       >>END-IF
000022      ************************************************************
000023      * COBOL DECLARATION FOR TABLE VDASNFHT                      *
000024      ************************************************************
000025       01  DCLVDASNFHT.
000026           10 BRANCH-NUM           PIC S9(5)V USAGE COMP-3.
- - -  - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - 6 Line(s) not Displayed
000033           10 SUPV-LOGON-ID        PIC X(8).
000034           10 HISTORY-VALUE.
000035              49 HISTORY-VALUE-LEN  PIC S9(4) USAGE COMP.
000036              49 HISTORY-VALUE-TEXT  PIC X(254).
000037      ************************************************************
000038      * THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 9  *
000039      ************************************************************
****** ********************************* Bottom of Data *****************************
```

The **>>IF SQL** statement means that the code between the >>IF and the >>END-IF is only included in the program if **SQL("APOSTSQL,VERSION(Annnnnn)")** is used to compile the program. This is of course the option that you would choose if your program actually accesses the Db2 table via SQL statements. You can see the evidence in the compile listing:

```
0999 E    USST.LEAHYD2.TESTSQL1.LIST
Command ===>
 000016              WORKING-STORAGE SECTION.
 000017                   copy       LDASNFHT.
 000018C             >>IF SQL
 000019C            ************************************************************
 000020C            * DCLGEN TABLE(VDASNFHT)                                    *
 000021C            *        LIBRARY(DAST.NFHIST.DCLGEN(LDASNFHT))              *
 000022C            *        ACTION(REPLACE)                                    *
 000023C            *        LANGUAGE(COBOL)                                    *
 000024C            *        QUOTE                                              *
 000025C            * ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS *
 000026C            ************************************************************
 000027C                 EXEC SQL DECLARE VDASNFHT TABLE                        *
 000028C                 ( BRANCH_NUM                     DECIMAL(5, 0) NOT NULL,
 000029C                   ACOUNT_NUM                     DECIMAL(7, 0) NOT NULL,
 000030C                   EVENT_D                        DATE NOT NULL,
 000031C                   EVENT_PROC_TS                  TIMESTAMP NOT NULL,
 000032C                   EVENT_ID                       DECIMAL(3, 0) NOT NULL,
 000033C                   EVENT_SEQ_NUM                  DECIMAL(3, 0) NOT NULL,
 000034C                   LOGON_ID                       CHAR(8) NOT NULL,
 000035C                   SUPV_LOGON_ID                  CHAR(8) NOT NULL,
 000036C                   HISTORY_VALUE                  VARCHAR(254) NOT NULL
 000037C                 ) END-EXEC.
 000038C             >>END-IF
 000039C            ************************************************************
 000040C            * COBOL DECLARATION FOR TABLE VDASNFHT                      *
 000041C            ************************************************************
 000042C             01  DCLVDASNFHT.
 000043C                 10 BRANCH-NUM           PIC S9(5)V USAGE COMP-3.
 000044C                 10 ACOUNT-NUM           PIC S9(7)V USAGE COMP-3.
 000045C                 10 EVENT-D              PIC X(10).
 000046C                 10 EVENT-PROC-TS        PIC X(26).
 000047C                 10 EVENT-ID             PIC S9(3)V USAGE COMP-3.
 000048C                 10 EVENT-SEQ-NUM        PIC S9(3)V USAGE COMP-3.
 000049C                 10 LOGON-ID             PIC X(8).
 000050C                 10 SUPV-LOGON-ID        PIC X(8).
 000051C                 10 HISTORY-VALUE.
 000052C                    49 HISTORY-VALUE-LEN  PIC S9(4) USAGE COMP.
 000053C                    49 HISTORY-VALUE-TEXT  PIC X(254).
 000054C            ************************************************************
 000055C            * THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 9  *
 000056C            ************************************************************
```

To compile a program that only reads the flat file data from the Db2 table, all you have to do is to replace the SQL INCLUDE with COPY as shown in the first example and choose a non-Db2 Endevor processor when you compile the program. This will compile **without** SQL("APOSTSQL,VERSION(Annnnnn)") , and the "SQL" variable will be set to false at compile time. The DECLARE TABLE will *not* be included in the code. The compiler makes this really obvious because instead of omitting the DECLARE TABLE from the listing, it is shown commented out:

```
000018                   COPY       LDASNFHT.
000019C             >>IF SQL
000020C            ************************************************************
000021C            * DCLGEN TABLE(VDASNFHT)                                    *
000022C            *        LIBRARY(DAST.NFHIST.DCLGEN(LDASNFHT))              *
000023C            *        ACTION(REPLACE)                                    *
000024C            *        LANGUAGE(COBOL)                                    *
000025C            *        QUOTE                                              *
000026C            * ... IS THE DCLGEN COMMAND THAT MADE THE FOLLOWING STATEMENTS *
000027C            ************************************************************
000028C            *    EXEC SQL DECLARE VDASNFHT TABLE
000029C            *    ( BRANCH_NUM                     DECIMAL(5, 0) NOT NULL,
000030C            *      ACOUNT_NUM                     DECIMAL(7, 0) NOT NULL,
000031C            *      EVENT_D                        DATE NOT NULL,
000032C            *      EVENT_PROC_TS                  TIMESTAMP NOT NULL,
000033C            *      EVENT_ID                       DECIMAL(3, 0) NOT NULL,
000034C            *      EVENT_SEQ_NUM                  DECIMAL(3, 0) NOT NULL,
000035C            *      LOGON_ID                       CHAR(8) NOT NULL,
000036C            *      SUPV_LOGON_ID                  CHAR(8) NOT NULL,
000037C            *      HISTORY_VALUE                  VARCHAR(254) NOT NULL
000038C            *    ) END-EXEC.
000039C             >>END-IF
000040C            ************************************************************
000041C            * COBOL DECLARATION FOR TABLE VDASNFHT                      *
000042C            ************************************************************
000043C             01  DCLVDASNFHT.
000044C                 10 BRANCH-NUM           PIC S9(5)V USAGE COMP-3.
000045C                 10 ACOUNT-NUM           PIC S9(7)V USAGE COMP-3.
000046C                 10 EVENT-D              PIC X(10).
000047C                 10 EVENT-PROC-TS        PIC X(26).
000048C                 10 EVENT-ID             PIC S9(3)V USAGE COMP-3.
000049C                 10 EVENT-SEQ-NUM        PIC S9(3)V USAGE COMP-3.
000050C                 10 LOGON-ID             PIC X(8).
000051C                 10 SUPV-LOGON-ID        PIC X(8).
000052C                 10 HISTORY-VALUE.
000053C                    49 HISTORY-VALUE-LEN  PIC S9(4) USAGE COMP.
000054C                    49 HISTORY-VALUE-TEXT  PIC X(254).
000055C            ************************************************************
000056C            * THE NUMBER OF COLUMNS DESCRIBED BY THIS DECLARATION IS 9  *
000057C            ************************************************************
```

Condition compilation provides a very simple way to permanently solve the unnecessary DBRM problem without creating two different versions of the a copybook that would have to be kept in sync forever.

March 2021
February 2021
January 2021
December 2
November 2020
October 2020
September 2020
August 2020
July 2020
June 2020
May 2020
April 2020
March 2020
February 2020
January 2020
December 2019
November 2019
October 2019
September 2019
August 2019
July 2019
June 2019
May 2019
April 2019
March 2019
February 2019
January 2019
December 2018
November 2018
October 2018
September 2018
August 2018
July 2018
June 2018
May 2018
April 2018
March 2018
February 2018
January 2018
December 2017
November 2017
October 2017
September 2017
August 2017
July 2017
June 2017
May 2017
April 2017
March 2017
February 2017
January 2017
December 2016
November 2016
October 2016
September 2016
August 2016

>>IF SQL ...>>END-IF can safely be used for all members created by DCLGEN, even when there is no intention of using the member in a non-Db2 program. I would recommend it as a coding standard, as it takes little effort, does not increase risk and has benefits. I would even go so far as to recommend to IBM that an option be added on the DCLGEN creation panel to allow the user to request the inclusion of >>IF SQL....>>END-IF when the DCLGEN member is created.

---

[1]At the risk of sounding pedantic, DCLGEN is the name of the *command* that creates a copy member which includes the DECLARE TABLE and the equivalent COBOL declaration for the table columns. I know that common practice is to refer to the output of the DCLGEN command as a "DCLGEN", but strictly speaking that is incorrect usage. Having gotten that on the record, I have to admit to being one of those who popularized that usage back when Db2 was first introduced here at the Bank. Mea Culpa. In my defense, in those days copybooks had not been integrated into the common library TDBP.C.P.BASE0.CPY. Most applications kept their copybooks in their MASTER.SOURCE libraries. DBA attempted to keep IMS segment defintions ("segment source") in a common location, and it was thought that they would do the same for DCLGEN output. Since they referred to the IMS stuff as "segment source", it was useful in common conversation to refer to DCLGEN output as a "DCLGEN".

Table of Contents

Modified on Jun 17, 2022 by Don Leahy

Add a Comment | Edit | More Actions

Comments (0)

There are no comments to display

Add a Comment

Previous Entry | Main | Next Entry

Feed for Blog Entries | Feed for Blog Comments | Feed for Comments for this Entry

July 2016
June 2016
May 2016
April 2016
March 2016
February 2016
January 2016
December 2015
November 2015
October 2015
September 2015
August 2015
July 2015
June 2015
May 2015
April 2015
March 2015
February 2015
January 2015
December 2014
November 2014
October 2014
September 2014

Blog Author


Don Le


Jerry Z


Marina

1 - 3 of 5 authors