Procedures, Triggers and User-Defined Function Tables

CTC 07 – App Dev – Wednesday, April 27 2016

Don Friedel Jr

Contents

- Procedures
  - External
  - Lite (SQL)
- Triggers
- UDFTs
- Examples

Procedures

- User-written programs which execute inside the Multi-User Facility (MUF) as a separate subtask. They can be explicitly called or automatically triggered
  - External Procedures are written using LE-conforming Assembler, COBOL, PL/I or C and defined to the system via a CREATE PROCEDURE statement. The program source code for the procedure must be preprocessed before issuing the CREATE PROCEDURE statement
  - SQL Procedures (Lite) are not written in a host-language program, but expressed as a series of SQL statements embedded within the CREATE PROCEDURE statement
- They can be coded to perform almost any task and generally contain SQL statements (although they do not have to)
- SQL has no way to verify the compatibility of the code with the procedure defined by the CREATE PROCEDURE statement, it is the sole responsibility of the programmer to ensure the parameter list is accurate (as well as that the procedure does what it is supposed to do!)
- Non-SQL requests (RAAT, etc.) are not allowed inside procedures.
- Triggers and their associated procedures cannot contain any Commit or Rollback logic.

Procedures (cont'd)

- External
  - Must have the libraries containing the program(s) to be executed as procedures added to the MUF library concatenation
  - Execute inside the MUF as a separate subtask
  - Execute under the same LUW as the task that either caused the trigger to fire, or that explicitly called the procedure

Procedures (cont'd)

- SQL Procedures (Lite)
  - Consist of user-written program logic composed of SQL statements and contained entirely within the CREATE PROCEDURE statements
  - Some of the SQL statements that are available for SQL Procedures include
    - SQL Variable declaration
    - Assignment Statement
    - CASE Statement
    - Compound Statement
    - IF-THEN Statement
    - ITERATE Statement
    - LOOP, REPEAT-UNTIL, WHILE Statements with optional LOOPLIMIT
    - Diagnostic statements (DATACOM DUMP, RAISE ERROR, SIGNAL, RESIGNAL)

Procedures (cont'd)

- Simulate Datacom Procedure Statement
  - Provides support in DBSQLPR for SQL variables
  - This would be useful when want to call a procedure that has parameters, therefore need to use the Simulate Datacom Procedure statement in order to define variables within DBSQLPR so that the procedure can be called

Procedures (cont'd)

- Miscellaneous Notes
  - DBSQLPR Option TERM=@
  - Since the procedure statements within the CREATE PROCEDURE statement end in semicolons ';' and in DBSQLPR the default termination character is also a semicolon, we recommend using the TERM= option and specify a different character for DBSQLPR to recognize as the terminating character, such as at sign (@)
  - This allows DBSQLPR to skip over semicolons embedded in statements and recognize the @ as the end of the statement

Triggers

- Defined to be ‘fired’ when an insert, delete, or update of a row in the selected table occurs
- Fired regardless of where or how the change was issued
- Definition determines what causes it to fire, on what table, and can be tailored to specific data values, and whether it fires before or after the event occurs
- It has no capability to perform some action, it relies upon the procedure it calls to perform whatever activities are desired
- A procedure that is part of a triggered action cannot have OUT or INOUT parameters
- If multiple triggers are defined for a single event, they will be invoked according to the date and time created, with the most recently created invoked last

UDFTs

- User-Defined Function Tables are very similar to a Procedure. Differences are:
  - Used as a normal table in a query (i.e. a program or trigger does not invoke it, it gets invoked when it's name is referenced as if it was a normal table)
    - Primary difference between a UDFT and a normal table is that the UDFT is Read-only
  - Returns a table row, whose columns are declared in the CREATE FUNCTION statement in the RETURNS clauses
  - Only External program can be used, there is no support for UDFTs to be created as SQL Procedures
  - The UDFT can keep track of its state throughout the life of the cursor in a memory area called the "scratch pad," which is passed to it as a parameter on each call
  - A trigger cannot be defined on a UDFT. However, you can think of a UDFT as a triggered procedure, but the trigger is to obtain the next row of the table
- Limitations
  - Support has not been added to record dependencies, so if a UDFT changes, the queries that use the UDFT will not be automatically rebound

Example 1 – SQL Procedure

```
//JOBLIB   DD DSN=DCMLV2.DBL2SQL.V140.CUSLIB,DISP=SHR
//         DD DSN=DCMPS.DATCM140.PRD.CABDLOAD,DISP=SHR
//SQLEXEC  EXEC PGM=DBSQLPR,
//         PARM='PRTWIDTH=999,INPUTWIDTH=72,AUTHID=SYSADM,TERM=@'
//SYSUDUMP DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//SYSIN    DD  *

CREATE TABLE DFTEMP
 (
 CUST_ID    INTEGER NOT NULL PRIMARY KEY,
 NAME       CHAR(10) NOT NULL,
 NUMBER     INTEGER NOT NULL,
 STREET     CHAR(20) NOT NULL,
 APT        INTEGER
)@
```

Example 1 – SQL Procedure (cont'd)

```
CREATE PROCEDURE FIXADDR
( )
MODIFIES SQL DATA
LANGUAGE SQL
STMT: BEGIN ATOMIC
UPDATE DFTEMP
  SET STREET = 'MNO ST'
  WHERE STREET = 'ZYZ ST'
;
END STMT@

CREATE TRIGGER CATCHADDR AFTER INSERT ON DFTEMP
 REFERENCING NEW ROW AS NEWROW
 WHEN NEWROW.STREET = 'ZYZ ST'
 EXECUTE PROCEDURE FIXADDR
 @

COMMIT@
```

Example 1 – SQL Procedure (cont'd)

```
SELECT * FROM DFTEMP;

    CUST_ID NAME                 NUMBER STREET                         APT
   INT N.N. CHAR(10) N.N.     INT N.N. CHAR(20) NOT NULL               INT
___________ _____________ ____________ ____________________ ____________
         10 TOMMY                  13 XYZ ST                         12
         20 DAVID                  26 ABC ST                         26
___ 2 rows returned ___


INPUT STATEMENT:
INSERT INTO DFTEMP VALUES (30, 'EDDIE', 42, 'ZYZ ST', 422);

___ SQLCODE=0, SQLSTATE=00000, ROWS AFFECTED=1 ___


INPUT STATEMENT:
```

Example 1 – SQL Procedure (cont'd)

```
COMMIT;

___ SQLCODE=0, SQLSTATE=00000 ___


INPUT STATEMENT:
SELECT * FROM DFTEMP;

    CUST_ID NAME                NUMBER STREET                        APT
   INT N.N. CHAR(10) N.N.     INT N.N. CHAR(20) NOT NULL             INT
___________ _____________ ____________ ____________________ ____________
         10 TOMMY                  13 XYZ ST                        12
         20 DAVID                  26 ABC ST                        26
         30 EDDIE                  42 MNO ST                       422
___ 3 rows returned ___
```

Example 2 – COBOL

```
//PRECMP01 EXEC PGM=DBXMMPR
//WORK1    DD   DSN=&&WORK1,UNIT=SYSDA,DISP=(NEW,PASS),
//          DCB=(RECFM=F,LRECL=80,BLKSIZE=80),SPACE=(TRK,(1,1))
//WORK2    DD   DSN=&&WORK2,UNIT=SYSDA,DISP=(NEW,PASS),
//          DCB=(RECFM=F,LRECL=80,BLKSIZE=80),SPACE=(TRK,(1,1))
//WORK3    DD   DSN=&&WORK3,UNIT=SYSDA,DISP=(NEW,PASS),
//          DCB=(RECFM=F,LRECL=80,BLKSIZE=80),SPACE=(TRK,(1,1))
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SNAPPER  DD   SYSOUT=*
//SYSPUNCH DD   DSN=&&SQLCOB,UNIT=SYSDA,DISP=(NEW,PASS),
//          DCB=(RECFM=F,LRECL=80,BLKSIZE=80),SPACE=(TRK,(1,1))
//SYSUDUMP DD   SYSOUT=*
//SYSIN    DD   *
      *$DBSQLOPT AUTHID=SYSADM COBMODE=VSCOB2
      *$DBSQLOPT PLANAME=DAF5
      ****************************
      *     PLANAME=DAFCOB       *
      ****************************
 . . . COBOL PROGRAM FOR PROCEDURE
```

Example 2 – COBOL (cont'd)

```
/COBOL   EXEC PGM=IGYCRCTL,                                              X
/             PARM=(ARITH(EXTEND),NUM,NODYN,APOST,NOSEQUENCE),
/             COND=(8,LT)
/STEPLIB  DD  DSNAME=MVSSYS.COBOL.V4R2M0.MAINT.SIGYPROC,DISP=SHR
/SYSLIN   DD  DISP=(MOD,PASS),DSN=&&COBOLOD,
/             UNIT=SYSDA,SPACE=(TRK,(15,15))
/SYSPRINT DD  SYSOUT=*
/SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
/SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
/SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
/SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
/SYSUT5   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
/SYSUT6   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
/SYSUT7   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
/SYSIN    DD  DSN=&&SQLCOB,UNIT=SYSDA,DISP=(OLD,DELETE,DELETE)
```

Example 2 – COBOL (cont'd)

```
//LINK   EXEC LKED,COND=(8,LT),
//            PARM.LKED='XREF,LIST,LET,MAP'
//LKED.SYSLIN DD DSN=&&COBOLOD,UNIT=SYSDA,DISP=(OLD,DELETE,DELETE)
//            DD DDNAME=SYSIN
//LKED.SYSLMOD DD DSN=FRIDO03.COBOL.LOADLIB,DISP=SHR
//LKED.SYSLIB  DD DSN=CEE.SCEELKED,DISP=SHR
//*KED.SYSLIB  DD DSNAME=MVSSYS.COB2.V1R3M2.COB2COMP,DISP=SHR
//LKED.OBJLIB  DD DSN=DCMPS.DATCM140.DEV.CABDLOAD,DISP=SHR
//LKED.SYSIN   DD *
 INCLUDE OBJLIB(DBSBTPR)
 INCLUDE OBJLIB(DBXHVPR)
 INCLUDE OBJLIB(DBXHAPR)
 INCLUDE LODLIB(DBXPIPR)
 ENTRY BEGIN
 NAME DAFCOB(R)
```

Example 3 – UDFT – Create table and contents

```
//DAFUDFT JOB (125301000),FRIDO03,CLASS=K,MSGCLASS=X,
//       PRTY=6,REGION=1024K
//JOBLIB   DD DSN=DCMLV2.DBL2SQL.V140.CUSLIB,DISP=SHR
//         DD DSN=DCMPS.DATCM140.PRD.CABDLOAD,DISP=SHR
//SQLEXEC  EXEC PGM=DBSQLPR
//SYSUDUMP DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//OPTIONS  DD  *
AUTHID=SYSADM
//SYSIN    DD  *
CREATE TABLE UDFTIN (
       FIRSTNAME  VARCHAR(32) NOT NULL,
       LASTNAME   VARCHAR(32) NOT NULL,
       SALARY     DEC(7,0)    NOT NULL)
;
INSERT INTO UDFTIN VALUES ('ROBERT',   'REDFORD',  1000000);
INSERT INTO UDFTIN VALUES ('HARRISON', 'FORD',     2000000);
INSERT INTO UDFTIN VALUES ('ANGELINA', 'JOLIE',    3000000);
INSERT INTO UDFTIN VALUES ('NATALIE',  'PORTMAN',  4000000);
```

Example 3 – UDFT – Create UDFT

```
//DAFUDFT JOB (125301000),FRIDO03,CLASS=K,MSGCLASS=X,
//       PRTY=6,REGION=1024K
//JOBLIB   DD DSN=DCMLV2.DBL2SQL.V140.CUSLIB,DISP=SHR
//         DD DSN=DCMPS.DATCM140.PRD.CABDLOAD,DISP=SHR
//SQLEXEC  EXEC PGM=DBSQLPR
//SYSUDUMP DD  SYSOUT=*
//STDERR   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//STDOUT   DD  SYSOUT=*
//OPTIONS  DD  *
AUTHID=SYSADM
//SYSIN    DD  *
CREATE FUNCTION SYSADM.DONKUDFT (UDFPARM1 CHAR(4))
       RETURNS TABLE ( FIRSTNAME VARCHAR(32),
                       LASTNAME  VARCHAR(32),
                       SALARY DEC(7,0))
       SCRATCHPAD 800
       CARDINALITY 20
       LANGUAGE COBOL
       READS SQL DATA
       EXTERNAL NAME DONUDFT
       PARAMETER STYLE DATACOM SQL
       QUERYNO 1959126768        ;
```

Example 3 – UDFT - Function

```
*$DBSQLOPT AUTHID=SYSADM PROCSQLUSAGE=READS ISOLEVEL=C
*$DBSQLOPT GENSECTN=O
*********************************************
*SEE UDFTCRIN TO CREATE INPUT TABLE.
*SEE UDFTLINK FOR COMPILE,LINK JCL.
*********************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. DONUDFT.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-370.
 OBJECT-COMPUTER. IBM-370.
 INPUT-OUTPUT SECTION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 LOCAL-STORAGE SECTION.
     EXEC SQL BEGIN DECLARE SECTION END-EXEC
 LINKAGE SECTION.
*****************************************************
* Declare each of the parameters                    *
*****************************************************
 01  UDFPARM1 PIC XXXX.
```

Example 3 – UDFT – Function (cont'd)

```
*********************************************************
* Declare these variables for result parameters         *
*********************************************************
 01  FIRSTNAME-PARM.
     49 FIRSTNAME-LN  PIC S9999 COMP.
     49 FIRSTNAME     PIC X(32).
 01  LASTNAME-PARM.
     49 LASTNAME-LN   PIC S9999 COMP.
     49 LASTNAME      PIC X(32).
 01  SALARY           PIC S9(7) COMP-3.

*********************************************************
* Declare a null indicator for each parameter           *
*********************************************************
 01  UDF-IND1 PIC S9(4) USAGE COMP.

*********************************************************
* Declare a null indicator for result parameter         *
*********************************************************
 01  UDF-RIND1 PIC S9(4) USAGE COMP.
 01  UDF-RIND2 PIC S9(4) USAGE COMP.
 01  UDF-RIND3 PIC S9(4) USAGE COMP.
```

Example 3 – UDFT – Function (cont'd)

```
**********************************************************
* Declare the SQLSTATE that can be set by the            *
* user-defined function                                  *
**********************************************************
 01  UDF-SQLSTATE PIC X(5).
**********************************************************
* Declare the qualified function name                    *
**********************************************************
 01  UDF-FUNC.
     49 UDF-FUNC-LEN PIC 9(4) USAGE BINARY.
     49 UDF-FUNC-TEXT PIC X(137).
**********************************************************
* Declare the specific function name                     *
**********************************************************
 01  UDF-SPEC.
     49 UDF-SPEC-LEN PIC 9(4) USAGE BINARY.
     49 UDF-SPEC-TEXT PIC X(128).
**********************************************************
* Declare SQL diagnostic message token                   *
**********************************************************
 01  UDF-DIAG.
     49 UDF-DIAG-LEN PIC 9(4) USAGE BINARY.
     49 UDF-DIAG-TEXT PIC X(1000).
```

Example 3 – UDFT – Function (cont'd)

```
*********************************************************
* Declare these variables for result parameters         *
*********************************************************
 01  FIRSTNAME-PARM.
     49 FIRSTNAME-LN  PIC S9999 COMP.
     49 FIRSTNAME     PIC X(32).
 01  LASTNAME-PARM.
     49 LASTNAME-LN   PIC S9999 COMP.
     49 LASTNAME      PIC X(32).
 01  SALARY           PIC S9(7) COMP-3.

*********************************************************
* Declare a null indicator for each parameter           *
*********************************************************
 01  UDF-IND1 PIC S9(4) USAGE COMP.

*********************************************************
* Declare a null indicator for result parameter         *
*********************************************************
 01  UDF-RIND1 PIC S9(4) USAGE COMP.
 01  UDF-RIND2 PIC S9(4) USAGE COMP.
 01  UDF-RIND3 PIC S9(4) USAGE COMP.
```

Example 3 – UDFT – Function (cont'd)

```
           MOVE +0      TO UDF-DIAG-LEN.
           MOVE +0      TO SQLCA-SQLCODE.
           MOVE +800    TO UDF-SPAD-LEN.
           MOVE +0      TO UDF-RIND1.
           MOVE +0      TO UDF-RIND2.
           MOVE '00000' TO UDF-SQLSTATE.
      *    MOVE +15     TO UDF-DIAG-LEN.
      *    MOVE 'UDFTCOB ENTERED' TO UDF-DIAG-TEXT.

           EXEC SQL
               DECLARE CRS1 INSENSITIVE SCROLL CURSOR FOR
               SELECT * FROM  UDFTIN
                WHERE LASTNAME LIKE '%FORD%'
           END-EXEC.
           IF UDF-CALL-TYPE EQUAL -1
                   MOVE ZERO TO ROWS-RETURNED
                   MOVE 'EYECATCH' TO SCRATCH-EYE
                   EXEC SQL
                       OPEN CRS1
                   END-EXEC
           END-IF
```

Example 3 – UDFT – Function (cont'd)

```
IF UDF-CALL-TYPE EQUAL 0
    ADD +1 TO ROWS-RETURNED
    EXEC SQL
        FETCH      CRS1 INTO :FIRSTNAME-PARM,
                             :LASTNAME-PARM,
                             :SALARY
    END-EXEC
END-IF
IF UDF-CALL-TYPE EQUAL 1
    EXEC SQL
        CLOSE CRS1
    END-EXEC
END-IF
IF SQLCODE EQUAL +100
    ADD +1 TO OPEN-CURSOR-CNT

    IF OPEN-CURSOR-CNT EQUAL TO 3
        MOVE +100    TO SQLCA-SQLCODE
        MOVE '02000' TO UDF-SQLSTATE
        MOVE +10     TO UDF-DIAG-LEN
        MOVE 'END OF SET' TO UDF-DIAG-TEXT
    ELSE
```

Example 3 – UDFT – Function (cont'd)

```
    ELSE
        IF OPEN-CURSOR-CNT LESS THAN 3
            EXEC SQL FETCH BEFORE CRS1 END-EXEC
            EXEC SQL
            FETCH      CRS1 INTO :FIRSTNAME-PARM,
                                 :LASTNAME-PARM,
                                 :SALARY
            END-EXEC
        END-IF
    END-IF
END-IF
IF SQLCODE LESS THAN ZERO
    MOVE '99999' TO UDF-SQLSTATE
    MOVE +31 TO UDF-DIAG-LEN
    MOVE 'ERROR IN READING CURSOR IN UDFT' TO UDF-DIAG-TEXT.
GOBACK.
```

Example 3 – UDFT – JCL/Test program

```
//DAFTSTL  JOB (125201000),'FRIEDEL',
//             CLASS=K,MSGCLASS=X,REGION=2048K
//JOBLIB   DD DSN=DCMDEV.DB.DBDVMF.R150.CUSLIB,DISP=SHR
//         DD DSN=DCMPS.DATCM150.PRD.CABDLOAD,DISP=SHR
//PRECOMP  EXEC PGM=DBXMMPR
//WORK1    DD   DSN=&&WORK1,UNIT=SYSDA,DISP=(NEW,PASS),
//         DCB=(RECFM=F,LRECL=80,BLKSIZE=80),SPACE=(TRK,(1,1))
//WORK2    DD   DSN=&&WORK2,UNIT=SYSDA,DISP=(NEW,PASS),
//         DCB=(RECFM=F,LRECL=80,BLKSIZE=80),SPACE=(TRK,(1,1))
//WORK3    DD   DSN=&&WORK3,UNIT=SYSDA,DISP=(NEW,PASS),
//         DCB=(RECFM=F,LRECL=80,BLKSIZE=80),SPACE=(TRK,(1,1))
//SYSOUT   DD   SYSOUT=*
//SYSPRINT DD   SYSOUT=*
//SNAPPER  DD   SYSOUT=*
//SYSPUNCH DD   DSN=&&SQLCOB,UNIT=SYSDA,DISP=(NEW,PASS),
//         DCB=(RECFM=F,LRECL=80,BLKSIZE=80),SPACE=(TRK,(1,1))
//SYSUDUMP DD   SYSOUT=*
//SYSIN    DD   *
```

Example 3 – UDFT – JCL/Test program

```
*$DBSQLOPT AUTHID=SYSADM ISOLEVEL=C
 IDENTIFICATION DIVISION.
 PROGRAM-ID. UDFTTST.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SOURCE-COMPUTER. IBM-370.
 OBJECT-COMPUTER. IBM-370.
 INPUT-OUTPUT SECTION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 LOCAL-STORAGE SECTION.
     EXEC SQL BEGIN DECLARE SECTION END-EXEC
 01  FIRSTNAME  PIC X(32).
 01  LASTNAME   PIC X(32).
 01  SALARY     PIC S9(7) COMP-3.
     EXEC SQL END DECLARE SECTION END-EXEC.
```

Example 3 – UDFT – JCL/Test program

```
PROCEDURE DIVISION.
    ENTRY 'DBMSCBL'.
    EXEC SQL
        DECLARE CRS1 INSENSITIVE SCROLL CURSOR FOR
        SELECT *
         FROM TABLE(SYSADM.DONUDFT ('FORD')) AS T1
         ORDER BY LASTNAME
    END-EXEC.

    EXEC SQL OPEN CRS1 END-EXEC.
    EXEC SQL FETCH BEFORE CRS1 END-EXEC.
LOOP1.
    EXEC SQL
        FETCH      CRS1 INTO :FIRSTNAME, :LASTNAME, :SALARY
    END-EXEC.
    IF SQLCODE NOT EQUAL +0
        IF SQLCODE EQUAL +100
            DISPLAY "END OF SET"
        ELSE
            DISPLAY "INVALID SQLCODE ", SQLCODE
        END-IF
        GO TO ENDIT
    END-IF
    DISPLAY FIRSTNAME, LASTNAME, SALARY.
    GO TO LOOP1.
ENDIT.
    EXEC SQL CLOSE CRS1 END-EXEC.
    GOBACK.
```

Example 3 – UDFT – JCL/Test program

```
//*-------------------------------------------------------------------*
//* STEP2 COMPILE COBOL USER PROGRAM OUTPUT FROM COBOL PRECOMPILER    *
//*-------------------------------------------------------------------*
//*
//COBOL  EXEC PGM=IGYCRCTL,                                          X
//            PARM=(ARITH(EXTEND),NUM,NODYN,APOST,NOSEQUENCE),
//            COND=(8,LT)
//STEPLIB  DD  DSNAME=MVSSYS.COBOL.V4R2M0.MAINT.SIGYPROC,DISP=SHR
//*TEPLIB  DD  DSNAME=MVSSYS.COB2.V1R3M2.COB2COMP,DISP=SHR
//*SYSLIB  DD  DSN=CICS.V170.MVS.COBLIB,DISP=SHR
//SYSLIN   DD  DISP=(MOD,PASS),DSN=&&COBOLOD,
//             UNIT=SYSDA,SPACE=(TRK,(15,15))
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT2   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT3   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT4   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT5   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT6   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSUT7   DD  UNIT=SYSDA,SPACE=(CYL,(1,1))
//SYSIN    DD  DSN=&&SQLCOB,UNIT=SYSDA,DISP=(OLD,DELETE,DELETE)
```

Example 3 – UDFT – JCL/Test program

```
//*-------------------------------------------------------------------*
//* STEP3 LINK USER PROGRAM WITH SYSTEM MODULES                       *
//*-------------------------------------------------------------------*
//*
//LINK   EXEC LKED,COND=(8,LT),
//            PARM.LKED='XREF,LIST,LET,MAP'
//LKED.SYSLIN DD DSN=&&COBOLOD,UNIT=SYSDA,DISP=(OLD,DELETE,DELETE)
//            DD DDNAME=SYSIN
//LKED.SYSLMOD DD DSN=DCMLV2.INFO.GUPCH01.LOADLIB,DISP=SHR
//LKED.LODLIB  DD DSN=DCMPS.DATCM150.DEV.CABDLOAD,DISP=SHR
//LKED.SYSLIB  DD DSN=CEE.SCEELKED,DISP=SHR
//             DD DSN=DCMDEV.DBDT.TESTPROG.LOADLIB(CEEUOPT),DISP=SHR
//LKED.OBJLIB  DD DSN=DCMPS.DATCM140.PRD.CABDLOAD,DISP=SHR
//LKED.SYSIN   DD *
 INCLUDE OBJLIB(DBSBTPR)
 INCLUDE OBJLIB(DBXHVPR)
 INCLUDE OBJLIB(DBXHAPR)
 INCLUDE OBJLIB(DBXPIPR)
 ENTRY BEGIN
 NAME UDFTTST(R)
/*
//TEST     EXEC PGM=UDFTTST,COND=(8,LT)
//SYSPRINT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
```

FOR INFORMATION PURPOSES ONLY

Terms of this Presentation

This presentation was based on current information and resource allocations as of April 2016 and is subject to change or withdrawal by CA at any time without notice. Notwithstanding anything in this presentation to the contrary, this presentation shall not serve to (i) affect the rights and/or obligations of CA or its licensees under any existing or future written license agreement or services agreement relating to any CA software product; or (ii) amend any product documentation or specifications for any CA software product. The development, release and timing of any features or functionality described in this presentation remain at CA's sole discretion. Notwithstanding anything in this presentation to the contrary, upon the general availability of any future CA product release referenced in this presentation, CA will make such release available (i) for sale to new licensees of such product; and (ii) to existing licensees of such product on a when and if-available basis as part of CA maintenance and support, and in the form of a regularly scheduled major product release. Such releases may be made available to current licensees of such product who are current subscribers to CA maintenance and support on a when and if-available basis. In the event of a conflict between the terms of this paragraph and any other information contained in this presentation, the terms of this paragraph shall govern.

Certain information in this presentation may outline CA's general product direction. All information in this presentation is for your informational purposes only and may not be incorporated into any contract. CA assumes no responsibility for the accuracy or completeness of the information. To the extent permitted by applicable law, CA provides this presentation "as is" without warranty of any kind, including without limitation, any implied warranties or merchantability, fitness for a particular purpose, or non-infringement. In no event will CA be liable for any loss or damage, direct or indirect, from the use of this document, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if CA is expressly advised in advance of the possibility of such damages. CA confidential and proprietary. No unauthorized copying or distribution permitted.

Questions?