SWG BetaWorks

Multi Row Fetch, INSERT and Get Diagnostics

How to put it into code

Paul Fletcher (Fletchpl@uk.ibm.com)

Important Disclaimer

THE INFORMATION CONTAINED IN THIS PRESENTATION IS PROVIDED FOR INFORMATIONAL PURPOSES ONLY.

WHILE EFFORTS WERE MADE TO VERIFY THE COMPLETENESS AND ACCURACY OF THE INFORMATION CONTAINED IN THIS PRESENTATION, IT IS PROVIDED “AS IS”, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED.

IN ADDITION, THIS INFORMATION IS BASED ON IBM’S CURRENT PRODUCT PLANS AND STRATEGY, WHICH ARE SUBJECT TO CHANGE BY IBM WITHOUT NOTICE.

IBM SHALL NOT BE RESPONSIBLE FOR ANY DAMAGES ARISING OUT OF THE USE OF, OR OTHERWISE RELATED TO, THIS PRESENTATION OR ANY OTHER DOCUMENTATION.

NOTHING CONTAINED IN THIS PRESENTATION IS INTENDED TO, OR SHALL HAVE THE EFFECT OF:

- CREATING ANY WARRANTY OR REPRESENTATION FROM IBM (OR ITS AFFILIATES OR ITS OR THEIR SUPPLIERS AND/OR LICENSORS); OR
- ALTERING THE TERMS AND CONDITIONS OF THE APPLICABLE LICENSE AGREEMENT GOVERNING THE USE OF IBM SOFTWARE.

Agenda

- **Multi-Row INSERT – What it is**
- **How to change a program to use Multi-Row Insert**
- **Multi-Row FETCH – What it is**
- **How to code Multi-Row FETCH**

Multi-row INSERT What it is

- Inserts multiple rows on one API call
- Can be ATOMIC or NOT ATOMIC
- Can be static or dynamic SQL
- Significant performance boost

```
INSERT INTO T1 FOR :hv ROWS
    VALUES( :ARRAY1, :ARRAY2) ATOMIC;
```


What The Arrays Do NOT contain

Single Row

COL1

COL2

COL3

COL4

Multi Row (2 rows)

COL1 (1)

COL2 (1)

COL3 (1)

COL4 (1)

COL1 (2)

COL2 (2)

COL3 (2)

COL4 (2)

What The Arrays Do contain

Single Row

COL1
COL2
COL3
COL4

Multi Row (2 rows)

COL1 (1)	COL2 (1)	COL3 (1)	COL4 (1)
COL1 (2)	COL2 (2)	COL3 (2)	COL4 (2)

There is an array for each column

NOT for each Row

Setting your system to use the Sample programs

1. **Create your own copy of SYSIBM.SYSTABLEPART – Remove the Constraints to keep the testing simple**
2. **Use DSNTIAUL to Unload the data from SYSIBM.SYSTABLEPART into a file**
3. **Programs PLFSRI08 (for V8) or PLFSRI09 (for DB2 9) will show how to use Single row INSERTS to read a record from the file and INSERT into your copy of SYSTABLEPART**
4. **Programs PLFMRI08 (for V8) and PLFMRI09 (for DB2 9) will show how to read records from the file and INSERT 100 rows at a time**
5. **DCLGEN TPART8 (for V8) TPART9 (for DB2 9)**

   set STRUCTURE NAME to W-TPART

   set FIELD NAME PREFIX to W-

Single Row INSERT Program Logic

- **Reads one Record**
- **Moves it into the storage created by the DCLGEN**
- **Inserts a row**
- **Loop around until end of file**

DCLGEN Used for single row INSERT

```
EXEC SQL DECLARE DMPLF.SYSTABLEPART TABLE
( PARTITION                       SMALLINT NOT NULL,
  TSNAME                          VARCHAR(24) NOT NULL,
  DBNAME                          VARCHAR(24) NOT NULL,
  IXNAME                          VARCHAR(128) NOT NULL,
  IXCREATOR                       VARCHAR(128) NOT NULL,
  PQTY                            INTEGER NOT NULL,
  SQTY                            SMALLINT NOT NULL,
  STORTYPE                        CHAR(1) NOT NULL,
```

```
 01   W-TPART.
*     ************************************************************************
*                           PARTITION
      10 W-PARTITION              PIC S9(4) USAGE COMP.
*     ************************************************************************
      10 W-TSNAME.
*                           TSNAME LENGTH
          49 W-TSNAME-LEN         PIC S9(4) USAGE COMP.
*                           TSNAME
          49 W-TSNAME-TEXT        PIC X(24).
*     ************************************************************************
      10 W-DBNAME.
*                           DBNAME LENGTH
          49 W-DBNAME-LEN         PIC S9(4) USAGE COMP.
*                           DBNAME
          49 W-DBNAME-TEXT        PIC X(24).
*     ************************************************************************
      10 W-IXNAME.
*                           IXNAME LENGTH
          49 W-IXNAME-LEN         PIC S9(4) USAGE COMP.
*                           IXNAME
          49 W-IXNAME-TEXT        PIC X(128).
*     ************************************************************************
      10 W-IXCREATOR.
*                           IXCREATOR LENGTH
          49 W-IXCREATOR-LEN      PIC S9(4) USAGE COMP.
*                           IXCREATOR
          49 W-IXCREATOR-TEXT
             PIC X(128).
*     ************************************************************************
*                           PQTY
      10 W-PQTY                   PIC S9(9) USAGE COMP.
*     ************************************************************************
*                           SQTY
      10 W-SQTY                   PIC S9(4) USAGE COMP.
*     ************************************************************************
*                           STORTYPE
      10 W-STORTYPE               PIC X(1).
*     ************************************************************************
```

Single Row Insert Code

```
READ FD-VB-FILE INTO FILE-INPUT
  AT END MOVE 'Y' TO W-END-OF-FILE.
PERFORM B000-PROCESS-RECORDS UNTIL END-OF-FILE.
 GOBACK.
B000-PROCESS-RECORDS SECTION.
     MOVE FILE-INPUT TO W-TPART.
 *********************************************
 * INSERT ONE ROW AT A TIME
 *********************************************
    EXEC SQL
              INSERT INTO SYSTABLEPART VALUES
               (:W-PARTITION ,
              :W-TSNAME ,
              :W-DBNAME ,
              .....
              :W-RELCREATED )
    END-EXEC.
    READ FD-VB-FILE INTO FILE-INPUT
    AT END MOVE 'Y' TO W-END-OF-FILE.
```

Read first record

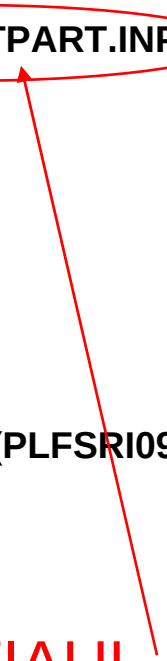
Move the record to the DCLGEN

Do the INSERT

Read the next record

Single Row INSERT Run JCL

```
//PLFSRI09 EXEC
PGM=IKJEFT01,DYNAMNBR=25,ACCT=SHORT,
//           REGION=4096K
//STEPLIB  DD DSN=SYS2.DB2.V910.SDSNLOAD,DISP=SHR
//        DD DSN=SYS2.DB2.V910.SDSNEXIT.PIC,DISP=SHR
//        DD  DSN=FLETCHP.MASTER.LOAD,DISP=SHR
//DDSEQ01R DD  DSN=FLETCHP.TPART.INPUT,DISP=SHR
//SYSUDUMP DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSABOUT DD  SYSOUT=*
//SYSTSIN  DD *
DSN SYSTEM(PB1I)
RUN PROGRAM(PLFSRI09) PLAN(PLFSRI09)
END
/*
```



File created by DSNTIAUL

Multi-Row INSERT Program Logic

- **A new Copy Member is included TPART28 or TPART29**
- **DCLGEN will not create a member for Multi-Row**
  - A Rexx Edit Macro (OCC) provides this function
- **Read a record**
- **Move it into the storage created by the single occurrence DCLGEN**
- **Move each field into the next occurrence of that field in the multiple occurrence DCLGEN**
- **Once 100 records have been read INSERT them using Multi-Row INSERT**
- **Loop until end of file**

2 Errors Encountered

On one DB2 9 system Precompiler failed with

HOST VARIABLE ARRAY "W-PARTITION" IS EITHER NOT DEFINED OR IS NOT USABLE

On another DB2 9 and V8 system it precompiled, compiled etc. but the run failed with

THE LENGTH OF INPUT HOST VARIABLE NUMBER 17 IS NEGATIVE OR GREATER THAN THE MAXIMUM

But Why?

Why where there problems?

- **PMR 24142,180,000 Has The Answer**
  - It refers to Application Programming and SQL Guide 2.4.3.6 Declaring host variable arrays
  - 2.4.3.6 could not be found but a search on Declaring host variable arrays in the PDF found:-

Example: The following example shows declarations of a fixed-length character array and a varying-length character array:

```
01 OUTPUT-VARS.
        05 NAME OCCURS 10 TIMES.
                49 NAME-LEN PIC S9(4) COMP SYNC.
                49 NAME-DATA PIC X(40).
        05 SERIAL-NUMBER PIC S9(9) COMP-4 OCCURS 10 TIMES.
```

Varchar Definition

- **The Varchar host variable is defined as**

```
05 NAME OCCURS 10 TIMES.
                49 NAME-LEN PIC S9(4) COMP SYNC.
                49 NAME-DATA PIC X(40).
```

The difference between the example and the DCLGEN is the word SYNC

What does SYNC do

- **The COBOL manual says**

The SYNCHRONIZED clause is never required, but can improve performance on some systems for binary items used in arithmetic.

For S9(4) COMP it aligns on a half word boundary i.e. 2bytes

For S9(9) COMP it aligns on a full word boundary i.e.4 bytes

Edit Macro – OCC to Add Multi-Row to A DCLGEN

- **/* REXX*/**
- **"ISREDIT MACRO (NUMOCC) NOPROCESS"** ← NUMOCC is No. Of Rows
- **"ISPEXEC CONTROL ERRORS RETURN"**
- **IF NUMOCC = '' THEN NUMOCC = 100** ← If no parm set it to 100
- **"ISREDIT SEEK ' 10 '"** ← Look for the first 10 level
- **RCD = RC** ← Store return code
- **CHTO = "' OCCURS "NUMOCC".'"** ← Build change to string
- **DO WHILE RCD = 0** ← Loop while return code is zero
- **"ISREDIT CHANGE '.' "CHTO** ← Issue change command
- **"ISREDIT SEEK ' 10 '"** ← Look for next 10 level
- **RCD = RC** ← Return code 4 is not found 0 is found
- **END**
- **"ISREDIT CHANGE ALL 'COMP.' 'COMP SYNC.'"** Finally change all comp To comp sync

New DCLGEN for Multi-Row after running OCC

```
01  W2-TPART.
    *   ************************************************************
    *                PARTITION
        10 W2-PARTITION        PIC S9(4) USAGE COMP OCCURS 100.
    *   ************************************************************
        10 W2-TSNAME OCCURS 100.
    *                TSNAME LENGTH
          49 W2-TSNAME-LEN     PIC S9(4) USAGE COMP SYNC.
    *                TSNAME
          49 W2-TSNAME-TEXT    PIC X(24).
    *   ************************************************************
        10 W2-DBNAME OCCURS 100.
    *                DBNAME LENGTH
          49 W2-DBNAME-LEN     PIC S9(4) USAGE COMP SYNC.
    *                DBNAME
          49 W2-DBNAME-TEXT    PIC X(24).
    *   ************************************************************
        10 W2-IXNAME OCCURS 100.
    *                IXNAME LENGTH
          49 W2-IXNAME-LEN     PIC S9(4) USAGE COMP SYNC.
    *                IXNAME
          49 W2-IXNAME-TEXT    PIC X(128).
    *   ************************************************************
        10 W2-IXCREATOR OCCURS 100.
    *                IXCREATOR LENGTH
          49 W2-IXCREATOR-LEN  PIC S9(4) USAGE COMP SYNC.
    *                IXCREATOR
          49 W2-IXCREATOR-TEXT
            PIC X(128).
```

Each 10 Level has an occurs

Each Length field has SYNC

Multi-Row Insert Code – Build the 100 occurrences

```
READ FD-FB-FILE
        AT END MOVE 'Y' TO W-END-OF-FILE
      END-READ.
PERFORM B000-PROCESS-RECORDS UNTIL END-OF-FILE.
B000-PROCESS-RECORDS SECTION.
   MOVE 0 TO W-NUM-ROWS.
   PERFORM VARYING W-NUM-ROWS FROM 1 BY 1
      UNTIL W-NUM-ROWS > 100
      OR END-OF-FILE
        MOVE FB-REC TO W-TPART
        MOVE W-PARTITION TO W2-PARTITION (W-NUM-ROWS)
        MOVE W-TSNAME TO W2-TSNAME (W-NUM-ROWS)
        MOVE W-DBNAME TO W2-DBNAME (W-NUM-ROWS)
        MOVE W-IXNAME TO W2-IXNAME (W-NUM-ROWS)
        MOVE W-IXCREATOR TO W2-IXCREATOR (W-NUM-ROWS)
         .......
       MOVE W-RELCREATED TO W2-RELCREATED (W-NUM-ROWS)
        READ FD-FB-FILE
          AT END MOVE 'Y' TO W-END-OF-FILE
        END-READ
     END-PERFORM.
```

Read the first record

Zeroise subscript

Loop for 100 records

Or end of file reached

Move record to 1st DCLGEN

Move 1st DCLGEN to

Next occurrence of

2nd DCLGEN

Read next record

Multi-Row INSERT Code – Insert the 100 rows

```
IF W-NUM-ROWS > 0
        SUBTRACT 1 FROM W-NUM-ROWS
        EXEC SQL
          INSERT INTO SYSTABLEPART
          VALUES
          (:W2-PARTITION ,
           :W2-TSNAME ,
           :W2-DBNAME ,
           :W2-IXNAME ,
           :W2-IXCREATOR ,
           :W2-PQTY ,
             ……..
           :W2-FORMAT ,
           :W2-REORG-LR-TS ,
           :W2-RELCREATED )
          FOR :W-NUM-ROWS ROWS
          NOT ATOMIC CONTINUE ON SQLEXCEPTION
        END-EXEC
```

Are there rows to insert

The host variables are the Names of the fields with the Occurs clause on but no Subscript is needed

Multi-row parameters

FOR :W-NUM-ROWS ROWS

- **FOR 100 ROWS could have been coded**
  - What if the file contained 226 records
  - First INSERT would insert 100 rows
  - Second INSERT would insert 100 rows
  - Third INSERT would insert 100 rows
  - So we now have 300 rows but only 226 records

  The last 74 would be the same as the last 74 of the second insert but using FOR :W-NUM-ROWS ROWS tells DB2 that the last Insert has 26 so no unexpected duplicates

NOT ATOMIC CONTINUE ON SQLEXCEPTION

- **ATOMIC** Specifies that if the insert for any row fails, all changes made to the database by any of the inserts, including changes made by successful inserts, are undone. This is the default.

- **NOT ATOMIC CONTINUE ON SQLEXCEPTION** Specifies that, regardless of the failure of any particular insert of a row, the INSERT statement will not undo any changes made to the database by the successful inserts of other rows, and inserting will be attempted for subsequent rows.

How to handle Errors in Single Row Insert Program

- **Add the following in B000 to cause an error**

```
IF W-ROW-COUNT = 10
   MOVE -10 TO W-DBNAME-LEN
END-IF.
```

- **This will stop the 11th record from being inserted**

Using DSNTIAR to Obtain the Message

Working Storage

```
01  W150-ERROR-MESSAGE.
    03  W150-ERR-LEN        PIC S9(4) COMP VALUE +288.
    03  W150-ERR-MSG-1      PIC X(72).
    03  W150-ERR-MSG-2      PIC X(72).
    03  W150-ERR-MSG-3      PIC X(72).
    03  W150-ERR-MSG-4      PIC X(72).
01  W150-ERROR-MESSAGE-LEN   PIC S9(9) COMP VALUE +72.
```

Total Length of following 4 fields

Length of each message line

Procedure Division

```
CALL 'DSNTIAR' USING SQLCA
                     W150-ERROR-MESSAGE
                     W150-ERROR-MESSAGE-LEN.

DISPLAY W150-ERR-MSG-1.
DISPLAY W150-ERR-MSG-2.
DISPLAY W150-ERR-MSG-3.
DISPLAY W150-ERR-MSG-4.
MOVE 12 TO RETURN-CODE.
GOBACK.
```

Output From Job after Error

DSNT408I SQLCODE = -311, ERROR: THE LENGTH OF INPUT HOST VARIABLE

NUMBER 3 IS NEGATIVE OR GREATER THAN THE MAXIMUM

DSNT418I SQLSTATE = 22501 SQLSTATE RETURN CODE

DSNT415I SQLERRP = DSNXRIHB SQL PROCEDURE DETECTING ERROR

This has been the recommended method

Of handling errors in programs using DB2

Put Same Error Into Multi-Row Insert

- **Add the following in B000 to cause an error**

MOVE -10 TO W2-DBNAME-LEN (11)

- **This will stop the 11th record from being inserted**

Job Output After Error in Multi-Row Insert

DSNT408I SQLCODE = -253, ERROR: A NON-ATOMIC INSERT STATEMENT SUCCESSFULLY COMPLETED FOR SOME OF THE REQUESTED ROWS, POSSIBLY WITH WARNINGS, AND ONE OR MORE ERRORS

DSNT418I SQLSTATE = 22529 SQLSTATE RETURN CODE

This is not very helpful as it only tells us that some worked

SQLERRD (3) can be used to find the number inserted

What Happens if the INSERT is Atomic

DSNT408I SQLCODE = -311, ERROR: THE LENGTH OF INPUT HOST VARIABLE

NUMBER 3 IS NEGATIVE OR GREATER THAN THE MAXIMUM

DSNT418I SQLSTATE = 22501 SQLSTATE RETURN CODE

DSNT415I SQLERRP = DSNXRIHB SQL PROCEDURE DETECTING ERROR

This message now tells what the real problem is but the 10 Inserted rows have been rolled back.

How can correct messages be issued and allow inserts to continue.

How to issue correct messages and allow inserts

- **NOT ATOMIC CONTINUE ON SQLEXCEPTION will allow processing to continue**
- **But the correct error can not be seen by calling DSNTIAR**
- **The solution is to use GET DIAGNOSTICS**

Coding GET DIAGNOSTICS

- **Working Storage**

```
03  W-DIAG-SUB           PIC S9(9) COMP.
03  W-DB2-RETURNED-SQLCODE PIC S9(9) COMP  VALUE 0.
03  W-DB2-RETURNED-SQLSTATE PIC X(5).
03  W-DB2-ROW-NUMBER       PIC S9(31) COMP-3.
03  W-DIAG-ERRORS         PIC S9(9) COMP.
01  W600-DIAG-AREA.
  10  W600-DIAGNOSTICS.
    49  W600-DIAGLEN       PIC S9(4) COMP VALUE 0.
    49  W600-DIAG          PIC X(32672).
```

These fields must be defined

Correctly or

GET DIAGNOSTICS

Will not work

- **Procedure Division**

```
EXEC SQL
  GET DIAGNOSTICS :W-DIAG-ERRORS = NUMBER
END-EXEC
IF W-DIAG-ERRORS > 0
  PERFORM VARYING W-DIAG-SUB FROM 1 BY 1
  UNTIL W-DIAG-SUB > W-DIAG-ERRORS
    EXEC SQL
      GET DIAGNOSTICS CONDITION :W-DIAG-SUB
      :W-DB2-RETURNED-SQLCODE = DB2_RETURNED_SQLCODE ,
      :W-DB2-RETURNED-SQLSTATE = RETURNED_SQLSTATE ,
      :W600-DIAGNOSTICS  = MESSAGE_TEXT ,
      :W-DB2-ROW-NUMBER     = DB2_ROW_NUMBER
    END-EXEC
```

This tells us how many errors have been found

Loop round so we get all errors

Get next error

Get SQLCODE

Get SQLSTATE

Get Message

Row number which caused this error

Results of DISPLAYING the Errors

- **SQLCODE -0311**
- **W-NUM-ROWS 100**
- **W-DIAG-ERRORS**
- **0000000001**
- **W-DB2-RETURNED-SQLCODE**
- **000000031J**
- **W-DB2-RETURNED-SQLSTATE**
- **22501**
- **W-DB2-ROW-NUMBER**
- **00000000000000000000000000000011**
- **THE LENGTH OF INPUT HOST VARIABLE NUMBER 3 IS NEGATIVE OR GREATER THAN THE MAXIMUM**

Multi-row FETCH

- Returns multiple rows on one API crossing
- "wide" cursor with locks on multiple rows
- Supports scrollable and non-scrollable, static and dynamic SQL
- Significant performance boost
- DSNTEP4 = DSNTEP2 + MRF

```
DECLARE C1 CURSOR
    WITH ROWSET POSITIONING
    FOR SELECT COL1, COL2 FROM T1;
OPEN C1;
FETCH FROM C1
    FOR :hv ROWS  INTO  :ARRAY1, :ARRAY2;
```


Single Row Fetch Example – PLFSR01

```
Working Storage
05 W-PARTITION          PIC S9(4) COMP.
05 W-TSNAME.
   49 W-TSNAME-LEN      PIC S9(4) COMP.
   49 W-TSNAME-TEXT     PIC X(24).
05 W-DBNAME.
   49 W-DBNAME-LEN      PIC S9(4) COMP.
   49 W-DBNAME-TEXT     PIC X(24).
05 W-IXNAME.
   49 W-IXNAME-LEN      PIC S9(4) COMP.
   49 W-IXNAME-TEXT     PIC X(128).
05 W-IXCREATOR.
   49 W-IXCREATOR-LEN   PIC S9(4) COMP.
   49 W-IXCREATOR-TEXT  PIC X(128).
```

Host Variables have been Hard coded rather than using DCLGEN Fields as Different installations have Different standard

```
Procedure Division
EXEC SQL
  DECLARE C1 SCROLL CURSOR FOR
  SELECT PARTITION, TSNAME, DBNAME, IXNAME, IXCREATOR
  FROM SYSTABLEPART
END-EXEC.
EXEC SQL
  OPEN C1
END-EXEC.
PERFORM B000-FETCH
UNTIL W-SQLCODE NOT = 0.
    B000-FETCH SECTION.
      EXEC SQL
        FETCH FROM C1
        INTO :W-PARTITION ,
             :W-TSNAME ,
             :W-DBNAME ,
             :W-IXNAME ,
             :W-IXCREATOR
      END-EXEC.
```

Declare the cursor using single row

Open the cursor

Fetch one row at a time

JCL to run PLFSRI01

```
//PLFSRI01 EXEC PGM=IKJEFT01,DYNAMNBR=25,ACCT=SHORT,
//           REGION=4096K
//STEPLIB  DD DSN=SYS2.DB2.V910.SDSNLOAD,DISP=SHR
//        DD DSN=SYS2.DB2.V910.SDSNEXIT.PIC,DISP=SHR
//        DD  DSN=FLETCHP.MASTER.LOAD,DISP=SHR
//SYSUDUMP DD  SYSOUT=*
//SYSTSPRT DD  SYSOUT=*
//SYSOUT   DD  SYSOUT=*
//SYSABOUT DD  SYSOUT=*
//SYSTSIN  DD *
DSN SYSTEM(PB1I)
RUN PROGRAM(PLFSRF01) PLAN(PLFSRF01)
END
/*
```

Multi-Row Fetch example – PLFMRI01

```
Working Storage
05 W-PARTITION          PIC S9(4) COMP OCCURS 100 TIMES.
05 W-TSNAME OCCURS 100 TIMES.
   49 W-TSNAME-LEN        PIC S9(4) COMP SYNC.
   49 W-TSNAME-TEXT       PIC X(24).
05 W-DBNAME OCCURS 100 TIMES.
   49 W-DBNAME-LEN        PIC S9(4) COMP SYNC.
   49 W-DBNAME-TEXT       PIC X(24).
05 W-IXNAME OCCURS 100 TIMES.
   49 W-IXNAME-LEN        PIC S9(4) COMP SYNC.
   49 W-IXNAME-TEXT       PIC X(128).
05 W-IXCREATOR OCCURS 100 TIMES.
   49 W-IXCREATOR-LEN     PIC S9(4) COMP SYNC.
   49 W-IXCREATOR-TEXT    PIC X(128).
Procedure Division
EXEC SQL
   DECLARE C1 SCROLL CURSOR WITH ROWSET POSITIONING FOR
   SELECT PARTITION, TSNAME, DBNAME, IXNAME, IXCREATOR
   FROM SYSTABLEPART
END-EXEC.
EXEC SQL
   OPEN C1
END-EXEC.
PERFORM B000-FETCH
UNTIL W-SQLCODE NOT = 0.
    B000-FETCH SECTION.
      EXEC SQL
        FETCH NEXT ROWSET FROM C1 FOR 100 ROWS
        INTO :W-PARTITION ,
             :W-TSNAME ,
             :W-DBNAME ,
             :W-IXNAME ,
             :W-IXCREATOR
      END-EXEC.
```

SYNC is required for length fields

WITH ROWSET POSTIONING is Required for multi-row

Tell DB2 how many rows to FETCH

Processing the rows returned by Multi-Row fetch

```
PERFORM VARYING W-SUB FROM 1 BY 1
UNTIL W-SUB > 100
   DISPLAY W-PARTITION (W-SUB) ' '
   W-TSNAME-TEXT (W-SUB) (1:W-TSNAME-LEN (W-SUB)) ' '
   W-DBNAME-TEXT (W-SUB) (1:W-DBNAME-LEN (W-SUB)) '',
   W-IXNAME-TEXT (W-SUB) (1:W-IXNAME-LEN (W-SUB)) ' '
   W-IXCREATOR-TEXT (W-SUB) (1:W-IXCREATOR-LEN (W-
  SUB))
END-PERFORM
```

Loop round in this case Displaying 100 rows returned

Processing Rows Returned

- **If the table has 226 rows**
- **First Fetch returns 100 – SQLCODE 0**
- **Second Fetch returns 100 – SQLCODE 0**
- **Third fetch can only return 26 – SQLCODE 100**

- **How can the program tell that 26 rows were returned on the last call?**

How to tell How many rows have been returned

- **SQLERRD (3) in SQLCA**

- **Or Use Get DIAGNOSTICS**

```
EXEC SQL
   GET DIAGNOSTICS
   :W-DIAG-ROW-COUNT = ROW_COUNT
END-EXEC
```

Processing last Rowset

```
PERFORM VARYING W-SUB FROM 1 BY 1
UNTIL W-SUB > W-DIAG-ROW-COUNT
   DISPLAY W-PARTITION (W-SUB) ' '
   W-TSNAME-TEXT (W-SUB) (1:W-TSNAME-LEN (W-SUB)) ' '
   W-DBNAME-TEXT (W-SUB) (1:W-DBNAME-LEN (W-SUB)) '“,
   W-IXNAME-TEXT (W-SUB) (1:W-IXNAME-LEN (W-SUB)) ' '
   W-IXCREATOR-TEXT (W-SUB) (1:W-IXCREATOR-LEN (W-SUB))
END-PERFORM.
```

Any Questions

- **Please contact**
- **Paul Fletcher**
- **FLETCHPL@UK.IBM.COM**