UNIT 5: HANDLING NULLS & ERRORS

SQLCA LAYOUT

SQL CA layout	REMARKS
SQLERRML BIN FIX(15) SQLERRMC CHAR (70)	Length of error description Short error description
SQLERRP CHAR(8) SQLERRD(6) BIN FIX(31)	Diagnostic info (Module name) Diagnostic into (Array) SQLERRD(3) : number of rows that were inserted, updated or deleted.
SQLWARNO CHAR(1) SQLWARN1 CHAR(1) SQLWARN2 CHAR(1) SQLWARN3 CHAR(1) SQLWARN4 CHAR(1)	If ' ' -> no warnings If 'w' -> warnings are present Truncation of a string NULL values eliminated from argument of a function. Number of cloumns is larger than number of host variables. Delete or update stmt does not include a WHERE clause
SQLAID CHAR(8) SQLCABC BIN FIX(31) SQLCODE BIN FIX(31)	'SQLCA' EYE CATCHER Length of SQLCA (136) Return code

Figure: 5.1 Handling NULLS and Errors

SQLCA LAYOUT

Error handling is based on information, DB2 returns in the SQLCA(SQL Communication Area). The most important fields are the SQLSTATE and SQLCODE.

Figure 5.2 SQLCA layouts

SQLSTATE

- **SQLSTATE is an SQL return code contained in a 5 digit character string.**

First digit : zero (successful) or not Zero

First 2 digits : SQLSTATE "CLASS"
- 01 : unqualified successful execution
- 02 : warning
- 03 : warning – no data

last 3 digits represent SQLSTATE "SUBCODE"

Example :

SQLSTATE 00000 : successful
SQLSTATE 01501 : string truncation
SQLSTATE 02000 : no more data

Figure: 5.3 SQLSTATE

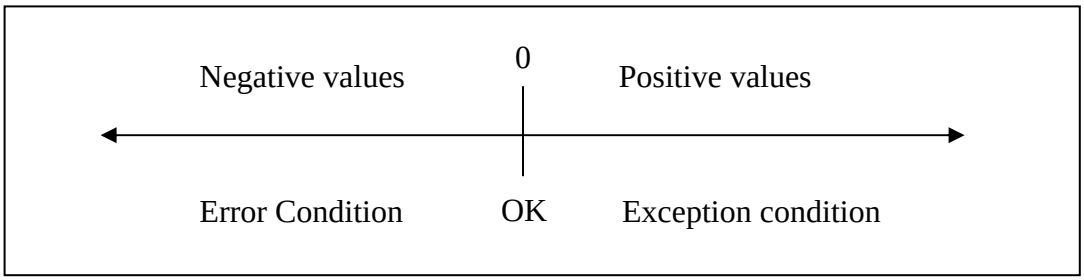
Notes:

SQLSTATE is a return code that is common to all DB2 family products. If you base your error handling on SQLSTATE, your interpretation of the error situation will be independent of the platform where the SQL statement was actually executed.

It is recommended, therefore to use SQLSTATE instead of SQLCODE.

SQLCA CODES

- **SQLCODE is the SQL return code for DB2 (on MVS) and contains a signed numeric value.**



Examples :

-084 Unacceptable SQL statement
000 Successful execution warning messages may have been issued.
+100 Row not found for FETCH, UPDATE or DELETE of the result of a query is an empty table.

Figure: 5.4 SQLCODE

Notes:

SQLCODE is a return code that is specific to DB2 for MVS. Other platforms, like DB2/ VM also have SQLCODEs. However, the values don't always match.

SQLCA CODES USAGE

- **Status information is given by :**
  - **SQLCODE, SQLSTATE and**
  - **SQLWARNO**
  - **SQLWARN1 – SQLWARNA**

CONDITION	INTEGER SQLCODE	Char(5) SQLSTATE	Char(1) SQLWARNO	REQUEST STATUS
Error	< 0	¬ = 00000 ¬ = 01ddd ¬ = 02000		Failed
Warning	> 0 & ¬ = 100	01dddd	Or ‘w’	Satisfied, with special conditions
Not found	+100	02000		(more) data not found
Success	0	00000	And ‘ ‘	Success

Figure 5.5 SQLCA codes Usage

Notes:

The chart gives you a general overview of which return codes will be generated in the indicated warning or error situation.

ERROR HANDLING

Example of testing return codes :

```
EXEC SQL SELECT……

IF SQLSTATE ¬ = '00000'
THEN IF SQLSTATE = '02000'
        THEN DISPLAY 'RECORD NOT FOUND'
           ELSE IF SUBSTR (SQLSTATE,1, 2) = '01'
                   THEN DISPLAY 'WARNSQL'
                ELSE DISPLAY 'ERRORSQL'
ELSE
        :
        :
```

Example of using WHENEVER

```
EXEC SQL WHENEVER NOT FOUND GOTO NO MORE
EXEC SQL WHENEVER SQLERROR  GOTO
ERRORSQL
EXEC SQL WHENEVER SQLWARNING GOTO
WARNSQL
        :
        :
        :
EXEC SQL SELECT…..
```

Figure: 5.6 Error Handling

ERROR HANDLING (Cont…)

Various techniques can be used to detect error conditions. One thing is sure : you need error handling. The first technique is to react individually after every SQL statement. A more general error routine should be developed and called after each SQL statement.

The second technique, WHENEVER, is not recommended for general use.

Figure: 5.7 Error Handling (Cont…)

SCOPE OF WHENEVER

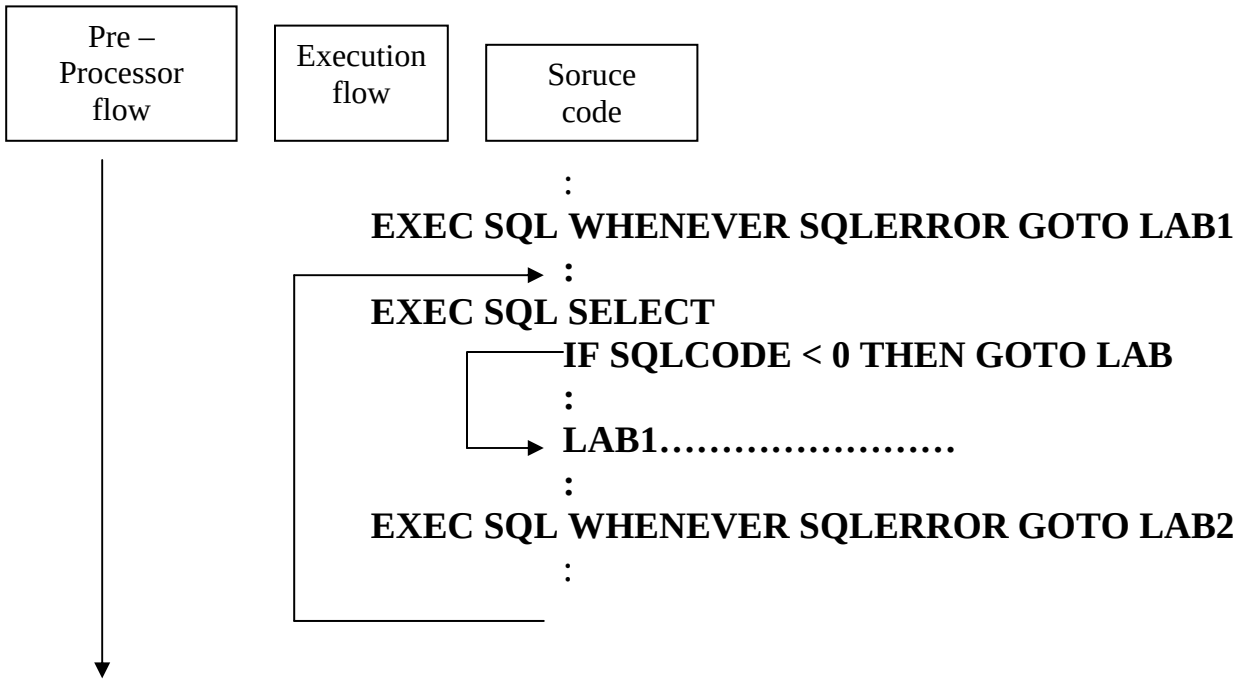


Figure: 5.8 Scope of WHENEVER

Notes:

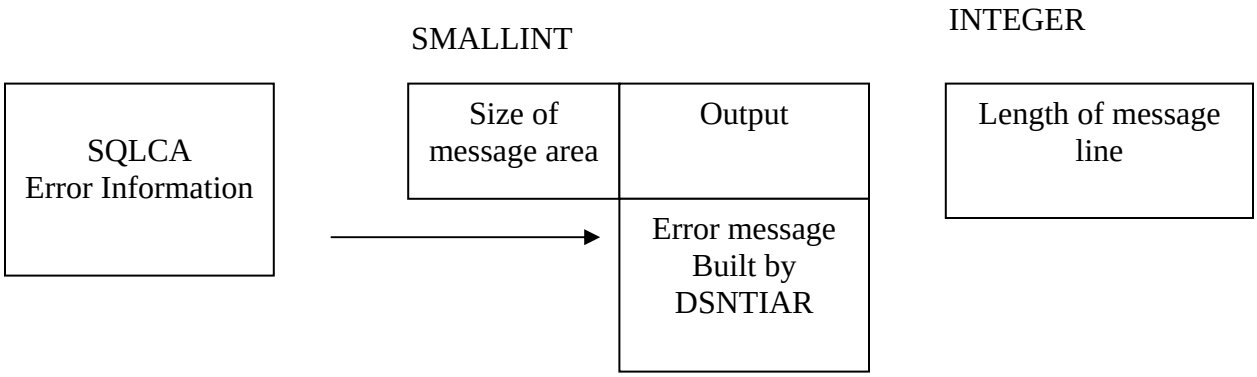
As you can see, the WHENEVER statement forces you to use GOTO and it is, therefore, sensitive to where the WHENEVER is placed in the code.

Going back to where you came from is difficult. It is also difficult to take specific actions for specific SQL statements (an error for an update might need another reaction than the same error for a select).

WHENEVER actually adds the checks to each of your statements at precompile.

ERROR MESSAGE FORMATTING ROUTINE

- **ERROR Message Formatting Routine :**
  - **CALL DSNTIAR**
    And pass it three areas :



- FORTRAN uses routine DSNTIR

Figure: 5.9 Error Message formatting Routine

Notes:

Attention should be paid **not to invoke DSNTIAR unless an error condition is detected**, because the module is dynamically loaded into storage when invoked. If you do this without care, you will waste a lot of resources.

ERROR HANDLING - GENERAL

- **WHENEVER will in general not be used**
  - **Different treatment needed for ‘ NOT FOUND’ after delete/ update/ insert.**
  - **Coming back after WHENEVER**
  - **Might want to intercept other return codes**

- **SYSADM will normally provide a STANDARD error routine for everyone to use**

- **Only CALL DSNTIAR in error condition.**

Figure: 5.10 Error Handling – General

Notes:

As mentioned previously, the use of WHENEVER is not recommended, and we advise using a general error routine that is called after each SQL statement.

RETRIEVING NULL VALUES

:REF1:REF2

- **Indicator Variable (:REF2) should always be provided if selected column allows NULL**

```
01  P-PHONE     PIC X(4)                  /*H-VARIABLE */

01  P-PHONE-I   PIC S9(4) COMP            /*I-VARIABLE*/

EXEC SQL SELECT PHONENO
                INTO  :P-PHONE : P-PHONE-I
                FROM EMP
                WHERE EMPNO = :INP-EMP
END-EXEC.
```

After the SELECT

- **If PHONENO contains the NULL value :**
  H-variable P-PHONE -> unchanged
  I- variable P-PHONE -> negative
- **If PHONENO contains data** :
  H-variable P-PHONE -> updated
  I- variable P-PHONE -> not negative

Figure: 5.11 Retrieving NULL Values

Notes :

If a column can contain the NULL value, a special host variable called an “Indicator Variable” should be defined in the program. This variable is used by DB2 to signal the existence of a NULL value for that column.

INSERTING NULL VALUES

Definition of Host Variables :

```
01 PEMPL.
        02 EMPNO          PIC  X(06).
        02 FIRSTNME       PIC  X(12).
        02 MIDINIT        PIC  X(01).
        02 LASTNAME       PIC  X(15).
        02 WORKDEPT       PIC  X(03).
        02 PHONENO        PIC  X(04).
```

Definition of Indicator Variable :

```
01  PEMPL.
    10  INDSTRUC PIC S9(4) COMP OCCURS 6 TIMES
```

Set Indicator Variable :

```
IF no_phone_number_exists, THEN
        NPHONE = -1.
```

Imbedded SQL :

```
EXEC SQL
        UPDATE PEMPL
        SET PHONENO = : PHONENO : PHONENO
        WHERE EMPNO = :EMPNO
END-EXEC.
```

Figure: 5.12 Inserting NULL values

Notes:

On the other hand, the indicator variable will allow you to pass a NULL value to DB2.

SELECTING NULL VALUES

```
IF null_value_wanted, THEN
                NPHONE = -1.

EXEC SQL SELECT LASTNAME, FIRSTNME
                        FROM PEMPL
                        WHERE PHONENO =:NPHONE
END-EXEC.
```

If the indicator variable NPHONE is set to a negative value, the statement is treated as

```
EXEC SQL SELECT LASTNAME, FIRSTNME
                FROM PEMPL
                WHERE PHONENO IS NULL
END-EXEC.
```

. . . . . and the contents of the host variable PHONENO is ignored.

Figure: 5.13 Selecting NULL Values

Notes:

A negative value set in the indicator variable indicates a NULL value for the associated host variable.

HOST STRUCTURE – INDICATOR VARIABLE

Definition of host variable (Structure) :

```
01 PEMPL.
        02 EMPNO          PIC   X(06).
        02 FIRSTNME       PIC   X(12).
        02 MIDINIT        PIC   X(01).
        02 LASTNAME       PIC   X(15).
        02 WORKDEPT       PIC  X(03).
        02 PHONENO        PIC   X(04).
```

Definition of Indicator Variable (array) :

```
01 PEMPL.
   10 INDSTRUC  PIC  S9(4) COMP OCCURS 6 TIMES.
```

Imbedded SQL :

```
EXEC SQL SELECT *
                INTO  :PEMPL :INDSTRUC
                FROM PEMPL
                WHERE EMPNO = :EMPNO
END-EXEC.
```

Test for NULL Values :

```
IF INDSTRUC(5) < 0 THEN
                 WORKDEPT  = '????' THEN
IF INDSTRUC(6) < 0 THEN
                 PHONENO = 'UNKN'
```

Figure: 5.14 Host Structures – Indicator Variables

Notes :

HOST STRUCTURE – INDICATOR VARIABLE (Cont...)

The example shows how a simple array can used to detect the NULL values for several columns.

Remember that the DCLGEN function can generate the indicator variables for your DB2 table.

Figure: 5.15 Host Structures – Indicator Variables (Cont..)

HANDLING ARITHMETIC ERRORS

- **Arithmetic error exceptions will be tolerated, if error occurs in the select list of the 'Outer' SELECT and an indicator variable**
    - **Result of expression is NULL**
    - **Indicator variable is set to -2**
    - **SQLSTATE 01519 returned in the SQLCA (SQLCODE +802)**
    - **Value in host variable left unchanged**
    - **Processing continues**
    - **Expressions and values not in error will be returned.**
- **Otherwise statement execution will be halted and SQLSTATE 22013 OR 22003 (SQLCODE –802) WILL BE RETURNED**

Figure: 5.16 Handling Arithmetic Error

Notes:

Indicator variables are also used by DB2 to signal arithmetic error conditions.

HANDLING CONVERSION ERROR

- **If a conversion error occurs in an embedded SELECT OR FETCH statement and an indicator variable has been provided for the expression.**
    - **Result of expression is NULL**
    - **Indicator variable is set to -2**
    - **SQLSTATE 01515 returned in the SQLCA (SQLCODE =-304)**
    - **Value in host variable left unchanged**
    - **Processing continues**
    - **Expressions and values not in error will be returned**
- **Otherwise statement execution will be halted and SQLSTATE 22003 (SQLCODE –304) will be returned**

Figure: 5.17 Handling Conversion Error

Notes:

Indicator Variable are also used by DB2 to signal conversion errors.

ARITHMETIC AND CONVERSION ERROR EXAMPLES

Example Background :

Declaration In program

```
01      MV1   PIC   S9(4) COMP.
01      MV2   PIC   S9(4) COMP.
01      MV3   PIC   S9(4) COMP.
01      MV11  PIC   S9(4) COMP.
01      MV21  PIC   S9(4) COMP.
01      MV31  PIC   S9(4) COMP.
01      IV1   PIC   S9(4) COMP.   /* Indicator Variable */
01      IV2   PIC   S9(4) COMP.   /*Indicator Variable */
01      IV3   PIC   S9(4) COMP.   /* Indicator Variable */
```

TABLE A values are

SM1	INT1	SMI2
1	123458	1
2	345678	0
3	123678	1
4	211111111	4

Figure: 5.18 Arithmetic And Conversion Error Examples

Notes :

Note the contents of the table and program definitions used for the coming examples.

EXAMPLE OF ARITHMETIC ERROR

```
EXEC SQL
            SELECT SMI1,SMI1*INTI, SMI1/SMI2
            INTO :MV1 : IV1, :MV2:IV2, :MV3:IV3
            FROM TABLEA
END-EXEC.
```

Result for first row :
MV1 = 1 IV1 = 0
MV2 = 123458 IV2 = 0
MV3 = 1 IV3 = 0
SQLSTATE = 00000

Result for Sec row :
MV1 = 2 IV1 = 0
MV2 = 691356 IV2 = 0
MV3 = 2/0 IV3 = -2 ←
SQLSTATE = 01519

Result for third row :
MV1 = 3 IV1 = 0
MV2 = 371034 IV2 = 0
MV3 = 3 IV3 = 0
SQLSTATE = 00000

Result for forth row :
MV1 = 4 IV1 = 0
MV2 = 844444444 IV2 = -2 ←
MV3 = 1 IV3 = 0
SQLSTATE = 01519

Figure: 5.19 Example of Arithmetic Error

Notes:

INDICATOR VARIABLE VALUES

‘0’	**Zero Denotes That Value In Data Variable Area Is Not NULL**
‘+n’	**any positive number indicates truncation of a string value, where 'n’ is the length of the original string (string field only)**
‘-n’	**any negative number indicates a NULL value in the data variable.**
‘-2’	**Negative 2 Indicates a NULL value in the data variable because of either a numeric conversion error or an arithmetic expression error**

Figure: 5.20 Indicator Variable Values

Notes:

An error handling routine could take specific actions for these error conditions. Error handling will generally be done through a standard routine that will be invoked after each SQL statement. Your DB2 or SYSADM will provide you with the details about the specific implementation chosen for your installation

OTHER ERROR CONDITIONS

TABLE CHECK Constraint

22513 / - 545
statement not executed

VIEW CHECK option

23501 / - 161
statement not executed

DATE Invalid

22007 / -180 0r -181
statement not executed

Figure: 5.21 Other Error Condition

Notes:

A “smart” program could also handle those error condition. ABENDing the program is not always needed.