NDB - CALLDBPROC

The CALLDBPROC statement is used to call DB2 stored procedures. It supports the result set mechanism of DB2 and it enables you to call DB2 stored procedures.

This section covers the following topics:

- Static and Dynamic Execution
- Result Sets
- List of Parameter Data Types
- CALLMODE=NATURAL
- Example of CALLDBPROC/READ RESULT SET

Further details and syntax: CALLDBPROC in Natural SQL Statements in the Natural Statements documentation.

Static and Dynamic Execution

If the CALLDBPROC statement is executed dynamically, all parameters and constants are mapped to the variables of the following DB2 SQL statement:

```
CALL :hv USING  DESCRIPTOR :sqlda  statement
```

:hv denotes a host variable containing the name of the procedure to be called and *:sqlda* is a dynamically generated sqlda describing the parameters to be passed to the stored procedure.

If the CALLDBPROC statement is executed statically, the constants of the CALLDBPROC statement are also generated as constants in the generated assembler SQL source for the DB2 precompiler.

Result Sets

If the SQLCODE created by the CALL statement indicates that there are result sets (SQLCODE +466 and +464), Natural for DB2 runtime executes a

```
DESCRIBE PROCEDURE :hv INTO :sqlda
```

statement in order to retrieve the result set locator values of the result sets created by the invoked stored procedure. These values are put into the RESULT SETS variables specified in the CALLDBPROC statement. Each RESULT SETS variable specified in a CALLDBPROC for which no result set locator value is present is reset to zero. The result set locator values can be used to read the result sets by means of the READ RESULT SET statement as long as the database transaction which created the result set has not yet issued a COMMIT or ROLLBACK.

If the result set was created by a cursor WITH HOLD, the result set locator value remains valid after a COMMIT operation.

Unlike other Natural SQL statements, CALLDBPROC enables you (optionally!) to specify a SQLCODE variable following the GIVING keyword which will contain the SQLCODE of the underlying CALL statement. If GIVING is specified, it is up to the Natural program to react on the SQLCODE (error message NAT3700 is not issued by the runtime).

List of Parameter Data Types

Below are the parameter data types supported by the CALLDBPROC statement:

Natural Format/Length	DB2 Data Type
A*n*	CHAR(*n*)
B2	SMALLINT
B4	INT
B*n* (*n* = not equal 2 or 4)	CHAR(*n*)
F4	REAL
F8	DOUBLE PRECISION
I2	SMALLINT
I4	INT
N*nn.m*	NUMERIC(*nn+m,m*)
P*nn.m*	NUMERIC(*nn+m,n*)
G*n*	GRAPHIC(*n*)
A*n*/1:m	VARCHAR(*n***m*)
D	DATE
T	TIME (see also TIME below)

TIME

The Natural format **T** has a wider data range than the equivalent DB2 TIME data type. Compared with DB2 TIME, in addition, the Natural **T** variable has a date fraction (year, month, day) and the tenths of a second.

As a result, converting a Natural **T** variable into a DB2 TIME value, NDB cuts off the date fraction and the tenths of a second part. Converting DB2 TIME into Natural **T**, the date fraction is reset to 0000-01-02 and the tenths of a second part is reset to 0 in Natural.

CALLMODE=NATURAL

This parameter is used to invoke Natural stored procedures defined with PARAMETER STYLE GENERAL/WITH NULL.

If the CALLMODE=NATURAL parameter is specified, an additional parameter describing the parameters passed to the Natural stored procedure is passed from the client, i.e. caller, to the server, i.e. the NDB server stub. The parameter is the Stored Procedure Control Block (STCB; see also STCB Layout in PARAMETER STYLE in the section Natural Stored Procedures and UDFs) and has the format VARCHAR from the viewpoint of DB2. Therefore, every Natural stored procedure defined with PARAMETER STYLE GENERAL/WITH NULL has to be defined with the CREATE PROCEDURE statement by using this VARCHAR parameter as the first in its PARMLIST row.

From the viewpoint of the caller, i.e. the Natural program, and from the viewpoint of the stored procedure, i.e. Natural subprogram, the STCB is invisible. It is passed as first parameter by the Natural for DB2 runtime and it is used as on the server side to build the copy of the passed data in the Natural thread and the corresponding CALLNAT statement. Additionally, this parameter serves as a container for error information created during execution of the Natural stored procedure by the Natural runtime. It also contains information on the library where you are logged on and the Natural subprogram to be invoked.

Example of CALLDBPROC/READ RESULT SET

Below is an example program for issuing CALLDBPROC and READ RESULT SET statements:

```
DEFINE DATA LOCAL
  1 ALPHA       (A8)
  1 NUMERIC     (N7.3)
  1 PACKED      (P9.4)
  1 VCHAR       (A20/1:5) INIT     <'DB25SGCP'>
  1 INTEGER2    (I2)
  1 INTEGER4    (I4)
  1 BINARY2     (B2)
  1 BINARY4     (B4)
  1 BINARY12    (B12)
  1 FLOAT4      (F4)
  1 FLOAT8      (F8)
  1 INDEX-ARRAY (I2/1:11)
  1 INDEX-ARRAY1(I2)
  1 INDEX-ARRAY2(I2)
  1 INDEX-ARRAY3(I2)
  1 INDEX-ARRAY4(I2)
  1 INDEX-ARRAY5(I2)
  1 INDEX-ARRAY6(I2)
  1 INDEX-ARRAY7(I2)
  1 INDEX-ARRAY8(I2)
  1 INDEX-ARRAY9(I2)
  1 INDEX-ARRAY10(I2)
  1 INDEX-ARRAY11(I2)
  1 #RESP        (I4)
  1 #RS1         (I4) INIT <99>
  1 #RS2         (I4) INIT <99>
  LOCAL
  1 V1 VIEW OF SYSIBM-SYSTABLES
  2 NAME
  1 V2 VIEW OF SYSIBM-SYSPROCEDURES
  2 PROCEDURE
  2 RESULT_SETS
  1 V (I2) INIT <99>
  END-DEFINE
  CALLDBPROC 'DAEFDB25.SYSPROC.SNGSTPC'  DSN8510-EMP
   ALPHA  INDICATOR :INDEX-ARRAY1
   NUMERIC INDICATOR :INDEX-ARRAY2
   PACKED  INDICATOR :INDEX-ARRAY3
```

```
  VCHAR(*) INDICATOR :INDEX-ARRAY4
  INTEGER2 INDICATOR :INDEX-ARRAY5
  INTEGER4 INDICATOR :INDEX-ARRAY6
  BINARY2  INDICATOR :INDEX-ARRAY7
  BINARY4  INDICATOR :INDEX-ARRAY8
  BINARY12 INDICATOR :INDEX-ARRAY9
  FLOAT4   INDICATOR :INDEX-ARRAY10
  FLOAT8   INDICATOR :INDEX-ARRAY11
   RESULT SETS #RS1 #RS2
  CALLMODE=NATURAL
READ (10) RESULT SET #RS2 INTO VIEW V2 FROM SYSIBM-SYSTABLES
WRITE 'PROC F RS  :' PROCEDURE 50T RESULT_SETS
END-RESULT
END
```