Writing a Natural Stored Procedure

This section provides a general guideline of how to write a Natural Stored Procedure and what to consider when writing it.

To write a Natural stored procedure

1. Determine the format and attributes of the parameters that are passed between the caller and the stored procedure. Consider creating a Natural PDA (parameter data area). Stored procedures do not support data groups and redefinition within their parameters.
2. Determine the PARAMETER STYLE of the stored procedure: GENERAL, GENERAL WITH NULL or DB2SQL.
   - If you use GENERAL WITH NULL, append the parameters to the Natural stored procedure by defining a NULL indicator array that contains a NULL indicator (I2) for each other parameter.
   - If you use DB2SQL, append the parameters of the Natural stored procedure by defining NULL indicators (one for each parameter), include the PDA DB2SQL_P and the PDA DBINFO_P (only with DBINFO specified), if desired. See also the relevant DB2 literature by IBM.
3. Decide which and how many result sets the stored procedure will return to the caller.
4. Code your stored procedure as a Natural subprogram.
   - **Returning result sets**
     To return result sets, code a SELECT statement with the WITH RETURN option.

     To return the whole result set, code an ESCAPE BOTTOM immediately after the SELECT.

     To return part of the result set code, an IF *COUNTER = 1 ESCAPE TOP END-IF immediately following the SELECT statement. This ensures that you do not process the first empty row that is returned by the SELECT WITH RETURN statement. To stop row processing, execute an ESCAPE BOTTOM statement.

     If you do not leave the processing loop initiated by the SELECT WITH RETURN via ESCAPE BOTTOM, the result set created is closed and nothing is returned to the caller.
   - **Attention when accessing other databases**
     You can access other databases (for instance Adabas) within a Natural stored procedure. However, keep in mind that your access to other databases is synchronized neither with the updates done by the caller of the stored procedure, nor with the updates done against DB2 within the stored procedure.
   - **NDB handling of COMMIT and ROLLBACK statements**
     DB2 does not allow a stored procedure to issue COMMIT or ROLLBACK statements (the execution of those statements puts the caller into a must-rollback state). Therefore, the NDB runtime handles those statements as follows when they are issued from a stored procedure:

     COMMIT against DB2 will be skipped. This allows the stored procedure to commit Adabas updates without getting a must-rollback state from DB2.

ROLLBACK against DB2 will be skipped if it is created by Natural itself.

ROLLBACK against DB2 will be executed if it is user-programmed. Thus, after a Natural error, the caller receives the Natural error information and not the unqualified must-rollback state. Additionally, this function ensures that, if the user program backs out the transaction, every database transaction of the stored procedure is backed out.

5. **For DB2 UDB:** Issue a CREATE PROCEDURE statement that defines your stored procedure, for example:

```
CREATE PROCEDURE <PROCEDURE>
  (INOUT  STCB       VARCHAR(274+13*N),
   INOUT  <PARM1>    <FORMAT>,
   INOUT  <PARM2>    <FORMAT>,
   INOUT  <PARM3>    <FORMAT>
   .
  )
  DYNAMIC RESULT SET <RESULT_SETS>
  EXTERNAL NAME <LOADMOD>
  LANGUAGE ASSEMBLE
  PROGRAM TYPE <PGM_TYPE>
  PARAMETER STYLE GENERAL <WITH NULLS depending on LINKAGE>;
```

   The data specified in angle brackets (< >) correspond to the data listed in the table above, PARM1 - PARM3 and FORMAT depend on the call parameter list of the stored procedure. See also Example Stored Procedure NDBPURGN, Member CR6PURGN.

6. Code your Natural program invoking the stored procedure via the CALLDBPROC statement.

   Code the parameters in the CALLDBPROC statement in the same sequence as they are specified in the stored procedure. Define the parameters in the calling program in a format that is compatible with the format defined in the stored procedure.

   If you use result sets, specify a RESULT SETS clause in the CALLDBPROC statement followed by a number of result set locator variables of FORMAT (I4). The number of result set locator variables should be the same as the number or result sets created by the stored procedure. If you specify fewer than are created, some result sets are lost. If you specify more than are created, the remaining result set locator variables are lost. The sequence of locator variables corresponds to the sequence in which the result sets are created by the stored procedure.

   Keep in mind that the fields into which the result set rows are read have to correspond to the fields used in the SELECT WITH RETURN statement that created the result set.