NDB - Writing a Natural UDF

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

See also the section Writing a Natural Stored Procedure.

To write a Natural UDF

1. Determine the format and attributes of the parameters, which are passed between the caller and the stored procedure.
2. Create a Natural parameter data area (PDA).
3. Append the parameter definitions of the Natural UDF by defining NULL indicators (one for each parameter) and include the PDA DB2SQL_P.
4. If required, code a SCRATCHPAD area in the parameter list.
5. If required, code a call-type parameter. If you have specified DBINFO, include the PDA DBINFO_P. See also the relevant DB2 literature by IBM.
6. Code your UDF as a Natural subprogram and consider the following:

   - **Attention when accessing other databases**
     You can access other databases (for example, Adabas) within a Natural UDF. 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 these statements results in a must-rollback state. If a Natural stored procedure issues a COMMIT or ROLLBACK, the NDB runtime processes these statements as follows:

     COMMIT against DB2 is skipped. This allows the stored procedure to commit Adabas updates without entering a must-rollback state by DB2.

     ROLLBACK against DB2 is skipped if it is implicitly issued by the Natural runtime.

     ROLLBACK against DB2 is executed if it is user-programmed. Thus, after a Natural error, the caller receives a corresponding Natural error message text, but does not enter an unqualified must-rollback state. Additionally, this reaction ensures that every database transaction the stored procedure performs is backed out if the user program backs out the transaction.

7. Issue a CREATE FUNCTION statement that defines your UDF, for example:

```
CREATE FUNCTION <FUNCTION>
  ([PARM1]    <FORMAT>,
   [PARM2]    <FORMAT>,
   [PARM3]    <FORMAT>

  )
  RETURNS <FORMAT>
```

NDB - Writing a Natural UDF

```
  EXTERNAL NAME <LOADMOD>
  LANGUAGE ASSEMBLE
  PROGRAM TYPE <PGM TYPE>
  PARAMETER STYLE DB2SQL
.
.
.;
```

In the example above, the variable data are enclosed in angle brackets (< >) and refer to the keywords preceding the brackets. Specify a valid value, for example:

LOADMOD denotes the NDB server stub module, for example, NDB41SRV. PARM1 - PARM3 and FORMAT relate to the call parameter list of the UDF. See also the Example UDF.

8. Code a Natural program containing SQL statements that invoke the UDF.

   Specify the parameters of the Natural UDF invocation in the same sequence as specified in the Natural UDF definition. The format of the parameters in the calling program must be compatible with the format defined in the Natural UDF.