Using SQL in REXX/VM

REXX is an extremely versatile language. It offers efficient programming structures, powerful functions and extensive mathematical capabilities. Commands to host environments can be freely intermixed with REXX statements. This makes the language particularly suitable for command procedures, application prototyping or TCP/IP connectivity.

REXXSQL, a product developed by Software Product Research, provides an SQL interface for REXX/VM procedures.

REXXSQL Calls

REXXSQL is implemented as an external REXX routine, invoked using the call "**REXXSQL input_argument".**

The input argument is a character string, a REXX variable or expression. It contains or refers to the SQL statement to be executed.

After submitting the statement to DB2, REXXSQL returns the completion status and the execution results to the invoking procedure as REXX variables or stems.

The completion status includes:

- the DB2 SQLCODE and other meaningful fields of the SQLCA
- the number of rows processed by the statement in the REXXSQL variable **_nrows**
- the estimated statement cost in the REXXSQL variable **_cost**

If a SELECT was submitted, the fetched rows are returned in REXX stem variables. Each selected column is passed in a REXX stem with a name equal to the table columnname.

All SQL statements valid in an application program can be submitted to REXXSQL. This includes CONNECT, SELECT, UPDATE, DELETE, INSERT, COMMIT and DDL statements like CREATE or DROP.

In dynamic mode, these SQL statements may be issued against any DB2 platform that can be reached from DB2/VM-VSE.

Static REXXSQL requests can be used against DB2/VM-VSE databases only, as the extended dynamic execution mode implied, is unknown in DB2 platforms other than DB2/VM-VSE.

REXXSQL provides additional interfaces:

- to issue DB2 operator commands
- to obtain DB2 help for a given SQLCODE

Dynamic SQL statements

The dynamic REXXSQL mode is the simplest interface to DB2. Input to the call is a character string or a REXX expression that contains the SQL statement to be executed. If a SELECT is submitted, the fetched columns are returned in REXX stems.

Dynamic FETCH interface

A SELECT statement that returns a large number of rows may need considerable amounts of storage for the column stems. Therefore, a FETCH interface has been designed to select one table row at a time. The interface is opened with a REXXSQL OPEN call. Each REXXSQL FETCH call transfers a single table row. A REXXSQL CLOSE call terminates the fetch sequence.

Static SQL statements

While the dynamic interface is easy to use, it incurs the overhead of DB2 prepare processing. Since this overhead is not trivial, SQL statements that are executed often can be prepped and executed in the static SQL mode.

Prepping SQL statements

A REXXSQL procedure can create DB2 packages, containing multiple SQL statements. Package creation is initiated using a REXXSQL CREATE PACKAGE call.

Each SQL statement is added to the new package by means of a REXXSQL PREPARE call. The statement is identified by a name that will be used as a reference to the statement, when executing the package. REXXSQL uses a package control table to store the relationship between the statement name and the corresponding package section number, assigned by DB2.

Using REXX/SQL for stored procedures

REXX/SQL provides the **REXXPP** interface that allows a REXX/SQL exec to be used as a stored procedure. In addition, the REXX/SQL **CALL** statement allows a REXX/SQL exec to call a stored procedure.

Executing prepped SQL statements

The **REXXSQL EXECUTE** call executes a named statement in a named package. If the prepped statement contains parameter markers or host variables, the substitution data is passed in the USING clause of the EXECUTE call. After execution, a number of status variables are available as for a dynamic execution (SQLCODE, _COST, _NROWS). If the executed statement is a SELECT, the selected columns are returned in REXX stems. REXX programs can execute statements from different packages within the same LUW.

Like the dynamic interface, the static interface allows to fetch single rows. Since each fetch sequence is identified by a statement name (which corresponds to an open cursor), multiple FETCH sequences can be open concurrently.

Sample REXX/SQL Procedure List DB2 tables by rowcount

```
/*Connect the database */
"REXXSQL CONNECT" user "IDENTIFIED BY" password "TO" db
if SQLCODE < 0 then /* process error */

/* Create the statement "get_rowcount" in package SAMPLE */
"REXXSQL CREATE PACKAGE SAMPLE"
s = "SELECT CREATOR,TNAME,ROWCOUNT FROM SYSTEM.SYSCATALOG" ,
    "WHERE ROWCOUNT >= :INTEGER ORDER BY ROWCOUNT DESC"
"REXXSQL PREPARE GET_ROWCOUNT FROM" s
"REXXSQL COMMIT"

/* Execute the prepped statement "get_rowcount" */
"REXXSQL EXECUTE GET_ROWCOUNT IN SAMPLE USING 1000"
if SQLCODE < 0 then /* process error */

/* Show results */
do i = 1 to _NROWS
   tablename = strip(creator.i)"."strip(tname.i)
   say "Table" tablename "has" rowcount.i "rows"
end
"REXXSQL COMMIT"
exit
```