RXSQL - SQL for REXX

iX Corporation
575 W Madison #3610
Chicago IL 60661

Contents

List of Tables

CHAPTER 1

Overview

Introduction

RXSQL provides an interface between uni-REXX from The Workstation Group and OpenREXX from iX Corporation with SQL databases. Currently both the Sybase and Oracle relational database management systems (RDBMS's) are supported.

iX Corporation sells OpenREXX for use as an embedded scripting language on all major computing platforms. OpenREXX and uni-REXX share the same basic implementation.

The best way to understand RXSQL is to first review the example at the end of this manual. See "Example" on page 25.

This manual is intended for an audience that knows both REXX and SQL.

RXSQL is compatible with the IBM SQL/DS product also called RXSQL (5798-DXT) described in manual SH20-7051.

For further information about OpenREXX or RXSQL, please contact:

Neil Milsted
iX Corporation
575 W Madison #3610
Chicago IL 60661
(312)902-2149
Internet: 76050.3673@compuserve.com
Compuserve: 76050,3673

CHAPTER 2

Commands

Introduction

The RXSQL interface consists of the commands described in this chapter.

Return codes

RXSQL commands can return the following return code (in the REXX special variable rc):

TABLE 1. RXSQL return codes

Return code	Description
0	Fine.
4	End of data (usually from `RXSQL FETCH`).
8	An error occurred in the underlying RDBMS (currently Oracle or Sybase). For more information “SQLERRM” on page 23 and “SQLCODE” on page 23.
50+	An error occurred in RXSQL processing. For more information see“RXSQLMSG” on page 19.

Cursors and Cursor State

Not all RXSQL commands will work at all times. For example, an `RXSQL CONNECT` must be issued for any processing of tables to occur.

Typically, after a connection is made a “cursor” will be prepared (i.e. created) by associating it with an SQL `DELETE`, `INSERT`, `SELECT` or `UPDATE` statement with the `RXSQL PREP` command. After the cursor is prepared it may be opened

for fetch processing with RXSQL OPEN and the actual fetching done with RXSQL FETCH and finally closed with RXSQL CLOSE.

To keep track of what operations are currently valid, two state values are kept for each cursor. These values may be examined with the RXSQL STATE command which returns the variable SQLSTATE in the form “s1 s2” (for states 1 and 2 respectively). The first state value is 0 if the cursor could not be prepared (i.e. RXSQL PREP failed), or it’s 1 if the cursor has been prepared. The second state variable is 1 if the cursor has been prepared, 2 if the cursor has been opened, and 0 (meaning unprepared) after a rollback or commit has occurred. Note that if the second state is 0, the cursor can still be opened. For a summary of cursor states see “SQLSTATE” on page 21.

Variable Transfer Between RXSQL and SQL

Whenever an RXSQL command accepts a variable list, the variables should be specified in the order of the corresponding rows, separated by blanks.

Whenever an SQL statement specifies a REXX variable to be substituted, the variable should be prefixed with a colon. For example the command:

```
"RXSQL EXEC INSERT INTO TABLE_DAT VALUES (:A :B :C)"
```

inserts the values from the REXX variables A, B, and C into the three rows of the table TABLE_DAT.

To ensure portability, the value of all input variables substituted in SQL statements should be enclosed in single quotes. Under the Oracle RDBMS these quotes are optional, but under Sybase and SQL/DS they are required.

RXSQL CALL

```
RXSQL CALL name [INTO ovarname-list]
```

Arguments

name	The name of a cursor prepared with RXSQL PREP.
ovarname-list	Blank separated list of REXX variable names, that receive any SQL command output.

Cursor State

Initial	(1, 0), (1, 1)	Unprepared or Prepared.
Result	(1, 1)	Prepared.

Description

`RXSQL CALL` executes an SQL statement prepared with `RXSQL PREP`.

See Also

“RXSQL EXEC” on page 10 and “RXSQL PREP” on page 14.

RXSQL CLOSE

```
RXSQL CLOSE name
```

Arguments

name	The name of a cursor prepared with RXSQL PREP.

Cursor State

Initial	(1, 2)	Open.
Result	(1, 1)	Prepared.

Description

`RXSQL CLOSE` closes an open cursor, leaving it in a prepared state.

See Also

“RXSQL OPEN” on page 13, and “RXSQL FETCH” on page 11.

RXSQL COMMIT

```
RXSQL COMMIT [WORK] [RELEASE]
```

Keywords

WORK	Stay connected to the database. This is the default action if no keywords are specified.
RELEASE	Disconnect from the database.

Cursor State

Initial		Any.
Result	(1, 0)	Unprepared.

Description

`RXSQL COMMIT` permanently commits (i.e. saves) all changes made to the database since processing started or the last `RXSQL COMMIT` or `RXSQL ROLLBACK`.

See Also

“RXSQL ROLLBACK” on page 17.

RXSQL CONNECT

```
RXSQL CONNECT [id [IDENTIFIED BY] password] [TO dbname]
```

Arguments

id	The SQL userid.
password	The password for the SQL userid.
dbname	The name of the database to connect.

Description

`RXSQL CONNECT` connects (i.e. logs into) an SQL database.

RXSQL DESCRIBE

```
RXSQL DESCRIBE name [USING] [NAMES | ANY]
```

Arguments and Keywords

name	The name of a cursor prepared with RXSQL PREP.
Names	Fetch row names.
Any	Fetch row types and names.

Cursor State

Initial	(1, 0), (1, 1), (1,2)	Unprepared, prepared or open.
Result		No change.

Description

RXSQL DESCRIBE describes the names and types of output rows for a select statement prepared with RXSQL PREP. The names and type descriptions are placed in the variables SQLDAN. and SQLDAT. respectively.

See Also

“SQLDAN.” on page 21 and “SQLDAT.” on page 21.

RXSQL EXEC

```
RXSQL EXEC stmt
```

Arguments

name	The name of a cursor prepared with RXSQL PREP.

Description

`RXSQL EXEC` executes an SQL statement directly.

See Also

“RXSQL CALL” on page 5 and “RXSQL PREP” on page 14.

RXSQL FETCH

```
RXSQL FETCH name [COUNT count] [INTO] ovarname-list
```

Arguments

name	The name of a cursor prepared with RXSQL PREP.
count	The number of rows to fetch.
ovarname-list	Blank separated list of REXX variable names that receive the SQL command output.

Cursor State

Initial	(1, 2)	Open.
Result		Unchanged.

Description

`RXSQL FETCH` fetches the next row from a `select` that was prepared with `RXSQL PREP` and opened with `RXSQL OPEN`. When the last row is fetched, the return code is set to 4.

The variable SQLROWS is set to the number of rows actually fetched.

If the ovarname-list consists only of a single stem variable (ending with a "."), the first column is set with the stem ".1", the second with ".2", up to the number of columns in the table (i.e. view). The number of columns may be determined with the `RXSQL DESCRIBE` command.

`Count` is only supported under Oracle.

If a count is specified, each variable in the ovarname-list is treated as a list of compound stems (but the "." shouldn't actually be specified). The first row is set with the stem ".1", the second in ".2", up to ".SQLROWS".

If both a single stem variable and a count are specified, the resulting compound

symbols are in the form “stem.column.row”.

See Also

“RXSQL OPEN” on page 13, “RXSQL CLOSE” on page 6 and “RXSQL DESCRIBE” on page 9.

RXSQL OPEN

```
RXSQL OPEN name
```

Arguments

name	The name of a cursor prepared with RXSQL PREP.

Cursor State

Initial	(1, 0), (1, 1)	Unprepared or Prepared.
Result	(1, 2)	Open.

Description

`RXSQL OPEN` opens an SQL statement that was prepared with `RXSQL PREP` for processing with `RXSQL FETCH`.

See Also

“RXSQL FETCH” on page 11 and “RXSQL CLOSE” on page 6.

RXSQL PREP

```
RXSQL PREP name stmt
```

Arguments

name	The name of a cursor prepared with RXSQL PREP.
stmt	An SQL statement.

Cursor State

Initial	(0, 0), (1, 0)	None or Unprepared.
Result	(1, 1)	Prepared.

Description

`RXSQL PREP` prepares and parses an SQL statement for use with `RXSQL CALL` or `RXSQL OPEN`.

See Also

“RXSQL CALL” on page 5, “RXSQL DESCRIBE” on page 9, “RXSQL OPEN” on page 13, “RXSQL PURGE” on page 15, and “RXSQL STATE” on page 16.

RXSQL PURGE

```
RXSQL PURGE name | *
```

Arguments

name	The name of a cursor prepared with RXSQL PREP.

Cursor State

Initial	Any.
Result	None.

Description

`RXSQL PURGE` purges a cursor and frees all resources associated with it. If the cursor is open, it is closed before being purged.

See Also

“RXSQL PREP” on page 14.

RXSQL STATE

```
RXSQL STATE name
```

Arguments

name	The name of a cursor prepared with RXSQL PREP.

Cursor State

Initial	Any.
Result	Unchanged.

Description

`RXSQL STATE` returns the state of the specified cursor in the variable named SQLSTATE.

See Also

“SQLSTATE” on page 21.

RXSQL ROLLBACK

```
RXSQL ROLLBACK [WORK] [RELEASE]
```

Keywords

WORK	Stay connected to the database. This is the default action if no keywords are specified.
RELEASE	Disconnect from the database.

Description

`RXSQL ROLLBACK` rolls back (i.e. undoes) all changes made to the database since processing started or the last `RXSQL COMMIT` or `RXSQL ROLLBACK`.

See Also

“RXSQL COMMIT” on page 7.

CHAPTER 3

Variables

Summary

RXSQL may set a number of variables during execution, such as when an error occurs. The following pages summarize these variables.

RXSQLMSG

The error message text associated with an RXSQL error. RXSQL errors set the command return code (the REXX variable rc) to a value greater than or equal to 50. Note that RDBMS errors are reported in the variable SQLERRM.

TABLE 2. RXSQL error message text

RXSQL Return code	Message text
50	<DBNAME> is not connected
51	COUNT <COUNT> invalid - must be numeric
101	Insufficient storage - processor not initialized
103	Insufficient storage has been allocated by this processor
104	Error from EXECCOMM <RC>
105	SQL data type ”<TYPE>” (column <COLUMN>) not supported by ORXXSQL
106	ORXXSQL error - invalid internal code
107	No EXECCOMM subcom environment; e.g. not called from REXX program
108	Invalid variable name ”<VARNAME>”
109	Unexpected EXECCOMM return code <RC>
112	The first parameter is not a recognized operation

TABLE 2. RXSQL error message text

RXSQL Return code	Message text
113	The ”<OPERATION>” operation expects <ARGS> arguments but received<PASSED>
115	”<VARNAME>” could not be set, value is too long <LENGTH>
116	“<CURSOR>” is not a cursor statement - fetch cannot be completed
117	”<CURSOR>” is not open - fetch cannot be completed
119	No variables named to fetch into
120	Attempt to prepare more than the allowed limit of <COUNT> statements
121	Insufficient storage to prepare ”<CURSOR>”
122	No variable specified after : in statement in position %u
125	Colon found in position <POSITION> of an exec statement
127	Variable stem too long
129	”<MODULE>” is an undefined module number
130	<LEVEL> is an undefined trace level
133	Insufficient storage for value list of “<VALUELIST>”
136	“<CURSOR>” is not PREPED or DECLARED - <OPERATION> operation cannot be completed
137	“<CURSOR>” is not prepared - <OPERATION> operation cannot be completed
139	“<CURSOR>” is not open - unable to <OPERATION>
143	Statement name of length <LENGTH> is too long
144	”<NAME>” is not a recognizable statement name
145	The statement ”<NAME>” does not exist
147	Insufficient memory to initialize database
148	Insufficient storage to allocate data buffer of length <LENGTH>
149	SQL statement of length <LENGTH> is too long
150	Value passed in position <POSITION> of length <LENGTH> is too long
162	Invalid CONNECT statement
170	Invalid option ”<OPTION>” on CREATE
171	Invalid option combination on CREATE
172	Creator or progname ”<PROGNAME>” is too long
173	Invalid statement number <NUMBER>
174	Statement name ”<NAME>” already in use
175	No variable named in USING clause
176	No statement given on XPREP request

TABLE 2. RXSQL error message text

RXSQL Return code	Message text
177	Invalid data type ”<TYPE>” in USING variable
178	Invalid length <LENGTH> in USING variable
179	Invalid argument ”<ARG>” on SQLISL request
180	No input variables given for PUT <NAME>
181	Invalid option for DESCRIBE <OPTION>
182	Variable list not allowed on XCALL request
184	INTO not supported on CALL for PREPed statement
185	”<FEATURE>” not supported in this version of ORXXSQL

SQLSTMT

SQL statement text (set by PREP or STMT).

SQLSTATE

RXSQL statement state (set by RXSQL STATE).

TABLE 3. SQLSTATE variable

SQLSTATE variable value	Description
0 0	Could not be prepared.
1 0	Unprepared.
1 1	Prepared.
1 2	Open.

SQLDAN.

SQL column name table (set by “RXSQL DESCRIBE” on page 9). The number of names returned (the number of columns) is set in SQLDAN.0. The actual values are in SQLDAN.1 through SQLDAN.n.

SQLDAT.

SQL column attribute table (set by “RXSQL DESCRIBE” on page 9). The number of types returned (the number of columns) is set in SQLDAN.0. The actual values are in SQLDAT.1 through SQLDAT.n. The values depend on how

the table was defined, and hence depends on the underlying RDBMS.

TABLE 4. Oracle types and their `RXSQL DESCRIBE` equivalents.

Oracle type	Oracle type number	Returned by `RXSQL DESCRIBE`.
CHAR	1	C n
NUMBER	2	NUMBER n
LONG	8	L n
VARCHAR	9	V n
ROWID	11	ROWID n
DATE	12	DT n
RAW	23	V n
LONG RAW	24	VG n
(any other)		? n

TABLE 5. Sybase types and their `RXSQL DESCRIBE` equivalents.

Sybase type.	Returned by `RXSQL DESCRIBE`.
SYBINT2	S
SYBINT4	I
SYBFLT8	F
SYBDECIMAL	D
SYBCHAR	C n
SYBTEXT	C n
SYBVARCHAR	V n
SYBDATETIME4	DT n
SYBDATETIME	DT n
SYBDATETIMN	DT n
SYBBINARY	G n
SYBBIT	G n
SYBINTN	G n
SYBIMAGE	G n
SYBVARBINARY	VG n
SYBFLTN	NUMBER n
SYBNUMERIC	NUMBER n
SYBREAL	NUMBER n
SYBMONEY4	$ n

TABLE 5. Sybase types and their `RXSQL DESCRIBE` equivalents.

Sybase type.	Returned by `RXSQL DESCRIBE`.
SYBMONEY	$ n
SYBMONEYN	$ n
SYBINT1	? n
(any other)	? n

SQLCODE

RDBMS error code. This value is dependent on the underlying RDMBS. This variable is set when the command return code (the REXX variable rc) is 8.

SQLERRD.3

Number of rows fetched (same as RXSQLROWS).

SQLROWS

Number of rows fetched (same as RXSQLERRD.3, but easier to remember).

SQLERRM

RDBMS error messages. The value is dependent on the underlying RDBMS.

CHAPTER 4

Example

Summary

The following program demonstrates most RXSQL features.

TABLE 6. Example program.

```
/*
** Demonstrate RXSQL
**
** This program:
**
**      1) Connects to the database.
**      2) Drops the table named rbtab to make sure a
**         fresh copy can be created.
**      3) Creates a table named rbtab.
**      4) Inserts data.
**      5) Commits the inserts.
**      6) Prepares a select statement for the table created
**         above.
**      7) Opens a cursor for the select above.
**      8) Displays the fields using describe.
**      9) Fetches the fields and displays them.
**      10) Displays the state of the cursor.
**      11) Closes the cursor.
**      12) Purges the cursor.
**      13) Issues a commit release to disconnect.
**      14) Drops the table.
*/
/*
 * trace commands
 */
trace c
/*
 * number of rows to insert
 */
count = 10
/*
 * use command to address rxsql
 */
address command
/*
 * (1) connect to the SQL server
 */
```

TABLE 6. Example program.

```
"rxsql connect userid identified by password"
/*
 * (2) ensure the table "rbtab" isn't there (will probably cause an error message).
 */
"rxsql exec drop table rbtab"
/*
 * turn on error handler for non-0 return codes
 */
call on error
/*
 * (3) create the table called rbtab
 */
"rxsql exec create table rbtab (" ,
             "a char(4) ," ,
             "b varchar(4) ," ,
             "c varchar(4))"
/*
 * (4) set variables and fill the table
 */
call time 'r'
do i = 1 to count
  a = "'"i"'"
  b = "'"i || i"'"
  c = "'"i || i || i"'"
  'rxsql exec insert into rbtab values (:a,:b,:c)'
end i
/*
 * (5) commit changes and display load time
 */
say 'Load time =' time('r')
drop a b c
"rxsql commit"
/*
 * (6) select for all fields just created above
 */
"rxsql prep cursor select * from rbtab"
/*
 * (7) open the cursor
 */
"rxsql open cursor"
/*
 * (8) display the fields using describe
 */
"rxsql describe cursor using any"
do i = 1 to sqldan.0
  say "Field name =" left(sqldan.i,15) "Type:" sqldat.i
end i
/*
 * set the fetch timer
 */
call time 'r'
/*
 * (9) fetch the fields until rc <> 0
 */
cn = 0
do forever
  "rxsql fetch cursor a b c"
  if rc <> 0 then leave
  cn = cn + sqlrows
  say a b c
end
if cn <> count then say "Error, insert/fetch count mismatch"
/*
 * (10) display the state of the cursor
```

TABLE 6. Example program.

```
 */
"rxsql state cursor"
say sqlstate
/*
 * (11) close the cursor
 */
"rxsql close cursor"
/*
 * (12) purge the cursor
 */
"rxsql purge cursor"
/*
 * display fetch time
 */
say 'Fetch time =' time('e')
/*
 * (13) commit release to disconnect
 */
"rxsql commit release"
/*
 * (14) drop the table
 */
"rxsql exec drop table rbtab"
exit
/*
 * display RXSQL vars when a non-0 rc occurs
 */
error:
say sourceline(sigl)
say "SQLCODE="SQLCODE
say "SQLERRM="SQLERRM
say "RXSQLMSG="RXSQLMSG
drop SQLCODE SQLERRM RXSQLMSG
return
```

Example
Summary