Contents

CHAPTER 1

Overview 1
Introduction 1

CHAPTER 2

Commands 3
Introduction 3
Return codes 3
Cursors and Cursor State 3
Variable Transfer Between RXSQL and SQL 4
RXSQL CALL 5
RXSQL CLOSE 6
RXSQL COMMIT 7
RXSQL CONNECT 8
RXSQL DESCRIBE 9
RXSQL EXEC 10
RXSQL FETCH 11
RXSQL OPEN 13
RXSQL PREP 14
RXSQL PURGE 15
Contents

CHAPTER 3  Variables  19
Summary  19
RXSQLMSG 19
SQLSTMT 21
SQLSTATE 21
SQLDAN. 21
SQLDAT. 21
SQLCODE 23
SQLERRD.3 23
SQLROWS 23
SQLERRM 23

CHAPTER 4  Example  25
Summary  25
List of Tables

| TABLE 1. | RXSQL return codes | 3 |
| TABLE 2. | RXSQL error message text | 19 |
| TABLE 3. | SQLSTATE variable | 21 |
| TABLE 4. | Oracle types and their RXSQL DESCRIBE equivalents | 22 |
| TABLE 5. | Sybase types and their RXSQL DESCRIBE equivalents | 22 |
| TABLE 6. | Example program | 25 |
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.
Overview
Introduction

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

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.

Regardless of the database, all input variables 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 oварname-list]

Arguments

- name
  - The name of a cursor prepared with RxSQL PREP.

- oварname-list
  - Blank separated list of REXX variable names, that receive any SQL command output.

Cursor State

<table>
<thead>
<tr>
<th>State</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>(1, 0), (1, 1)</td>
<td>Unprepared or Prepared.</td>
</tr>
<tr>
<td>Result</td>
<td>(1, 1)</td>
<td>Prepared.</td>
</tr>
</tbody>
</table>

Description

RxSQL CALL executes an SQL statement prepared with RxSQL PREP.

See Also

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

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

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK</td>
<td>Stay connected to the database. This is the default action if no keywords are specified.</td>
</tr>
<tr>
<td>RELEASE</td>
<td>Disconnect from the database.</td>
</tr>
</tbody>
</table>

Cursor State

<table>
<thead>
<tr>
<th>Initial</th>
<th>Any.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>(1, 0)</td>
</tr>
<tr>
<td></td>
<td>Unprepared.</td>
</tr>
</tbody>
</table>

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
password
dbname

The SQL userid.
The password for the SQL userid.
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
Names
Any

The name of a cursor prepared with RXSQL PREP.
Fetch row names.
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

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of a cursor prepared with RXSQL PREP.</td>
</tr>
<tr>
<td>count</td>
<td>The number of rows to fetch.</td>
</tr>
<tr>
<td>ovarname-list</td>
<td>Blank separated list of REXX variable names that receive the SQL command output.</td>
</tr>
</tbody>
</table>

Cursor State

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>(1, 2) Open.</td>
</tr>
<tr>
<td>Result</td>
<td>Unchanged.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Initial</th>
<th>(1, 0), (1, 1)</th>
<th>Unprepared or Prepared.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>(1, 2)</td>
<td>Open.</td>
</tr>
</tbody>
</table>

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

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Initial</td>
<td>(0, 0), (1, 0)</td>
<td>None or Unprepared.</td>
</tr>
<tr>
<td>Result</td>
<td>(1, 1)</td>
<td>Prepared.</td>
</tr>
</tbody>
</table>

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.
Commands
RXSQL PURGE

RXSQL PURGE

RXSQL PURGE name | *

Arguments

name

The name of a cursor prepared with RXSQL PREP.

Cursor State

<table>
<thead>
<tr>
<th>Initial</th>
<th>Any.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result</td>
<td>None.</td>
</tr>
</tbody>
</table>

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

**TABLE 2. RXSQL error message text**

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

**Summary**

---

**TABLE 2. RXSQL error message text**

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

**SQLSTMT**

SQL statement text (set by PREP or STMT).

**SQLSTATE**

RXSQL statement state (set by RXSQL STATE).

**TABLE 3. SQLSTATE variable**

<table>
<thead>
<tr>
<th>SQLSTATE variable value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>Could not be prepared.</td>
</tr>
<tr>
<td>1 0</td>
<td>Unprepared.</td>
</tr>
<tr>
<td>1 1</td>
<td>Prepared.</td>
</tr>
<tr>
<td>1 2</td>
<td>Open.</td>
</tr>
</tbody>
</table>

**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.**

<table>
<thead>
<tr>
<th>Oracle type</th>
<th>Oracle type number</th>
<th>Returned by RXSQL DESCRIBE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHAR</td>
<td>1</td>
<td>C n</td>
</tr>
<tr>
<td>NUMBER</td>
<td>2</td>
<td>NUMBER n</td>
</tr>
<tr>
<td>LONG</td>
<td>8</td>
<td>L n</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>9</td>
<td>V n</td>
</tr>
<tr>
<td>ROWID</td>
<td>11</td>
<td>ROWID n</td>
</tr>
<tr>
<td>DATE</td>
<td>12</td>
<td>DT n</td>
</tr>
<tr>
<td>RAW</td>
<td>23</td>
<td>V n</td>
</tr>
<tr>
<td>LONG RAW</td>
<td>24</td>
<td>VG n</td>
</tr>
<tr>
<td>(any other)</td>
<td></td>
<td>? n</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Sybase type</th>
<th>Returned by RXSQL DESCRIBE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYBINT2</td>
<td>S</td>
</tr>
<tr>
<td>SYBINT4</td>
<td>I</td>
</tr>
<tr>
<td>SYBFLOAT8</td>
<td>F</td>
</tr>
<tr>
<td>SYBDECIMAL</td>
<td>D</td>
</tr>
<tr>
<td>SYBCHAR</td>
<td>C n</td>
</tr>
<tr>
<td>SYBTEXT</td>
<td>C n</td>
</tr>
<tr>
<td>SYBVARCHAR</td>
<td>V n</td>
</tr>
<tr>
<td>SYBDATETIME4</td>
<td>DT n</td>
</tr>
<tr>
<td>SYBDATETIME</td>
<td>DT n</td>
</tr>
<tr>
<td>SYBDATETIMN</td>
<td>DT n</td>
</tr>
<tr>
<td>SYBBINARY</td>
<td>G n</td>
</tr>
<tr>
<td>SYBBIT</td>
<td>G n</td>
</tr>
<tr>
<td>SYBINTN</td>
<td>G n</td>
</tr>
<tr>
<td>SYBIMAGE</td>
<td>G n</td>
</tr>
<tr>
<td>SYBVARBINARY</td>
<td>VG n</td>
</tr>
<tr>
<td>SYBFLOAT</td>
<td>NUMBER n</td>
</tr>
<tr>
<td>SYBDECIMAL</td>
<td>NUMBER n</td>
</tr>
<tr>
<td>SYBVARCHAR</td>
<td>NUMBER n</td>
</tr>
<tr>
<td>SYBINT2</td>
<td>$ n</td>
</tr>
</tbody>
</table>
Variables

Summary

TABLE 5. Sybase types and their RXSQL DESCRIBE equivalents.

<table>
<thead>
<tr>
<th>Sybase type.</th>
<th>Returned by RXSQL DESCRIBE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYBMONEY</td>
<td>$ n</td>
</tr>
<tr>
<td>SYBMONEYN</td>
<td>$ n</td>
</tr>
<tr>
<td>SYBINT1</td>
<td>? n</td>
</tr>
<tr>
<td>(any other)</td>
<td>? n</td>
</tr>
</tbody>
</table>

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.
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) Purses 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
```
Example
Summary

TABLE 6. Example program.

```rxsql
/*
"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
```

---

RXSQL - SQL for REXX 27