SQLRDD for xHarbour Builder

A Database Driver for all major SQL database systems

Reference guide

© 2004, Marcelo Lombardo
marcelo@xharbour.com.br

Chapter 1

Connection Functions

Connection Functions List

SR_GetConnection(nConnection)	Obtain the connection object to <nConnection>
SR_AddConnection(nType, cDSN, cUser, cPassword, cOwner, lCounter, lAutoCommit, lNoSetEnv)	Open a new connection to the database and setup the environment and management tables
SR_DeletedName(cName)	Set the name of the column to store the deleted status
SR_EvalFilters(lEval)	Set if filters are expressions (that must be evaluated in run time) or literal data
SR_ExistIndex(cTableName)	Same as file(cTableName) to be used with the SQL database
SR_ExistTable(cTableName)	Same as file(cTableName) to be used with the SQL database
SR_GetActiveConnection()	Get the active connection number
SR_RecnoName(cName)	Set the name of the column to store the record number
SR_SetActiveConnection(nCnn)	Set the active connection number
SR_SetUseSequences(lOpt)	Set the record numbering behavior to use sequence or not (see record numbering in the user's manual)
SR_UseDeleteds(lDeleteds)	Set the use of deleted records that can be recovered. If you issue “SR_UseDeleteds(.F.)” before create the table, “SR_DELETED” control column will not be created. If you issue it on a existing table, dbDelete() will become a real DELETE that cannot be recovered.
SR_UseSequences()	Test if sequences are enabled (see record numbering in the user's manual)
SR_SetToolsOwner(cOwner)	Set an username / schema / owner to the SQLRDD management tables
SR_GetToolsOwner()	Get the SQLRDD management tables owner

SR_AddConnection(<nType>, <cDSN>) ➔ nHandle

SR_AddConnection creates a new connection to the database system, and set it as active, returning a connection handle.

<nType> may be:

CONNECT_ODBC	(supports Unix ODBC and Windows ODBC32)
CONNECT_RPC	(not available)
CONNECT_MYSQL	(Native support for MySQL >= 4.1)
CONNECT_POSTGRES	(Native support for Postgres >= 7.3)
CONNECT_FIREBIRD	(Under development – But you can use ODBC)
CONNECT_ORACLE	(supports Oracle 9 and 10, under Windows or Linux)

<cDSN> is a string with connection information such as Host, Database Name, User Name, Password, etc. The string structure is different depending on the **connection type**:

CONNECT_ODBC

"DSN=ODBCdataSourceName;UID=username;PWD=password "

Where:

ODBCdataSourceName : the Data Source Name created in ODBC administrator panel.
username : Database login name
password : Database password

Example: Connect to SQL Server using data source name "Northwind", username "sa" and a blank password:

```
SR_AddConnection(CONNECT_ODBC, "DSN=Northwind;UID=sa;PWD=" )
```

CONNECT_MYSQL, CONNECT_POSTGRES and CONNECT_ORACLE

```
"HST=ServerNameOrIP;UID=username;PWD=password;DTB=DataBaseName;PRT=Port"
```

Where:

ServerNameOrIP: The network server name or its IP address.
username : Database login name
password : Database password
DataBaseName: Database name to be used (optional)
Port : TCP Port Number (optional)

Example: Connect to server “localhost”, database “test”, username “root” and password “1234”

```
SR_AddConnection(CONNECT_POSTGRES,
“HST=localhost;UID=root;PWD=1234;DTB=test“ )
```

SR_SetActiveConnection(<nHandle>)

Sets the actual active connection to the next dbUseArea() function, or USE command. This is not needed if the application has only one connection to the database server.

SR_GetActiveConnection() ➔ nHandle

Retrieves the active connection handle

SR_GetConnection(<nHandle>) ➔ oSql

Retrieves the active connection object. You can use the connection object to issue SQL Statements direct to the target database system, using following syntax:

```
oSql:exec( <cCommand>, [<lMsg>], [<lFetch>], [<aArray>], [<cFile>],
[<cAlias>], [<nMaxRecords>], [<lNoRecno>] ) ➔ nRetStatus
```

Where:

<cCommand> is a valid SQL statement to current database.

<lMsg> Set it to .F. if you don’t want to generate a Run Time error if command fails. Default is .T.

<lFetch> Set it to .T. if you want to fetch the resulting record set. You can choose to fetch row(s) to an array or file. Default is .F.

Note: If you try to fetch the result set of a DDL command or a INSERT, UPDATE or DELETE, database may generate a Run Time Error.

<aArray> An empty array (by reference) to be filled with record set, as follow:

{ { Line1Col1, Line1Col2, … }, { Line2Col1, Line2Col2, … }, … }

<cFile> Is the target filename to output record set. If file is already opened (with alias name like filename) SQLRDD will push result set in this file. File should have exact same structure as result set, or Run Time Error will be raised. If file does not exist, it will be created using default RDD. At the end of operation, file will be opened in exclusive mode and record positioned in Top.

<cAlias> Is an optional alias name to above file.

<nMaxRecords> Is an optional parameter to LIMIT fetch size.

<lNoRecno> If you set to .T., SR_RECNO column will not be fetched to target file or array. Default is .F.

Chapter 2

Other Functions

Function List

SR_LogFile(cFileName, aInfo)	Writes a log file
SR_SetCurrDate(d)	Set the current date to be used in Historic Workareas
SR_IsWAHist()	Returns .T. if current workarea supports Historic data
SR_SetNextDt(d)	Sets the historic date to the next write operation in the current workarea
SR_DisableHistoric()	Disable Historic support
SR_EnableHistoric()	Enable Historic support
SR_GetActiveDt()	Get the current date to the Historic workareas
SR_SetActiveDt(d)	Set the historic date to the current workarea
SR_cDBValue(uData, nSystemID)	Quotes any data type to the specific database
SR_SetTableInfoBlock(b)	Sets the table properties codeblock
SR_GetTableInfoBlock()	Obtain the current table information codeblock
SR_SetNextRecordBlock(b)	Sets a codeblock to be the record enumerator (if we does not use database sequences)
SR_GetNextRecordBlock()	Gets the current record enumerator code block
SR_Msg(nMsg)	Returns a error message description in the current language
SR_SetBaseLang(nLang)	Sets the current language
SR_SetTimeTrace(nConnection, nMilisseconds)	Sets the minimum execution time to trace queries to LONG_QRY.DBF
SR_StartLog(nConnection)	Starts tracing SQL commands to SQLLOG.DBF
SR_StopLog(nConnection)	Stops tracing SQL commands to SQLLOG.DBF
SR_SetFastOpen(lSet)	If you disable FastOpen, SQLRDD will behave 100% Clipper compatible in USE EXCLUSIVE behavior. Otherwise, all table open are always SHARED.
SR_GetFastOpen()	Get FatsOpen status

Chapter 3

SQL Parser Functions

SQL Parser Function List

SR_SQLCodeGen(apCode, aParam, nSystemId)	Create a specific-database SQL command based on the pCode array, and optionally fills it with parameters
SR_SQLParse(cCommand, @nError, @nErrorPos)	Parse a SQL command and create a pCode array
SR_ParserVersion()	Returns the SQL Parser version

Chapter 4

SQL Parser Syntax

SELECT Statement

SELECT [ALL | DISTINCT] [LIMIT iRecords]
  [TableAlias.]* | [TableAlias.]column 1, column 2, column n...
FROM
  ["]table["] | (subquery) | ? | :BINDVAR   [TableAlias]
[**WHERE**
  WhereExpression
  [TableAlias.Column [LEFT | RIGHT OUTER] JOIN TableAlias.Column]]
[**GROUP BY** [TableAlias.]column 1, column 2, column n...]
[**ORDER BY** [TableAlias.]column 1, column 2, column n...]
[**UNION** ...]

Column can be an expression list or a single expression:

  [(] column expression 01 [OPERATOR column expression 02 [OPERATOR column expression nn..]]] [)]
      [AS ["]ColumnNickName["]]

column expression can be:

  [(] Column Name | Integer Value | Real Value | QuotedString |
      DateString | NULL | FUNCTION(ParamList) | ? | ?? [)]

  The question mark symbol "?" works like a compile time parameter. A double question mark
  means a NOT NULL parameter

Column Name can be a column, a parameter or a bindvar:

  [alias.]["]ColumnName["] | [alias.]? | [alias.]:BINDVAR

QuotedString can be any sequence like: 'any character sequence'
DateString is: [YYYYMMDD]

OPERATOR can be:
   + | - | * | / | = | == | < | > | <> | != | AND | OR | LIKE | IS | CONCAT | IN

FUNCTION can be:

  count | max | min | isnull | substr | abs | power | round | trim | sum | avg

ParamList depends on the function, but generally is:

  column expression 1 [, column expression 2 [,...]]

WhereExpression is an expression list using all of the operators and most of the functions

UPDATE Statement

UPDATE
  ["]table["] | ? | :BINDVAR
SET
  ColumnName = column expression
[**WHERE**
  WhereExpression]

DELETE Statement

DELETE FROM
  ["]table["] | ? | :BINDVAR
 [**WHERE**
  WhereExpression]

INSERT Statement

INSERT INTO
  ["]table["] | ? | :BINDVAR
[(["]column["]… | ? | :BINDVAR)]
VALUES
(Integer Value | Real Value | QuotedString |
      DateString | NULL | FUNCTION(ParamList) | ? | ?? | :BINDVAR …)