RODBC function | R Documentation

10-13 minutes

ODBC Database Connectivity

`RODBC` implements odbc database connectivity with compliant databases where drivers exist on the host system. Two groups of commands are provided. `odbc*` commands implement relatively low level access to the odbc functions of similar name. `sql*` commands are higher level constructs to read, save, copy and manipulate data between data frames and sql tables. In general `sql*` commands return a data frame on success, or -1/verbose on error depending on the `errors` parameter. The `odbc*` group return -1 in `stat` on error. Up to 16 connections can be open at once to any combination of dsn/hosts. Columns are limited to 255 chars of non-binary data. The functions where usage is obvious from the name are not described below.

Keywords

database, SQL, odbc, access, mysql, postgresql, sybase, sqlserver, oracle

Usage

```
sqlQuery(channel, query, errors=T,
as.is=F,transposing=F,max=0,transposing=F)
sqlSave(channel, dat, tablename=NULL,append=F,
rownames=F, colnames=F,verbose=F,test=F)
sqlFetch(channel, dat, verbose=T,
as.is=F,rownames=F,colnames=F)
sqlTables(channel)
sqlPrimaryKeys(channel, sqtable)
sqlColumns(channel, sqtable, errors=F,special=F)
sqlDrop(channel, sqtable, errors=T, verbose=T)
sqlClear(channel, sqtable, errors=T, verbose=T)
sqlCopy(channel, query, destination,
destchannel=-1, verbose=T, errors=T)
sqlCopyTable(channel, srctable, desttable,
destchannel=-1, verbose=T, errors=T)
sqlGetResults(channel,as.is=F,errors=F,transposing=F,max=0,buffsize=10(
sqlUpdate(channel,dat,verbose=F,test=F)
odbcConnect(dsn, uid="", pwd="",
host="localhost",case="nochange")
odbcClose(channel)

odbcClearError(channel)
```

```
odbcQuery(channel, query)
odbcTables(channel)
odbcGetErrMsg(channel)
odbcColumns(channel, table)
odbcPrimaryKeys(channel, table)
odbcFetchRow(channel)
odbcFetchRows(channel,max,buffsize,transposing)
odbcColData(channel)
odbcNumRows(channel)
odbcNumFields(channel)
odbcNumCols(channel)

tolower(expr)
toupper(expr)
```

Arguments

host
: Hostname of the database server

channel
: connection handle returned by odbcConnect()

query
: any valid SQL statement

.*table
: a database table name accessible from the connected dsn

errors
: if TRUE halt and display error, else return -1

verbose
: Display statements as they are sent to the server

dat
: a data frame

rownames
: save row labels as the first column in the table

colnames
: save column names as first row of table

case
: Controls case changes for different rdbms engines

as.is
: as in read.table

transposing
: return rows and columns transposed

special
: return columns needed to specify a row uniquely

test
: show what would be done

Details

`odbcConnect(dsn,uid,pwd="",host="localhost",case="nochange")` establishes a connection to the dsn at `host`. It returns a integer, which is used as handle if no error occurred, -1 otherwise. For

databases that translate table and column names to case must be set as appropriate. Allowable values are `nochange`,`toupper` and `tolower` as well as the names of databases where the behaviour is known to me (currently `mysql`,`postgresql`, `oracle` and `msaccess`.

`sqlQuery(channel, query, errors=TRUE, as.is=FALSE,transposing=F, max=0,buffsize=1000)` is the workhorse function. It sends the SQL statement `query` to the server, using connection `channel`, returned by `odbcConnect`. Returns a data frame of results, transformed according to `as.is`. If `errors=FALSE` returns -1 on error, otherwise halts with a message from the server. `transposing` reverses columns and rows if `TRUE`. `buffsize` will yield a marginal increase in speed if increased for some database engines eg MSaccess. SQL beginners should note that the term `Query' includes any valid SQL statement including table creation, alteration, updates etc as well as SELECTs. The `sqlQuery` command is a convenience wrapper that calls first `odbcQuery` and then `sqlGetResults`. If finer grained control, for example over the number of rows fetched, these functions should be called manually.

`sqlGetResults(channel,as.is=FALSE,errors=FALSE,transposing=FALSE,max=0,` is a mid-level function. It should be called after a call to `odbcQuery` and used to retrieve waiting results into a dataframe. Its main use is with `max` set to non zero it will retrieve the result set in batches with repeated calls. This is useful for very large result sets which can be subjected to intermediate processing.

`sqlSave(channel, dat, tablename=NULL, append=F, rownames=F, colnames=F,verbose=T,test=F)` saves the data frame `dat` in the table `dat`. The table name is taken from tablename if given or the name of the dataframe.If the table exists and has the appropriate structure it is used, or else it is created anew with type `varchar(255)`. If `rownames=TRUE` the first column of the table will be the row labels with colname `rowname`. `rownames` can also be a string giving the desired name (see example). `colnames` copied the column names into row 1. This is intended for cases where case conversion alters the original column names and it is desired that they are retained. Note that there are drawbacks to this approach: it presupposes that the rows will be returned in correct order; not always valid. It will also cause numeric rows to be returned as factors. WARNING: This function uses the 'great white shark' method of testing tables (bite it and see). The logic will unceremoniously DROP the table and create it anew with VARCHAR column types in its attempt to find a writeable solution. `test=T` will not necessarily predict this behaviour. Attempting to write indexed columns or writing to pseudo- columns are less obvious causes of failed writes followed by a DROP. If your table structure is precious to you back it up.

`sqlFetch(channel,dat,verbose=T,as.is=F,colnames=F,rownames=F)` loads the the entire contents of the table `dat`. (The reverse of `sqlSave`) Rownames and column names are restored as indicated. (More accurately the first row and column returned is transferred to the row/col names).

`sqlCopy(channel, query, destination, destchannel=-1, verbose=TRUE, errors=TRUE)` as above, but saves the output of `query` in table `destination` on dsn `destchannel`.

`sqlCopyTable(channel, srctable, desttable, destchannel=-1, verbose=TRUE, errors=TRUE)` copies the structure of `srctable` to `desttable` on dsn `destchannel`. This is within the limitation of the odbc lowest common denominator. More precise control is possible via `sqlQuery`. `sqlClear(channel, sqtable, errors=TRUE, verbose=TRUE)` deletes the content of the table `sqtable`. No confirmation is requested.

`sqlDrop(channel, sqtable, errors=TRUE, verbose=TRUE)` removes the table `sqtable`. No confirmation is requested.

`sqlUpdate(channel,dat,verbose=F,test=F)` updates the table where the rows already exist. The dataframe must contain a column named after the row that the database regards as teh optimal for defining a row uniquely. (This is returned by `sqlColumns(...,special=T)`). `sqlColumns`, `sqlTables`, and `sqlPrimaryKeys` return information as data frames. Note that the column names contain underscores and are invalid in S unless quoted. The column names are not constant across ODBC versions so the data should be accessed by column number. The argument `special` to `sqlColumns` returns the rows needed to specify a row uniquely. This is intended to form the basis of a WHERE clause for updates.

`odbcClose(channel)` Clean up and free resources.

`odbcFetchRows(channel,max,buffsize,transposing)` This function returns a matrix of the pending rowset in `$data` limited to `max` rows if `max` is greater than 0. `buffsize` may be increased from the default of 1000 (rows*cols) for increased performance in a large dataset. This only has an effect with servers that do not return the number of rows affected by a query eg MSAccess, MSSqlServer. If `transposing` is `TRUE` the matrix will be transposed. This function is called by sqlGetResults, which then converts the matrix to a dataframe. This step incurs a significant performance penalty and working with matrices is much faster in large rowsets.

`odbcFetchRow(channel)` is a deprecated function that returns a vector comprising the next row of the waiting rowset.

The remaining functions beginning `odbc` are lower level functions that normally require explicit looping to deal with the results. Most return -1 on failure, indicating that a message is waiting for `odbcGetErrMsg`. The exception is that an invalid channel returns -2. Examples are present in the sql.R code.

`sqlgetresults(channel, as.is=FALSE, errors=FALSE)` returns the last result set created by an `odbc*` call as a data frame.

`toupper(expr)` and `tolower(expr)` do case conversion on strings or vectors.

Examples

```
library(RODBC)
data(USArrests)
channel <- odbcConnect("test", "", "")
sqlSave(channel, USArrests, rownames="State",
verbose=T)
options("dec",".")
sqlQuery(channel, "select State,Murder from
USArrests where rape > 30 order by Murder")
sqlFetch(channel,"USArrests")
sqlDrop(channel, USArrests)
odbcClose(channel)
rm(USArrests)
```

Documentation reproduced from package RODBC, version 0.7-2, License: GPL

Community examples

Looks like there are no examples yet.

Post a new example:

/////