Package 'RODBC'

April 14, 2016

Version 1.3-13
Revision \$Rev: 3461 \$
Date 2016-04-14
Title ODBC Database Access
Description An ODBC database interface.
SystemRequirements An ODBC3 driver manager and drivers.
Depends R (>= $3.0.0$)
Imports stats
LazyLoad yes
Biarch yes
License GPL-2 GPL-3
NeedsCompilation yes
Author Brian Ripley [aut, cre], Michael Lapsley [aut] (1999 to Oct 2002)
Maintainer Brian Ripley <ripley@stats.ox.ac.uk></ripley@stats.ox.ac.uk>
Repository CRAN
Date/Publication 2016-04-14 14:42:10
R topics documented:
RODBC-package 2 odbc-low-level 2 odbcClose 4 odbcConnect 5 odbcDataSources 6 odbcGetInfo 8 odbcSetAutoCommit 9 setSqlTypeInfo 10 sqlColumns 11 sqlCopy 12 sqlDrop 15
squitop

2 odbc-low-level

RODBC-package ODBC Database Connectivity																									
Index																									27
	sqlTypeInfo	• •			•			•	•	•				•		•		•				•			25
	sqlSave . sqlTables																								
	sqlQuery .																								17
	sqlFetch .																								16

Description

Package RODBC implements ODBC database connectivity.

See the package manual for details of installation and use. (This will show up as a vignette, and can be accessed *via* RShowDoc("RODBC", package="RODBC").)

Details

Two groups of functions are provided. The mainly internal odbc* commands implement low-level access to the ODBC functions of similar name. The sq1* functions operate at a higher level to read, save, copy and manipulate data between data frames and SQL tables. Many connections can be open at once to any combination of DSN/hosts.

Author(s)

Michael Lapsley and Brian Ripley

odbc-low-level

Low-level ODBC functions

Description

R functions which talk directly to the ODBC interface.

Usage

odbc-low-level 3

Arguments

channel connection handle as returned by odbcConnect, of class "RODBC".

catalog, schema, tableName, tableType

NULL or character: whether these do anything depends on the ODBC driver. The first three can be length-one character vectors, and tableType can specify zero

or more types.

literal logical: should arguments be interpreted literally or including wildcards?

query any valid SQL statement.

rows_at_time The number of rows to fetch at a time, between 1 and 1024. Not all drivers work

correctly with values > 1: see sqlQuery.

max limit on the number of rows to fetch, with 0 indicating no limit.

buffsize the number of records to be transferred at a time.

nullstring character string to be used when reading SQL_NULL_DATA items in a column

transferred as character.

believeNRows logical. Is the number of rows returned by the ODBC connection believable?

Details

These are low-level functions called by sqlTables, sqlQuery, sqlGetResults and similar high-level functions. They are likely to be confind to the **RODBC** namespace in the near future.

odbcTables enquires about the tables on the connected database. Whether arguments after the first do anything and what they do depends on the ODBC driver: see the help on sqlTables for some driver-specific details.

odbcFetchRows returns a data frame of the pending rowset, limited to max rows if max is greater than 0.

buffsize may be increased from the default of 1000 rows for increased performance on a large dataset. This only has an effect when max = 0 and believeNRows = FALSE (either for the ODBC connection or for this function call), in which case buffsize is used as the initial allocation length of the R vectors to hold the results. (Values of less than 100 are increased to 100.) If the initial size is too small the vector length is doubled, repeatedly if necessary.

Value

odbcGetErrMsg returns a (possibly zero-length) character vector of pending messages. odbcClearError returns nothing, invisibly.

The otheres return 1 on success and -1 on failure, indicating that a message is waiting to be retrieved odbcGetErrMsg. odbcFetchRows may return -2 indicating "No Data", the message that would be returned by odbcGetErrMsg.

Author(s)

Michael Lapsley and Brian Ripley

See Also

 ${\sf sqlQuery}, odbc {\sf Connect}, odbc {\sf GetErrMsg}.$

4 odbcClose

odbcClose

ODBC Close Connections

Description

Close connections to ODBC databases.

Usage

```
odbcClose(channel)
## S3 method for class 'RODBC'
close(con, ...)
odbcCloseAll()
```

Arguments

```
channel, con RODBC connection object as returned by odbcConnect.
... additional arguments passed from the generic.
```

Details

odbcClose cleans up and frees resources. It is also the method for the generic function close. odbcCloseAll closes all open channels (amongst the first 1000 used in the session).

Channels are closed at the end of an R session, and may also be closed by garbage collection if no object refers to them. In both cases a warning is given (but may not be seen at the end of a console session).

Value

Function odbcClose returns a logical indicating if it succeeded, invisibly unless a warning is given. The close method returns 0 (success) or 1, invisibly.

Author(s)

Michael Lapsley and Brian Ripley

See Also

odbcConnect

odbcConnect 5

odbcConnect ODBC Open Connections	
-----------------------------------	--

Description

Open connections to ODBC databases.

Usage

Arguments

dsn character string. A registered data source name.

uid, pwd UID and password for authentication (if required).

connection character string. See your ODBC documentation for the format.

... further arguments to be passed to odbcDriverConnect.

case Controls case changes for different DBMS engines. See 'Details'.

channel RODBC connection object returned by odbcConnect.

believeNRows logical. Is the number of rows returned by the ODBC connection believable?

Not true for some Oracle and Sybase drivers, apparently, nor for Actual Tech-

nologies' SQLite driver for Mac OS X.

colQuote, tabQuote

how to quote column (table) names in SQL statements. Can be of length 0 (no quoting), a length-1 character vector giving the quote character to be used at both ends, or a length-2 character vector giving the beginning and ending quotes. ANSI SQL uses double quotes, but the default mode for a MySQL server is to use backticks.

The defaults are backtick ('`') if the DBMS is identified as "MySQL" by the driver, and double quote otherwise. A user reported that the SAS ODBC driver required colQuote = NULL.

interpretDot logical. Should table names of the form *qualifier*. table be interpreted as table

table in schema qualifier (and for MySQL 'schema' means database)?

DBMSencoding character string naming the encoding returned by the DBMS. The default means the encoding of the locale R is running under. Values other than the default re-

quire iconv to be available: it always is from R 2.10.0, otherwise see capabilities.

6 odbcConnect

rows_at_time The default number of rows to fetch at a time, between 1 and 1024. Not all drivers work correctly with values > 1: see sqlQuery.

readOnlyOptimize

logical: should the connection be optimized for read-only access?

Details

odbcConnect establishes a connection to the specified DSN, and odbcDriverConnect allows a more flexible specification *via* a connection string. odbcConnect uses the connection string "DSN=*dsn*; UID=*uid*; PWD=*pwd*",

omitting the last two components if they are empty.

For DBMSs that translate table and column names case must be set appropriately. Allowable values are "nochange", "toupper" and "tolower" as well as the names of databases where the behaviour is known to us (currently "mysql", which maps to lower case on Windows but not on Linux, "postgresql" (lower), and "msaccess" (nochange)). If case is not specified, the default is "nochange" unless the appropriate value can be figured out from the DBMS name reported by the ODBC driver. It is likely that "toupper" is desirable on IBM's DB2, but this is not enforced.

Note that readOnlyOptimize may do nothing, and is **not** guaranteed to enforce read-only access. With drivers that support it, it is used to optimize locking strategies, transaction management and so on. It does make access to Mimer read-only, and has no effect on MySQL.

Function odbcReConnect re-connects to a database using the settings of an existing (and presumably now closed) channel object. Arguments given in the original call can be overridden as needed.

Note that if a password is supplied (either as a pwd argument or as part of the DSN) it may be stored in the connection.string element of the return value, but the value is (from **RODBC** 1.3-0) replaced by ******. (This will break odbcReConnect.)

If it is possible to set the DBMS or ODBC driver to communicate in the character set of the R session then this should be done. For example, MySQL can set the communication character set *via* SQL, e.g. 'SET NAMES 'utf8''.

Value

A non-negative integer which is used as handle if no error occurred, -1 otherwise. A successful return has class "RODBC", and attributes including

connection.string

the full ODBC connection string.

case the value of case.

id a numeric ID for the channel.
believeNRows the value of believeNRows.
rows_at_time the value of rows_at_time.

Note

Several errors which have been reported as bugs in **RODBC** 1.3-0 which were in fact ODBC driver errors that can be circumvented by setting rows_at_time = 1 (and the warning under that argument has always been there). The drivers involved have been third-party Oracle drivers and old SQL Server drivers.

odbcDataSources 7

Author(s)

Michael Lapsley, Brian Ripley

See Also

```
odbcClose, sqlQuery, odbcGetInfo
```

Examples

```
## Not run:
# MySQL
channel <- odbcConnect("test", uid="ripley", pwd="secret")
# PostgreSQL: 'case' should be detected automatically
channel <- odbcConnect("pg", uid="ripley", pwd="secret", case="postgresql")
# re-connection
odbcCloseAll()
channel <- odbcReConnect(channel) # must re-assign as the data may change
## End(Not run)</pre>
```

odbcDataSources

List ODBC Data Sources

Description

List known ODBC data sources.

Usage

```
odbcDataSources(type = c("all", "user", "system"))
```

Arguments

type

User DSNs, system DSNs or all?

Value

A named character vector of DSN descriptions, with names the DSNs.

Author(s)

Brian Ripley

8 odbcGetInfo

Examples

```
## Not run:
> odbcDataSources()
        test
                  sqlite3
                                testpg
     "MySQL"
                "sqlite3" "PostgreSQL"
or
                           testdb3
                                                             sqlite3
         "MySQL ODBC 3.51 Driver"
                                               "SQLite3 ODBC Driver"
                          bdr.xls
                                                             testacc
 "Microsoft Excel Driver (*.xls)" "Microsoft Access Driver (*.mdb)"
                           testpg
                                                           SQLServer
                "PostgreSQL ANSI"
                                                 "SQL Native Client"
                           Oracle
                                                                 DB2
       "Oracle in OraDb10g_home1" "IBM DB2 ODBC DRIVER - DB2COPY1"
                          testpgw
             "PostgreSQL Unicode"
                                             "MySQL ODBC 5.1 Driver"
                SQLite Datasource
                                             SQLite UTF-8 Datasource
                                        "SQLite ODBC (UTF-8) Driver"
             "SQLite ODBC Driver"
               SQLite3 Datasource
                                                             "MIMER"
            "SQLite3 ODBC Driver"
## End(Not run)
```

odbcGetInfo

Request Information on an ODBC Connection

Description

Request information on an ODBC connection.

Usage

```
odbcGetInfo(channel)
```

Arguments

channel

connection handle as returned by odbcConnect of class "RODBC".

Value

A named character string giving information on the database and ODBC driver in use on the connection channel.

Author(s)

Brian Ripley

odbcSetAutoCommit 9

Examples

```
## Not run:
odbcGetInfo(channel) # under Windows XP
## MySQL returned
               DBMS_Name
                                            DBMS_Ver
                                                               Driver_ODBC_Ver
       "MySQL" "5.1.35-community"
Data_Source_Name Driver_Name
                                                                         "03.51"
                                      Driver_Name
                                                                     Driver_Ver
               "testdb5"
ODBC_Ver
                                     "myodbc5.dll"
                                                                   "05.01.0005"
                                        Server_Name
            "03.52.0000" "localhost via TCP/IP"
## MS Access returned
        DBMS_Name
                           DBMS_Ver Driver_ODBC_Ver Data_Source_Name
                                                   "03.51"
         "ACCESS" "04.00.0000"
                                                                     "testacc"
  Driver_Name Driver_Ver "odbcjt32.dll" "04.00.6305"
                                                ODBC_Ver
                                                                   Server_Name
                                             "03.52.0000"
                                                                      "ACCESS"
## SQL Server 2008 Express returned
               DBMS_Name
                                            DBMS_Ver
                                                               Driver_ODBC_Ver
      DBMS_Ver

DSOFT SQL Server" "10.00.1600"

Data_Source_Name Driver_Name

"SQLServer" "SQLNCLI.DLL"

ODBC_Ver Server_Name

"03.52.0000" "AUK\\SQLEXPRESS"
"Microsoft SQL Server"
                                                                         "03.52"
                                                                     Driver_Ver
                                                                   "09.00.4035"
## End(Not run)
```

odbcSetAutoCommit

ODBC Set Auto-Commit Mode

Description

Set ODBC database connection's auto-commit mode.

Usage

```
odbcSetAutoCommit(channel, autoCommit = TRUE)
odbcEndTran(channel, commit = TRUE)
```

Arguments

channel RODBC connection object returned by odbcConnect.

autoCommit logical. Set auto-commit on?

commit logical. Commit or rollback pending transaction?

10 setSqlTypeInfo

Details

Auto-commit is a concept supported only by ODBC connections to transactional DBMSs.

If a connection to a transactional DBMS is in auto-commit mode (the default), then all its SQL statements will be executed and committed as individual transactions. Otherwise, its SQL statements are grouped into transactions that are terminated by an execution of commit or rollback. Switching a connection to auto-commit mode commits the pending transaction.

By default, new connections are in auto-commit mode. If auto-commit mode has been disabled, a call to odbcEndTran or an SQL commit statement must be executed in order to commit changes; otherwise, pending database changes will not be saved.

Value

odbcSetAutoCommit stops if channel is an invalid connection. The function returns -1 on error, 0 on success and on success with a message that would be returned by odbcGetErrMsg.

Author(s)

Norman Yamada, Yasser El-Zein

setSqlTypeInfo

Specify or Query a Mapping of R Types to DBMS Types

Description

Specify or retrieve a mapping of R types to DBMS datatypes.

Usage

```
setSqlTypeInfo(driver, value)
getSqlTypeInfo(driver)
```

Arguments

driver A character string specifying the DBMS_name as returned by odbcGetInfo. Op-

tional for getSqlTypeInfo.

value A named list with character values. This should have names "double", "integer",

"character" and "logical", and values SQL types appropriate to the DBMS.

Details

This information is used by sqlSave if it creates a table in the DBMS and is not overridden by arguments typeInfo or varTypes. Mappings are included for MySQL, PostgreSQL, SQLite, Oracle, Mimer, DB2 on Windows, and the Microsoft SQL Server, Access, Excel and Dbase drivers.

The SQL types chosen should be nullable to allow NAs to be represented. (Bit and boolean types often are not.)

sqlColumns 11

Value

For setSqlTypeInfo none.

For getSqlTypeInfo with an argument, a named list. Without an argument, a data frame.

Author(s)

Brian Ripley

See Also

```
sqlTypeInfo, sqlSave.
```

Examples

sqlColumns

Query Column Structure in ODBC Tables

Description

Enquire about the column structure of tables on an ODBC database connection.

Usage

Arguments

channel	connection object as returned by odbcConnect.
sqtable	character string: a database table (or view or similar) name accessible from the connected DSN. If wildcards are allowed (only for sqlColumns(special=FALSE)), results for all matching tables.
errors	logical: if true halt and display error, else return -1.
as.is	see sqlGetResults.

12 sqlColumns

special logical. If true, return only the column(s) needed to specify a row uniquely.

Depending on the database, there might be none.

catalog, schema

NULL or character: additional information on where to locate the table: see sqlTables for driver-specific details. Wildcards may be supported in schema

for sqlColumns(special=FALSE).

literal logical: wildcards may be interpreted in schema and sqtable: if so this may

suppress such interpretation.

Details

The argument special = TRUE to sqlColumns returns the column(s) needed to specify a row uniquely. This is intended to form the basis of an SQL WHERE clause for update queries (see sqlUpdate), and what (if anything) it does is DBMS-specific. On many DBMSs it will return the primary keys if present: on others it will return a pseudo-column such as 'ROWID' (Oracle) or '_ROWID_' (SQLite), either always (Oracle) or if there is no primary key.

Primary keys are implemented in some DBMSs and drivers. A table can have a single column designated as a primary key or, in some cases, multiple columns. Primary keys should not be nullable (that is, cannot contain missing values). They can be specified as part of a 'CREATE TABLE' statement or added by a 'ALTER TABLE' statement.

In principle specifying catalog should select an alternative database in MySQL or an attached database in SQLite, but neither works with current drivers.

If sqtable contains '.' and neither catalog nor schema is supplied, an attempt is made to interpret *qualifier*. *table* as table table in schema *qualifier* (and for MySQL 'schema' means 'database', but the current drivers fail to interpret catalog=, so this does not yet work). (This can be suppressed by opening the connection with interpretDot = FALSE.) This has been tested successfully on PostgreSQL, SQL Server, Oracle, DB2 and Mimer.

Whether wildcards are accepted for sqtable and schema in sqlColumns(special = FALSE) depends on the driver and may be changed by the value of literal. For example, the PostgreSQL driver tested allowed wildcards in schema only if literal = FALSE and never in sqtable, whereas two MySQL drivers both failed to match a database when catalog was supplied and always allowed wildcards in sqtable even if literal = TRUE.

Value

A data frame on success. If no data is returned, either a zero-row data frame or an error. (For example, if there are no primary keys or special column(s) in this table an empty data frame is returned, but if primary keys are not supported by the ODBC driver or DBMS, an error code results.)

The column names are not constant across ODBC versions so the data should be accessed by column number.

For sqlPrimaryKeys and sqlColumns(special=FALSE) the first four columns give the catalog, schema, table and column names (where applicable). For sqlPrimaryKeys the next two columns are the column sequence number (starting with 1) and name of the primary key: drivers can define further columns. For sqlColumns(special=FALSE) there are 18 columns: see http://msdn.microsoft.com/en-us/library/ms711683(VS.85).aspx. Those beyond the first 6 shown in the examples give the 'ordinal position' (column 17) and further characteristics of the column type: see sqlTypeInfo.

sqlCopy 13

For the numeric values returned by sqlColumns(special=TRUE) see http://msdn.microsoft.com/en-us/library/ms714602(VS.85).aspx: the scope should always be 2 (the session) since that is the scope requested in the call. For the PSEUDO_COLUMN column, the possible values are 0 (unknown), 1 (no) and 2 (yes).

Author(s)

Michael Lapsley and Brian Ripley

See Also

odbcConnect, sqlQuery, sqlFetch, sqlSave, sqlTables, odbcGetInfo

Examples

```
## Not run: ## example results from MySQL
> channel <- odbcConnect("test")</pre>
> sqlDrop(channel, "USArrests", errors = FALSE) # precautionary
> sqlSave(channel, USArrests, addPK = TRUE)
> sqlColumns(channel, "USArrests")
 TABLE_CAT TABLE_SCHEM TABLE_NAME COLUMN_NAME DATA_TYPE TYPE_NAME
   ripley <NA> USArrests rownames 12 varchar
1
                <NA> USArrests Murder
2
  ripley
                                              8 double
                3
   ripley
   ripley
    ripley
... 12 more columns
> sqlColumns(channel, "USArrests", special = TRUE)
 SCOPE COLUMN_NAME DATA_TYPE TYPE_NAME COLUMN_SIZE BUFFER_LENGTH
    2 rownames 12 varchar
                                          255
                                                      255
 DECIMAL_DIGITS PSEUDO_COLUMN
> sqlPrimaryKeys(channel, "USArrests")
 TABLE_CAT TABLE_SCHEM TABLE_NAME COLUMN_NAME KEY_SEQ PK_NAME
      <NA> <NA> USArrests rownames
                                              1 PRIMARY
> sqlDrop(channel, "USArrests")
> close(channel)
## End(Not run)
```

sqlCopy

ODBC Copy

Description

Functions to copy tables or result sets from one database to another.

14 sqlCopy

Usage

Arguments

channel, destchannel

connection handle as returned by odbcConnect.

query any valid SQL statement destination, srctable, desttable

character: a database table name accessible from the connected DSN.

verbose Display statements as they are sent to the server? errors if TRUE halt and display error, else return -1.
... additional arguments to be passed to sqlSave.

Details

sqlCopy as is like sqlQuery, but saves the output of query in table destination on channel destchannel.

sqlCopyTable copies the structure of srctable to desttable on DSN destchannel. This is within the limitations of the ODBC lowest common denominator. More precise control is possible *via* sqlQuery.

Value

See sqlGetResults.

Author(s)

Michael Lapsley and Brian Ripley

See Also

```
sqlQuery, sqlSave
```

Examples

sqlDrop 15

sqlDrop

Deletion Operations on Tables in ODBC databases

Description

```
sqlClear deletes all the rows of the table sqtable. sqlDrop removes the table sqtable (if permitted).
```

Usage

```
sqlClear(channel, sqtable, errors = TRUE)
sqlDrop(channel, sqtable, errors = TRUE)
```

Arguments

channel connection object as returned by odbcConnect.

sqtable character string: a database table name accessible from the connected DSN. This

can be a 'dotted' name of the form schema. table.

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

Details

These submit 'TRUNCATE TABLE' and 'DROP TABLE' SQL queries respectively.

'Dotted' table names are allowed on systems that support them but the existence of the table is not checked and so attempting these operations on a non-existent table will give a low-level error. (This can be suppressed by opening the connection with interpretDot = FALSE.)

The default 'drop' behaviour in Oracle is to move the table to the 'recycle bin': use

```
sqlQuery(channel, "PURGE recyclebin")
```

to empty the recycle bin.

The current user might not have privileges to allow these operations, and Actual Technologies' Mac OS X SQLite driver has a bug causing them silently to fail.

Value

If errors = FALSE, a numeric value, invisibly. Otherwise a character string or invisible().

Author(s)

Michael Lapsley and Brian Ripley

See Also

```
odbcConnect, sqlQuery, sqlFetch, sqlSave, sqlTables, odbcGetInfo
```

16 sqlFetch

sqlFetch	Reading Tables from ODBC Databases

Description

Read some or all of a table from an ODBC database into a data frame.

Usage

```
sqlFetch(channel, sqtable, ..., colnames = FALSE, rownames = TRUE)
sqlFetchMore(channel, ..., colnames = FALSE, rownames = TRUE)
```

Arguments

channel	connection handle returned by odbcConnect.
sqtable	a database table name accessible from the connected DSN. This should be either a literal character string or a character vector of length 1.
	additional arguments to be passed to sqlQuery or sqlGetResults. See 'Details'.
colnames	logical: retrieve column names from first row of table? (For use when sqlSave(colnames = TRUE) was used.)
rownames	either logical or character. If logical, retrieve row names from the first column (rownames) in the table? If character, the column name to retrieve them from.

Details

Note the 'table' includes whatever table-like objects are provided by the DBMS, in particular views and system tables.

sqlFetch by default retrieves the entire contents of the table sqtable. Rownames and column names are restored as indicated (assuming that they have been placed in the table by the corresponding arguments to sqlSave).

Alternatively, sqlFetch can fetch the first max rows, in which case sqlFetchMore will retrieve further result rows, provided there has been no other ODBC query on that channel in the meantime.

These functions try to cope with the peculiar way the Excel ODBC driver handles table names, and to quote Access table names which contain spaces. Dotted table names, e.g. myschema.mytable, are allowed on systems that support them, unless the connection was opened with interpretDot = FALSE.

Useful additional parameters to pass to sqlQuery or sqlGetResults include

max: limit on the number of rows to fetch, with 0 (the default) indicating no limit.

nullstring: character string to be used when reading SQL_NULL_DATA character items from the database: default NA_character_.

na.strings: character string(s) to be mapped to NA when reading character data: default "NA". as.is: as in sqlGetResults.

sqlQuery 17

dec: The character for the decimal place to be assumed when converting character columns to numeric.

rows_at_time: Allow for multiple rows to be retrieved at once. See sqlQuery.

Value

A data frame on success, or a character or numeric error code (see sqlQuery).

Note

If the table name desired is not a valid SQL name (alphanumeric plus _) and these functions are not able to interpret the name, you can use sqlQuery with whatever quoting mechanism your DBMS vendor provides (e.g. [] on some Microsoft products and backticks on MySQL).

Author(s)

Michael Lapsley and Brian Ripley

See Also

```
sqlSave, sqlQuery, odbcConnect, odbcGetInfo
```

Examples

```
## Not run:
channel <- odbcConnect("test")
sqlSave(channel, USArrests)
sqlFetch(channel, "USArrests") # get the lot
sqlFetch(channel, "USArrests", max = 20, rows_at_time = 10)
sqlFetchMore(channel, max = 20)
sqlFetchMore(channel) # get the rest
sqlDrop(channel, "USArrests")
close(channel)
## End(Not run)</pre>
```

sqlQuery

Query an ODBC Database

Description

Submit an SQL query to an ODBC database, and retrieve the results.

18 sqlQuery

Usage

Arguments

channel connection handle as returned by odbcConnect.

query any valid SQL statement.

errors logical: if true halt and display error, else return -1.
... additional arguments to be passed to sqlGetResults.

rows_at_time The number of rows to fetch at a time, between 1 and 1024. See 'Details'.

as.is which (if any) columns returned as character should be converted to another

type? Allowed values are as for read. table. See 'Details'.

max limit on the number of rows to fetch, with 0 indicating no limit.

buffsize an initial guess at the number of rows, used if max = 0 and believeNRows == FALSE.

nullstring character string to be used when reading SQL_NULL_DATA character items from

the database.

na.strings character vector of strings to be mapped to NA when reading character data.

believeNRows logical. Is the number of rows returned by the ODBC connection believable?

This might have been set to false when the channel was opened, and if so that

setting cannot be overridden.

dec The character for the decimal place to be assumed when converting character

columns to numeric.

stringsAsFactors

logical: should columns returned as character and not excluded by as.is and

not converted to anything else be converted to factors?

Details

sqlQuery is the workhorse function of **RODBC**. It sends the SQL statement query to the server, using connection channel returned by odbcConnect, and retrieves (some or all of) the results *via* sqlGetResults.

The term 'query' includes any valid SQL statement including table creation, alteration, updates etc as well as 'SELECT's. The sqlQuery command is a convenience wrapper that first calls odbcQuery and then sqlGetResults. If finer-grained control is needed, for example over the number of rows fetched, additional arguments can be passed to sqlQuery or the underlying functions called directly.

sqlGetResults is a mid-level function. It is called after a call to sqlQuery or odbcQuery to retrieve waiting results into a data frame. Its main use is with max set to non-zero when 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.

sqlQuery 19

Where possible sqlGetResults transfers data in binary form: this happens for columns of (ODBC) SQL types double, real, integer and smallint, and for binary SQL types (which are transferred as lists of raw vectors, given class "ODBC_binary"). All other SQL data types are converted to character strings by the ODBC interface.

This paragraph applies only to SQL data types which are returned by ODBC as character vectors. If when creating the connection (see odbcConnect) DBMSencoding was set to a non-empty value, the character strings are re-encoded. Then if as.is is true for a column, it is returned as a character vector. Otherwise (where detected) date, datetime and timestamp values are converted to the "Date" or "POSIXct" class. (Some drivers seem to confuse times with dates, so times may get converted too. Also, some DBMSs (e.g. Oracle's) idea of date is a date-time.) Remaining cases are converted by R using type.convert. When character data are to be converted to numeric data, the setting of options("dec") is used to map the character used by the ODBC driver in setting decimal points—this is set to a locale-specific value when RODBC is initialized if it is not already set

Using buffsize will yield a marginal increase in speed if set to no less than the maximum number of rows when believeNRows = FALSE. (If set too small it can result in unnecessarily high memory use as the buffers will need to be expanded.)

Modern drivers should work (and work faster, especially if communicating with a remote machine) with rows_at_time = 100, the usual default, or more. (However, some drivers may mis-fetch multiple rows, in which case set rows_at_time = 1 when creating the connection.) However, if max is specified then this may fetch too many rows and hence it could be reduced (but then this setting applies to all subsequent fetches from that result set). Another circumstance in which you might want to reduce rows_at_time is if there are large character columns in the result set: with the default value up to 6Mb of buffer for each such column could be allocated to store intermediate results.

Value

On success, a data frame (possibly with 0 rows) or character string. On error, if errors = TRUE a character vector of error message(s), otherwise an invisible integer error code -1 (general, call odbcGetErrMsg for details) or -2 (no data, which may not be an error as some SQL statements do return no data).

Author(s)

Michael Lapsley and Brian Ripley

See Also

```
odbcConnect, sqlFetch, sqlSave, sqlTables, odbcQuery
```

Examples

```
## Not run:
channel <- odbcConnect("test")
sqlSave(channel, USArrests, rownames = "State", verbose = TRUE)
# options(dec=".") # optional, if DBMS is not locale-aware or set to ASCII
## note case of State, Murder, Rape are DBMS-dependent,
## and some drivers need column and table names double-quoted.</pre>
```

20 sqlSave

sqlSave

Write a Data Frame to a Table in an ODBC Database

Description

Write or update a table in an ODBC database.

Usage

```
sqlSave(channel, dat, tablename = NULL, append = FALSE,
    rownames = TRUE, colnames = FALSE, verbose = FALSE,
    safer = TRUE, addPK = FALSE, typeInfo, varTypes,
    fast = TRUE, test = FALSE, nastring = NULL)

sqlUpdate(channel, dat, tablename = NULL, index = NULL,
    verbose = FALSE, test = FALSE, nastring = NULL,
    fast = TRUE)
```

Arguments

channel connection handle returned by odbcConnect.

dat a data frame.

tablename character: a database table name accessible from the connected DSN. If missing,

the name of dat.

index character. Name(s) of index column(s) to be used.
append logical. Should data be appended to an existing table?

rownames either logical or character. If logical, save the row names as the first column

rownames in the table? If character, the column name under which to save the

rownames.

colnames logical: save column names as the first row of table? verbose display statements as they are sent to the server?

safer logical. If true, create a non-existing table but only allow appends to an existing

table. If false, allow sqlSave to attempt to delete all the rows of an existing

table, or to drop it.

addPK logical. Should rownames (if included) be specified as a primary key?

typeInfo optional list of DBMS datatypes. Should have elements named "character",

"double" and "integer".

varTypes an optional named character vector giving the DBMSs datatypes to be used for

some (or all) of the columns if a table is to be created.

sqlSave 21

fast logical. If false, write data a row at a time. If true, use a parametrized INSERT INTO

or UPDATE query to write all the data in one operation.

test logical: if TRUE show what would be done, only.

nastring optional character string to be used for writing NAs to the database. See 'Details'.

Details

sqlSave saves the data frame dat in the table tablename. If the table exists and has the appropriate structure it is used, or else it is created anew. If a new table is created, column names are remapped by removing any characters which are not alphanumeric or _, and the types are selected by consulting arguments varTypes and typeInfo, then looking the driver up in the database used by getSqlTypeInfo or failing that by interrogating sqlTypeInfo.

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 column name (see 'Examples'). If colnames is true, the column names are copied 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 the correct order; not always valid. It will also cause numeric columns to be returned as factors.

Argument addPK = TRUE causes the row names to be marked as a primary key. This is usually a good idea, and may allow database updates to be done. However, the ODBC drivers for some DBMSs (e.g. Access) do not support primary keys, and earlier versions of the PostgreSQL ODBC driver generated internal memory corruption if this option is used.

sqlUpdate updates the table where the rows already exist. Data frame dat should contain columns with names that map to (some of) the columns in the table. It also needs to contain the column(s) specified by index which together identify the rows to be updated. If index = NULL, the function tries to identify such columns. First it looks for a primary key for the table, then for the column(s) that the database regards as the optimal for defining a row uniquely (these are returned by sqlColumns(special = TRUE): if this returns a pseudo-column it cannot be used as we do not have values for the rows to be changed). Finally, the row names are used if they are stored as column "rownames" in the table.

When fast = TRUE, NAs are always written as SQL nulls in the database, and this is also the case if fast = FALSE and nastring = NULL (its default value). Otherwise nastring gives the character string to be sent to the driver when NAs are encountered: for all but the simplest applications it will be better to prepare a data frame with non-null missing values already substituted.

If fast = FALSE all data are sent as character strings. If fast = TRUE, integer and double vectors are sent as types SQL_C_SLONG and SQL_C_DOUBLE respectively. Some drivers seem to require fast = FALSE to send other types, e.g. datetime. SQLite's approach is to use the data to determine how it is stored, and this does not work well with fast = TRUE.

If tablename contains '.' and neither catalog nor schema is supplied, an attempt is made to interpret *qualifier*. *table* names as table *table* in schema *qualifier* (and for MySQL 'schema' means 'database'). (This can be suppressed by opening the connection with interpretDot = FALSE.)

Value

1 invisibly for success (and failures cause errors).

22 sqlTables

Warning

sqlSave(safer = FALSE) 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 its own choice of column types in its attempt to find a writable solution. test = TRUE 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 it is up to you back it up.

Author(s)

Michael Lapsley and Brian Ripley

See Also

```
sqlFetch, sqlQuery, odbcConnect, odbcGetInfo
```

Examples

```
## Not run:
channel <- odbcConnect("test")
sqlSave(channel, USArrests, rownames = "state", addPK=TRUE)
sqlFetch(channel, "USArrests", rownames = "state") # get the lot
foo <- cbind(state=row.names(USArrests), USArrests)[1:3, c(1,3)]
foo[1,2] <- 222
sqlUpdate(channel, foo, "USArrests")
sqlFetch(channel, "USArrests", rownames = "state", max = 5)
sqlDrop(channel, "USArrests")
close(channel)
## End(Not run)</pre>
```

sqlTables

List Tables on an ODBC Connection

Description

List the table-like objects accessible from an ODBC connection. What objects are 'table-like' depends on the DBMS, ODBC driver and perhaps even the configuration settings: in particular some connections report system tables and some do not.

Usage

sqlTables 23

Arguments

channel connection handle as returned by odbcConnect.

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

as.is as in sqlGetResults.

catalog, schema, tableName, tableType

NULL or character: whether these do anything depends on the ODBC driver. The first three can be length-one character vectors, and tableType can specify zero or more types in separate elements of a character vector.

literal logical: (where supported) should arguments be interpreted literally or including

. logical: (where supported) should arguments be interpreted literally or includ

wildcards?

Value

A data frame on success, or character/numeric on error depending on the errors argument. (Use sqlGetResults for further details of errors.)

The column names depend on the database, containing a third column TABLE_NAME (not always in upper case): however, they are supposed to be always in the same order.

The first column is the 'catalog' or (in ODBC2 parlance) 'qualifier', the second the 'schema' or (ODBC2) 'owner', the third the name, the fourth the table type (one of "TABLE", "VIEW", "SYSTEM TABLE", "ALIAS", "SYNONYM", or a driver-specific type name) and the fifth column any remarks.

Oddly, the Microsoft Excel driver considers worksheets to be system tables, and named ranges to be tables.

Driver-specific details

Whether the additional arguments are implemented and what they do is driver-specific. The standard SQL wildcards are *underscore* to match a single character and *percent* to match zero or more characters (and often backslash will escape these): these are not used for table types. All of these drivers interpret wildcards in tableName, and in catalog or schema where supported.

Setting one of catalog or schema to "%" and the other and tableName to "" should give a list of available catalogs or schemas, whereas

```
catalog = "", schema = "", tableName = "", tableType = "%"
```

should list the supported table types.

For MySQL, catalog refers to a database whereas schema is mostly ignored, and literal is ignored. To list all databases use just catalog = "%". In the 5.1.x driver, use catalog="db_name", tableName="%" to list the tables in another database, and to list the table types use the form displayed above.

For PostgreSQL's ODBC driver catalog is ignored (except that catalog = "" is required when listing schema or table types) and literal works for both schema and for tableName.

SQLite ODBC ignores catalog and schema, except that the displayed form is used to list table types. So although it is possible to attach databases and to refer to them by the *dotted name* notation, it is apparently impossible to list tables on attached databases.

24 sqlTables

Microsoft SQL Server 2008 interprets both catalog and schema. With literal = TRUE it only finds tables if schema is set (even to an empty string). Schemas are only listed if they contain objects.

Oracle's Windows ODBC driver finds no matches if anything non-empty is supplied for the catalog argument. Unless a schema is specified it lists tables in all schemas. It lists available table types as just "TABLE" and "VIEW", but other types appear in listings. With literal = TRUE it only finds tables if schema is set (even to an empty string).

DB2 implements schemas but not catalogs. literal = TRUE has no effect. In some uses case matters and upper-case names must be used for schemas.

The Microsoft Access and Excel drivers interpret catalog as the name of the Access .mdb or Excel .xls file (with the path but without the extension): wildcards are interpreted in catalog (for files in the same folder as the attached database) and tableName. Using schema is an error except when listing catalogs or table types. The Excel driver matched tableType = "TABLE" (a named range) but not tableType = "SYSTEM TABLE" (the type returned for worksheets).

The Actual Technologies Access/Excel driver ignores all the additional arguments.

Author(s)

Michael Lapsley and Brian Ripley

See Also

sqlGetResults

Examples

```
## Not run:
> sqlTables(channel, "USArrests")
## MySQL example
 TABLE_CAT TABLE_SCHEM TABLE_NAME TABLE_TYPE REMARKS
     ripley
                         USArrests
                                         TABLE
## PostgreSQL example
 TABLE_QUALIFIER TABLE_OWNER TABLE_NAME TABLE_TYPE REMARKS
                       public usarrests
                                               TABLE
           ripley
## Microsoft Access example
> sqlTables(channel)
      TABLE_CAT TABLE_SCHEM
                                    TABLE_NAME
                                                 TABLE_TYPE REMARKS
1 C:\bdr\test
                     <NA> MSysAccessObjects SYSTEM TABLE
                                                              <NA>
2 C:\bdr\test
                     <NA>
                                    MSysACEs SYSTEM TABLE
                                                              <NA>
3 C:\bdr\test
                     <NA>
                                MSysObjects SYSTEM TABLE
                                                              <NA>
4 C:\bdr\test
                     <NA>
                                 {\tt MSysQueries} \ {\tt SYSTEM} \ {\tt TABLE}
                                                              <NA>
                     <NA> MSysRelationships SYSTEM TABLE
                                                              <NA>
5 C:\bdr\test
6 C:\bdr\test
                     <NA>
                                       hills
                                                   TABLE
                                                              <NA>
7 C:\bdr\test
                     <NA>
                                   USArrests
                                                    TABLE
                                                              <NA>
## End(Not run)
```

sqlTypeInfo 25

sqlTypeInfo	Request Information about Data Types in an ODBC Database

Description

Request information about data types in an ODBC database

Usage

```
sqlTypeInfo(channel, type = "all", errors = TRUE, as.is = TRUE)
```

Arguments

channel	connection handle as returned by odbcConnect.
type	The types of columns about which information is requested. Possible values are "all", "char", "varchar", "wchar", "wvarchar" (Unicode), "real", "float", "double", "integer", "smallint", "date", "time", "timestamp", "binary", "varbinary", "longvarbinary" and (its alias) "blob".
errors	logical: if true halt and display error, else return -1.
as.is	as in sqlGetResults.

Details

sqlTypeInfo attempts to find the types of columns the database supports: ODBC drivers are not required to support this (but all known examples do). Where it is supported, it is used by sqlSave to decide what column types to create when creating a new table in the database.

Value

A data frame on success, or character/numeric on error depending on the errors argument. Use sqlGetResults for further details of errors.

The columns returned may depend on the ODBC driver manager. For a fully ODBC 3 manager, see http://msdn.microsoft.com/en-us/library/ms714632(VS.85).aspx: the symbolic constants mentioned there will be returned as numbers (and the values of the numeric constants can be found in the ODBC headers such as 'sql.h' and 'sqlext.h').

Author(s)

Brian Ripley

See Also

```
sqlGetResults, odbcGetInfo
```

26 sqlTypeInfo

Examples

```
## Not run:
> names(sqlTypeInfo(channel))
[1] "TYPE_NAME" "DA"
```

[1] "TYPE_NAME" "DATA_TYPE" "COLUMN_SIZE"

[4] "LITERAL_PREFIX" "LITERAL_SUFFIX" "CREATE_PARAMS"

[7] "NULLABLE" "CASE_SENSITIVE" "SEARCHABLE"

[10] "UNSIGNED_ATTRIBUTE" "FIXED_PREC_SCALE" "AUTO_UNIQUE_VALUE"

[13] "LOCAL_TYPE_NAME" "MINIMUM_SCALE" "MAXIMUM_SCALE" [16] "SQL_DATATYPE" "SQL_DATETIME_SUB" "NUM_PREC_RADIX"

[19] "INTERVAL_PRECISION"

End(Not run)

Index

*Topic IO	iconv, 5
odbc-low-level, 2	100117, 3
odbcClose, 4	odbc-low-level, 2
odbcConnect, 5	odbcClearError (odbc-low-level), 2
odbcGetInfo, 8	odbcClose, 4, 7
odbcSetAutoCommit, 9	odbcCloseAll (odbcClose), 4
RODBC-package, 2	odbcConnect, 3, 4, 5, 8, 11, 13–20, 22, 23, 25
setSqlTypeInfo, 10	odbcDataSources, 7
sqlColumns, 11	odbcDriverConnect (odbcConnect), 5
sqlCopy, 13	odbcEndTran (odbcSetAutoCommit), 9
sqlDrop, 15	odbcFetchRows (odbc-low-level), 2
sqlFetch, 16	odbcGetErrMsg, 3, 19
sqlQuery, 17	odbcGetErrMsg (odbc-low-level), 2
sqlSave, 20	odbcGetInfo, 7, 8, 10, 13, 15, 17, 22, 25
sqlTables, 22	odbcQuery, <i>18</i> , <i>19</i>
sqlTypeInfo, 25	odbcQuery(odbc-low-level), 2
*Topic database	odbcReConnect (odbcConnect), 5
odbc-low-level, 2	odbcSetAutoCommit, 9
odbcClose, 4	odbcTables (odbc-low-level), 2
odbcConnect, 5	
odbcGetInfo, 8	read.table, 18
odbcSetAutoCommit, 9	RODBC (RODBC-package), 2
RODBC-package, 2	RODBC-package, 2
setSqlTypeInfo, 10	
sqlColumns, 11	setSqlTypeInfo, 10
sqlCopy, 13	sqlClear(sqlDrop), 15
sqlDrop, 15	sqlColumns, 11, 21
sqlFetch, 16	sqlCopy, 13
sqlQuery, 17	sqlCopyTable (sqlCopy), 13
sqlSave, 20	sqlDrop, 15
sqlTables, 22	sqlFetch, 13, 15, 16, 19, 22
sqlTypeInfo, 25	sqlFetchMore (sqlFetch), 16
*Topic utilities	sqlGetResults, 3, 11, 14, 16, 23–25
odbcDataSources, 7	sqlGetResults (sqlQuery), 17
	sqlPrimaryKeys (sqlColumns), 11
capabilities, 5	sqlQuery, 3, 6, 7, 13–17, 17, 22
<pre>close.RODBC (odbcClose), 4</pre>	sqlSave, 10, 11, 13–17, 19, 20, 25
10.17 7.0.21	sqlTables, 3, 12, 13, 15, 19, 22
getSqlTypeInfo, 21	sqlTypeInfo, 11, 12, 21, 25
<pre>getSqlTypeInfo(setSqlTypeInfo), 10</pre>	sqlUpdate, 12

28 INDEX

```
sqlUpdate(sqlSave), 20
type.convert, 19
```