Package 'RODBC'

April 17, 2009

Version 1.2-5

Date 2009-01-21

Author Originally Michael Lapsley <mlapsley@sthelier.sghms.ac.uk>, from Oct 2002 B. D. Ripley <ripley@stats.ox.ac.uk>

Maintainer B. D. Ripley <ripley@stats.ox.ac.uk>

Title ODBC Database Access

Description An ODBC database interface

SystemRequirements An ODBC3 driver manager and drivers. See README.

Depends R (>= 2.5.0), utils

Imports stats

LazyLoad yes

License GPL-2 | GPL-3

Repository CRAN

Date/Publication 2009-01-21 11:10:17

R topics documented:

odbc 2
odbcClose 3
odbcConnect 4
odbcDataSources 6
odbcErrors 6
odbcGetInfo 7
odbcSetAutoCommit 8
RODBC 9
RODBCtables 10
setSqlTypeInfo 11

sqlCopy 12
sqlFetch 14
sqlQuery 15
sqlSave 17
sqlTables 20
sqlTypeInfo 21

Index **23**

`odbc` *Low-level ODBC functions*

Description

R functions which talk directly to the ODBC interface.

Usage

```
odbcTables(channel)

odbcColumns(channel, table)
odbcPrimaryKeys(channel, table)

odbcQuery(channel, query, rows_at_time = attr(channel, "rows_at_time"))
odbcFetchRows(channel, max = 0, buffsize = 1000, nullstring = NA,
              believeNRows = TRUE)
```

Arguments

`channel`	connection handle as returned by `odbcConnect()` of class `"RODBC"`.
`query`	any valid SQL statement
`table`	a database table name accessible from the connected dsn. This can be either a character string or an (unquoted) symbol.
`rows_at_time`	The number of rows to fetch at a time, up to 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 from the database.
`believeNRows`	logical. Is the number of rows returned by the ODBC connection believable? Not true for Oracle, apparently.

Details

`odbcFetchRows` returns a data frame 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 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 e.g. MS Access, MS SQLServer.

Value

Most return `1` on success and `-1` on failure, indicating that a message is waiting for `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

`sqlQuery`, `odbcConnect`, `odbcGetErrMsg`.

`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 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.

Value

Function `odbcClose` returns invisibly a logical indicating if it succeeded.

Author(s)

Michael Lapsley, Brian Ripley

See Also

odbcConnect

---

odbcConnect *ODBC Open Connections*

Description

Open connections to ODBC databases.

Usage

```
odbcConnect(dsn, uid = "", pwd = "", ...)

odbcDriverConnect(connection = "", case, believeNRows = TRUE,
                  colQuote, tabQuote = colQuote, DBMSencoding = "",
                  rows_at_time = 1000, bulk_add = NULL)

odbcReConnect(channel, case, believeNRows)
```

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 Oracle and Sybase, apparently. Nor for MySQL Connector/ODBC 5.00.11.
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 for both ends, or a length-2 character string 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.
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 require iconv to be available: see capabilities.

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

`bulk_add` if `"yes"`, `SQLBulkOperations` will be used in `sqlSave`. Set to `"no"` to suppress this.

Details

`odbcConnect` establishes a connection to the 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 comments if they are empty. See the examples for other uses of connection strings.

For databases that translate table and column names the 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 us (currently `"mysql"` (which maps to lower case on Windows but not on Linux), `"postgresql"` (lower), `"oracle"` (upper) 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.

Function `odbcReConnect` re-connects to a database using the settings of an existing (and presumably now closed) channel object. Arguments `case` and `believeNRows` are taken from the object, but can be overridden by supplying those arguments.

If it is possible to set the DBMS 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`.

Author(s)

Michael Lapsley, Brian Ripley

See Also

`odbcClose`, `sqlQuery`, `odbcGetInfo`

Examples

```
## Not run:
# MySQL
channel <- odbcConnect("test", uid="ripley", pwd="secret")
# PostgreSQL
channel <- odbcConnect("pg", uid="ripley", pwd="secret", case="postgresql")

# re-connection
odbcCloseAll()
channel <- odbcReConnect(channel) # must re-assign as the data may well change
## End(Not run)
```

---

odbcDataSources *List ODBC Data Sources*

Description

List ODBC data sources.

Usage

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

Arguments

type	User DSNs, system DSNs or all?

Value

A character vector of descriptions, with names the DSNs.

---

odbcErrors *RODBC Error-Handling Utility Functions*

Description

Utility functions for examining and clearing error messages which have been collected during low-level ODBC calls.

Usage

```
odbcGetErrMsg(channel)
odbcClearError(channel)
```

Arguments

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

Value

odbcGetErrMsg returns a (possibly zero-length) character vector of pending messages.

odbcClearError returns nothing, invisibly.

Author(s)

Michael Lapsley

See Also

odbcQuery, odbcConnect

---

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

Examples

```
## Not run:
odbcGetInfo(channel) # under Windows XP
## MySQL returned
             DBMS_Name                DBMS_Ver        Driver_ODBC_Ver
               "MySQL"           "4.0.20a-nt"                "03.51"
      Data_Source_Name             Driver_Name             Driver_Ver
             "testdb3"           "myodbc3.dll"             "03.51.09"
              ODBC_Ver             Server_Name
          "03.52.0000" "localhost via TCP/IP"
## MS Access returned
```

```
       DBMS_Name          DBMS_Ver  Driver_ODBC_Ver Data_Source_Name
        "ACCESS"     "04.00.0000"          "03.51"        "testacc"
     Driver_Name       Driver_Ver         ODBC_Ver      Server_Name
  "odbcjt32.dll"     "04.00.6304"     "03.52.0000"         "ACCESS"
## MSDE 2000 returned
             DBMS_Name               DBMS_Ver        Driver_ODBC_Ver
"Microsoft SQL Server"           "08.00.0760"               "03.52"
      Data_Source_Name            Driver_Name            Driver_Ver
                 "MSDE"         "SQLSRV32.DLL"          "03.85.1117"
              ODBC_Ver            Server_Name
           "03.52.0000"                 "AUK"
## 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?

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 (1) on success with a message that would be returned by `odbcGetErrMsg`.

Author(s)

Norman Yamada, Yasser El-Zein

RODBC *ODBC Database Connectivity*

Description

RODBC implements ODBC database connectivity with compliant databases where drivers exist on the host system.

Details

Two groups of commands are provided. The mainly internal 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. Many connections can be open at once to any combination of dsn/hosts.

The functions try to cope with the peculiar way the Excel ODBC driver handles table names. However, SQL expects both table and column names to be alphanumeric plus _, and RODBC does not support vendor extensions. Most of the functions will drop other characters from table and column names.

options(dec) can be used to set the decimal point to be used when reading numbers from character data on the database: the default is taken from the current locale by Sys.localeconv.

Author(s)

Michael Lapsley and Brian Ripley

See Also

odbcConnect, sqlFetch, sqlSave, sqlTables, odbcGetInfo

Examples

```
## Not run:
channel <- odbcConnect("test")
sqlSave(channel, USArrests, rownames = "State", verbose = TRUE)
sqlQuery(channel, paste("select State, Murder from USArrests",
                        "where Rape > 30 order by Murder"))
sqlFetch(channel, "USArrests", rownames = "State")
sqlDrop(channel, "USArrests")
close(channel)
## End(Not run)
```

RODBCtables *Operations on Tables in ODBC databases*

Description

Operations on tables in ODBC databases.

Usage

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

sqlDrop(channel, sqtable, errors = TRUE)

sqlColumns(channel, sqtable, errors = FALSE, as.is = TRUE,
           special = FALSE)

sqlPrimaryKeys(channel, sqtable, errors = FALSE, as.is = TRUE)
```

Arguments

channel	connection object as returned by odbcConnect.
sqtable	character: a database table name accessible from the connected dsn.
errors	if TRUE halt and display error, else return -1
as.is	as in sqlGetResults.
special	return only the column(s) needed to specify a row uniquely. Depending on the database, there might be none.

Details

sqlClear deletes the content of the table sqtable. No confirmation is requested.

sqlDrop removes the table sqtable. No confirmation is requested.

sqlColumns and sqlPrimaryKeys return information as data frames. 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 columns needed to specify a row uniquely. This is intended to form the basis of a WHERE clause for updates (see sqlUpdate).

Value

A data frame on success, or character/numeric on error depending on the errors parameter. 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 DBMS, an error code results.)

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")
sqlDrop(channel, "USArrests", errors = FALSE) # precautionary
sqlSave(channel, USArrests)
sqlColumns(channel, "USArrests")
##   Table_cat Table_schema Table_name Column_name Data_type Type_name
## 1                        USArrests    rownames        12   varchar
## 2                        USArrests      murder         8    double
## 3                        USArrests     assault         4   integer
## 4                        USArrests    urbanpop         4   integer
## 5                        USArrests        rape         8    double
##   Column_size Buffer_length Decimal_digits Num_prec_radix Nullable Remarks
## 1         255           255           <NA>           <NA>        0
## 2          22             8             31             10        1
## 3          11             4              0             10        1
## 4          11             4              0             10        1
## 5          22             8             31             10        1
sqlColumns(channel, "USArrests", special = TRUE)
##   Scope Column_name Data_type Type_name Precision Length Scale
## 1     2    rownames        12   varchar         0      0     0
##   Pseudo_column
## 1             1
sqlPrimaryKeys(channel, "USArrests")
##   Table_qualifier Table_owner Table_name Column_name Key_seq Pk_name
## 1           <NA>        <NA>  USArrests    rownames       1 PRIMARY
sqlClear(channel, "USArrests")
sqlDrop(channel, "USArrests")
close(channel)
## End(Not run)
```

---

setSqlTypeInfo *Specify 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. Optional for getSqlTypeInfo.
value	A named list with character values. This should have names "double", "integer", "character" and "logical".

Details

This information is used by sqlSave if it creates a table in the DBMS.

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

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

```
## Not run:
getSqlTypeInfo()
getSqlTypeInfo("MySQL")
setSqlTypeInfo("SQLServer",
               list(double="float", integer="int",
                    character="varchar(255)", logical="varchar(5)"))
## End(Not run)
```

---

sqlCopy *ODBC Copy*

Description

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

Usage

```
sqlCopy(channel, query, destination, destchannel = channel,
        verbose = FALSE, errors = TRUE, ...)

sqlCopyTable(channel, srctable, desttable, destchannel = channel,
             verbose = FALSE, errors = TRUE)
```

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

```
## Not run:
## example for a l/casing DBMS
sqlSave(channel, USArrests, rownames = "state")
query <- paste("select state, murder from usarrests",
               "where rape > 30", "order by murder")
sqlCopy(channel, query, "HighRape", rownames = FALSE)
sqlFetch(channel, "HighRape", rownames = "state", max = 5)
sqlDrop(channel, "HighRape")
## End(Not run)
```

sqlFetch *Reading Tables from ODBC Databases*

Description

Read a table of 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 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.

Details

sqlFetch retrieves the 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). sqlFetchMore will retrieve further results from the query (provided there has been no other ODBC query on that channel in the meantime).

It tries to cope with the peculiar way the Excel ODBC driver handles table names, and to quote Access table names which contain spaces.

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.

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

as.is: as in sqlGetResults.

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. The default is 1000.

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 _), use `sqlQuery` with whatever quoting mechanism your DBMS vendor provides (e.g. `[ ]` on Microsoft products and backticks on recent versions of 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)
sqlFetchMore(channel, max=20)
sqlFetchMore(channel) # get the rest
sqlDrop(channel, "USArrests")
close(channel)
## End(Not run)
```

`sqlQuery` *Query an ODBC Database*

Description

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

Usage

```
sqlQuery(channel, query, errors = TRUE, ..., rows_at_time = 1)

sqlGetResults(channel, as.is = FALSE,  errors = FALSE,
              max = 0, buffsize = 1000,
              nullstring = NA, na.strings = "NA",
              believeNRows = TRUE, dec = getOption("dec"),
              stringsAsFactors = default.stringsAsFactors())
```

Arguments

channel	connection handle as returned by odbcConnect.
query	any valid SQL statement
errors	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, up to 1024. Not all drivers work correctly with values > 1. See Details.
as.is	which (if any) character columns should be converted, as in read.table? See the 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 for the driver.
nullstring	character string to be used when reading SQL_NULL_DATA character items from the database.
na.strings	character string(s) to be mapped to NA when reading character data.
believeNRows	logical. Is the number of rows returned by the ODBC connection believable? This might already be set to false when the channel was opened, and can that setting cannot be overridden.
dec	The character for the decimal place to be assumed when converting character columns to numeric.
stringsAsFactors	should character columns 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.

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 is needed, for example over the number of rows fetched, these functions should be called directly or additional arguments passed to sqlQuery.

sqlGetResults is a mid-level function. It should be called after a call to sqlQuery or odbcQuery and used 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.

Where possible sqlGetResults transfers data directly: this happens for double, real, integer and smallint columns in the table. All other SQL data types are converted to character strings by the ODBC interface. If the as.is is true for a column, it is returned as character. Otherwise (where detected) date, datetime and timestamp values are converted to "Date" and "POSIXct" values respectively. (Some drivers seem to confuse times with dates, so times may get converted

too.) Other types are converted by R using type.convert. When character data are to be converted to numeric data, the setting of options("dec") to map the character used up 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_a_time = 1024. However, some drivers may mis-fetch multiple rows, so set this to 1 if the results are incorrect.

Value

A data frame (possibly with 0 rows) on success. If errors = TRUE, a character vector of error message(s), otherwise error code −1 (general, call odbcGetErrMsg for details) or −2 (no data, which may not be an error as some SQL commands 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
## note case of State, Murder, rape are DBMS-dependent.
sqlQuery(channel, paste("select State, Murder from USArrests",
                        "where Rape > 30 order by Murder"))
close(channel)
## End(Not run)
```

---

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, oldstyle = 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?
oldstyle	logical. If false, attempt to use sqlTypeInfo to choose the types of columns when a table has to be created. If true, create all columns as varchar(255).
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.
fast	logical. If false, write data a row at a time. If true, use a parametrized INSERT INTO / UPDATE query to write all the data in one operation, or SQLBulkOperations where supported.
test	show what would be done, only.
nastring	character string to be used for writing NAs to the database. The default, NULL, attempts to write a missing value as a database null. Only supported for fast=FALSE.

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 example). colnames copies 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.

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

sqlUpdate updates the table where the rows already exist. Data frame dat should 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 rows. First it looks for a primary key in the data frame, 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 there is no such column the rownames are used provided they are stored as column "rownames" in the table.

The value of nastring is used for all the columns and no attempt is made to check if the column is nullable. 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 is 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.

Value

1 invisibly for success (and failures cause errors).

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 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)
```

---

sqlTables *List Tables on an ODBC Database*

Description

List the tables on an ODBC database.

Usage

```
sqlTables(channel, errors = FALSE, as.is = TRUE)
```

Arguments

channel	connection handle as returned by odbcConnect.
errors	if TRUE halt and display error, else return -1
as.is	as in sqlGetResults.

Value

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

The column names depend on the database, containing a column table_name in some mixture of cases (see the examples).

Author(s)

Michael Lapsley and Brian Ripley

See Also

sqlGetResults

Examples

```
## Not run:
channel <- odbcConnect("test")
sqlTables(channel, "USArrests")
## MySQL example
##   Table_qualifier Table_owner Table_name Table_type     Remarks
##1                            usarrests      TABLE MySQL table
## Microsoft Access example
##      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>       MSysQueries SYSTEM TABLE    <NA>
##5 C:\bdr\test        <NA> MSysRelationships SYSTEM TABLE    <NA>
##6 C:\bdr\test        <NA>             hills        TABLE    <NA>
##7 C:\bdr\test        <NA>         USArrests        TABLE    <NA>
close(channel)
## End(Not run)
```

`sqlTypeInfo` *Request Information about DataTypes in an ODBC Database*

Description

Request information about datatypes 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"`, `"real"`, `"float"`, `"double"`, `"integer"`, `"smallint"`, `"timestamp"`.
`errors`	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. Not all ODBC drivers support this. Where it is supported, it is used to decide what column types to create when creating a new table in the database.

Value

A data frame on success, or character/verbose on error depending on the `errors` parameter. See `sqlGetResults` for further details.

Author(s)

Brian Ripley

See Also

sqlGetResults, odbcGetInfo

Index

∗Topic **IO**
odbc, 2
odbcClose, 3
odbcConnect, 4
odbcErrors, 6
odbcGetInfo, 7
odbcSetAutoCommit, 8
RODBC, 9
RODBCtables, 10
setSqlTypeInfo, 11
sqlCopy, 12
sqlFetch, 14
sqlQuery, 15
sqlSave, 17
sqlTables, 20
sqlTypeInfo, 21
∗Topic **database**
odbc, 2
odbcClose, 3
odbcConnect, 4
odbcErrors, 6
odbcGetInfo, 7
odbcSetAutoCommit, 8
RODBC, 9
RODBCtables, 10
setSqlTypeInfo, 11
sqlCopy, 12
sqlFetch, 14
sqlQuery, 15
sqlSave, 17
sqlTables, 20
sqlTypeInfo, 21
∗Topic **utilities**
odbcDataSources, 6

capabilities, *4*
close.RODBC *(odbcClose)*, 3

getSqlTypeInfo, *19*
getSqlTypeInfo *(setSqlTypeInfo)*, 11

iconv, *4*

odbc, 2
odbcClearError *(odbcErrors)*, 6
odbcClose, 3, *5*
odbcCloseAll *(odbcClose)*, 3
odbcColumns *(odbc)*, 2
odbcConnect, *2, 3*, 4, *6, 7*, *9–11*, *13–18*, *20, 21*
odbcDataSources, 6
odbcDriverConnect *(odbcConnect)*, 4
odbcEndTran *(odbcSetAutoCommit)*, 8
odbcErrors, 6
odbcFetchRows *(odbc)*, 2
odbcGetErrMsg, *2, 3*, *17*
odbcGetErrMsg *(odbcErrors)*, 6
odbcGetInfo, *5*, 7, *9*, *11, 12*, *15*, *20*, *22*
odbcPrimaryKeys *(odbc)*, 2
odbcQuery, *6*, *16, 17*
odbcQuery *(odbc)*, 2
odbcReConnect *(odbcConnect)*, 4
odbcSetAutoCommit, 8
odbcTables *(odbc)*, 2

print.RODBC *(odbcConnect)*, 4

read.table, *16*
RODBC, 9
RODBCtables, 10

setSqlTypeInfo, 11
sqlClear *(RODBCtables)*, 10
sqlColumns, *19*
sqlColumns *(RODBCtables)*, 10
sqlCopy, 12
sqlCopyTable *(sqlCopy)*, 12
sqlDrop *(RODBCtables)*, 10
sqlFetch, *9*, *11*, 14, *17*, *20*

sqlFetchMore *(sqlFetch)*, 14
sqlGetResults, *10*, *13, 14*, *20–22*
sqlGetResults *(sqlQuery)*, 15
sqlPrimaryKeys *(RODBCtables)*, 10
sqlQuery, *2–5*, *11*, *13, 14*, 15, *15*, *20*
sqlSave, *4*, *9*, *11–15*, 17, *17*
sqlTables, *9*, *11*, *17*, 20
sqlTypeInfo, *12*, *19*, 21
sqlUpdate, *10*
sqlUpdate *(sqlSave)*, 17
Sys.localeconv, *9*

type.convert, *17*