inside-R

A Community Site for R – Sponsored by Revolution Analytics

sqlQuery {RODBC}

Query an ODBC Database

`Package:` RODBC
Version: 1.3-13

Description

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

Usage

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

sqlGetResults(channel, as.is = FALSE,  errors = FALSE,
              max = 0, buffsize = 1000,
              nullstring = NA_character_, 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`
: 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 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 SELECTs. 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.

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.

Values

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).

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.
sqlQuery(channel, paste("select State, Murder from USArrests",
                        "where Rape > 30 order by Murder"))
close(channel)
## End(Not run)
```

Author(s)

Michael Lapsley and Brian Ripley

Documentation reproduced from package RODBC, version 1.3-13. License: GPL-2 | GPL-3