Package 'odbc'

October 2, 2017

Title Connect to ODBC Compatible Databases (using the DBI Interface)

Version 1.1.2

Description A DBI-compatible interface to ODBC databases.

License MIT + file LICENSE

URL https://github.com/rstats-db/odbc

BugReports https://github.com/rstats-db/odbc/issues

SystemRequirements C++11, GNU make, An ODBC3 driver manager and drivers.

Roxygen list(markdown = TRUE)

RoxygenNote 6.0.1

LazyData true

ByteCompile true

Imports DBI (>= 0.3.0),
methods,
Rcpp (>= 0.12.11),
blob (>= 1.1.0),
bit64,
hms

Suggests tibble,
DBItest,
testthat,
covr,
magrittr

LinkingTo Rcpp, BH

Collate 'odbc.R'
'Driver.R'
'Connection.R'
'DataTypes.R'
'RcppExports.R'
'Result.R'
'Table.R'
'Viewer.R'
'db_oracle.R'
'hidden.R'
'utils.R'
'zzz.R'

R topics documented:

`odbc-package` *odbc: Connect to ODBC Compatible Databases (using the DBI Interface)*

Description

A DBI-compatible interface to ODBC databases.

Author(s)

Maintainer: Jim Hester <james.hester@rstudio.com>

Authors:

- Hadley Wickham <hadley@rstudio.com>

Other contributors:

- lexicalunit (nanodbc library) [copyright holder]
- Google Inc. (cctz library) [copyright holder]
- RStudio [copyright holder, funder]

See Also

Useful links:

- https://github.com/rstats-db/odbc
- Report bugs at https://github.com/rstats-db/odbc/issues

dbConnect,OdbcDriver-method

Connect to a ODBC compatible database

Description

Connect to a ODBC compatible database

Usage

```
## S4 method for signature 'OdbcDriver'
dbConnect(drv, dsn = NULL, ..., timezone = "UTC",
  encoding = "", bigint = c("integer64", "integer", "numeric", "character"),
  driver = NULL, server = NULL, database = NULL, uid = NULL,
  pwd = NULL, .connection_string = NULL)
```

Arguments

drv	an object that inherits from DBIDriver, or an existing DBIConnection object (in order to clone an existing connection).
dsn	The Data Source Name.
...	Additional ODBC keywords, these will be joined with the other arguments to form the final connection string.
timezone	The Server time zone. Useful if the database has an internal timezone that is *not* 'UTC'. If the database is in your local timezone set to `Sys.timezone()`. See `OlsonNames()` for a complete list of available timezones on your system.
encoding	The text encoding used on the Database. If the database is the same as your local encoding set to `""`. See `iconvlist()` for a complete list of available encodings on your system. Note strings are always returned `UTF-8` encoded.
bigint	The R type that `SQL_BIGINT` types should be mapped to, default is bit64::integer64, which allows the full range of 64 bit integers.
driver	The ODBC driver name.
server	The server hostname.
database	The database on the server.
uid	The user identifier.
pwd	The password to use.
.connection_string	A complete connection string, useful if you are copy pasting it from another source. If this argument is used any additional arguments will be appended to this string.

Details

The connection string keywords are driver dependent. The parameters documented here are common, but some drivers may not accept them. Please see the specific driver documentation for allowed parameters, https://www.connectionstrings.com is also a useful resource of example connection strings for a variety of databases.

dbUnQuoteIdentifier *Un-Quote identifiers*

Description

Call this method to generate a string that is unquoted. This is the inverse of `DBI::dbQuoteIdentifier`.

Usage

```
dbUnQuoteIdentifier(conn, x, ...)

## S4 method for signature 'OdbcConnection,SQL'
dbUnQuoteIdentifier(conn, x)

## S4 method for signature 'OdbcConnection,character'
dbUnQuoteIdentifier(conn, x)
```

Arguments

conn	A subclass of DBIConnection, representing an active connection to an DBMS.
x	A character vector to un-quote.
...	Other arguments passed on to methods.

odbc *Odbc driver*

Description

Driver for an ODBC database.

Usage

```
odbc()
```

Examples

```
## Not run:
#' library(DBI)
odbc::odbc()

## End(Not run)
```

odbc-tables *Convenience functions for reading/writing DBMS tables*

Description

Convenience functions for reading/writing DBMS tables

Usage

```
## S4 method for signature 'OdbcConnection,character,data.frame'
dbWriteTable(conn, name, value,
  overwrite = FALSE, append = FALSE, temporary = FALSE, row.names = NA,
  field.types = NULL, ...)

## S4 method for signature 'OdbcConnection'
sqlData(con, value, row.names = NA, ...)

## S4 method for signature 'OdbcConnection'
sqlCreateTable(con, table, fields,
  field.types = NULL, row.names = NA, temporary = FALSE, ...)
```

Arguments

conn	a OdbcConnection object, produced by DBI::dbConnect()
name	a character string specifying a table name. Names will be automatically quoted so you can use any sequence of characters, not just any valid bare table name.
value	A data.frame to write to the database.
overwrite	Allow overwriting the destination table. Cannot be TRUE if append is also TRUE.
append	Allow appending to the destination table. Cannot be TRUE if overwrite is also TRUE.
temporary	If TRUE, will generate a temporary table statement.
row.names	Either TRUE, FALSE, NA or a string. If TRUE, always translate row names to a column called "row_names". If FALSE, never translate row names. If NA, translate rownames only if they're a character vector. A string is equivalent to TRUE, but allows you to override the default name. For backward compatibility, NULL is equivalent to FALSE.
field.types	Additional field types used to override derived types.
...	Other arguments used by individual methods.
con	A database connection.
table	Name of the table. Escaped with dbQuoteIdentifier().
fields	Either a character vector or a data frame. A named character vector: Names are column names, values are types. Names are escaped with dbQuoteIdentifier(). Field types are unescaped. A data frame: field types are generated using dbDataType().

Examples

```
## Not run:
library(DBI)
con <- dbConnect(odbc::odbc())
dbListTables(con)
dbWriteTable(con, "mtcars", mtcars, temporary = TRUE)
dbReadTable(con, "mtcars")

dbListTables(con)
dbExistsTable(con, "mtcars")

# A zero row data frame just creates a table definition.
dbWriteTable(con, "mtcars2", mtcars[0, ], temporary = TRUE)
dbReadTable(con, "mtcars2")

dbDisconnect(con)

## End(Not run)
```

OdbcConnection *Odbc Connection Methods*

Description

Implementations of pure virtual functions defined in the DBI package for OdbcConnection objects.

Usage

```
## S4 method for signature 'OdbcConnection'
show(object)

## S4 method for signature 'OdbcConnection'
dbIsValid(dbObj, ...)

## S4 method for signature 'OdbcConnection'
dbDisconnect(conn, ...)

## S4 method for signature 'OdbcConnection,character'
dbSendQuery(conn, statement, ...)

## S4 method for signature 'OdbcConnection,character'
dbSendStatement(conn, statement, ...)

## S4 method for signature 'OdbcConnection,ANY'
dbDataType(dbObj, obj, ...)

## S4 method for signature 'OdbcConnection,data.frame'
dbDataType(dbObj, obj, ...)

## S4 method for signature 'OdbcConnection,character'
dbQuoteString(conn, x, ...)
```

```
## S4 method for signature 'OdbcConnection,character'
dbQuoteIdentifier(conn, x, ...)

## S4 method for signature 'OdbcConnection'
dbListTables(conn, ...)

## S4 method for signature 'OdbcConnection,character'
dbExistsTable(conn, name, ...)

## S4 method for signature 'OdbcConnection,character'
dbListFields(conn, name, ...)

## S4 method for signature 'OdbcConnection,character'
dbRemoveTable(conn, name, ...)

## S4 method for signature 'OdbcConnection'
dbGetInfo(dbObj, ...)

## S4 method for signature 'OdbcConnection,character'
dbGetQuery(conn, statement, n = -1, ...)

## S4 method for signature 'OdbcConnection'
dbBegin(conn, ...)

## S4 method for signature 'OdbcConnection'
dbCommit(conn, ...)

## S4 method for signature 'OdbcConnection'
dbRollback(conn, ...)
```

Arguments

`object`	Any R object
`dbObj`	An object inheriting from DBIObject, i.e. DBIDriver, DBIConnection, or a DBIResult
`...`	Other arguments to methods.
`conn`	A DBIConnection object, as returned by `dbConnect()`.
`statement`	a character string containing SQL.
`obj`	An R object whose SQL type we want to determine.
`x`	A character vector to quote as string.
`name`	A character string specifying a DBMS table name.
`n`	maximum number of records to retrieve per fetch. Use `n = -1` or `n = Inf` to retrieve all pending records. Some implementations may recognize other special values.

odbcConnectionActions *List the actions supported for the connection*

Description

Return a list of actions that can be performed on the connection.

Usage

```
odbcConnectionActions(connection)
```

Arguments

connection	A connection object, as returned by dbConnect().

Details

The list returned is a named list of actions, where each action has the following properties:

callback A function to be invoked to perform the action

icon An optional path to an icon representing the action

Value

A named list of actions that can be performed on the connection.

odbcConnectionIcon *Get an icon representing a connection.*

Description

Return the path on disk to an icon representing a connection.

Usage

```
odbcConnectionIcon(connection)
```

Arguments

connection	A connection object, as returned by dbConnect().

Details

The icon returned should be a 32x32 square image file.

Value

The path to an icon file on disk.

odbcDataType *Return the corresponding ODBC data type for an R object*

Description

This is used when creating a new table with `dbWriteTable()`. Databases with default methods defined are

- MySQL
- PostgreSQL
- SQL Server
- SQLite
- Spark
- Hive
- Impala
- Redshift
- Vertica

Usage

```
odbcDataType(con, obj, ...)
```

Arguments

`con`	A driver connection object, as returned by `dbConnect()`.
`obj`	An R object.
`...`	Additional arguments passed to methods.

Details

If you are using a different database and `dbWriteTable()` fails with a SQL parsing error the default method is not appropriate, you will need to write a new method.

Value

Corresponding SQL type for the `obj`.

Defining a new dbDataType method

The object type for your connection will be the database name retrieved by `dbGetInfo(con)$dbms.name`. Use the documentation provided with your database to determine appropriate values for each R data type. An example method definition of a fictional `foo` database follows.

```
con <- dbConnect(odbc::odbc(), "FooConnection")
dbGetInfo(con)$dbms.name
#> [1] "foo"

`odbcDataType.foo <- function(con, obj, ...) {
  switch_type(obj,
```

```
    factor = "VARCHAR(255)",
    datetime = "TIMESTAMP",
    date = "DATE",
    binary = "BINARY",
    integer = "INTEGER",
    double = "DOUBLE",
    character = "VARCHAR(255)",
    logical = "BIT",
    list = "VARCHAR(255)",
    stop("Unsupported type", call. = FALSE)
  )
}
```

OdbcDriver *Odbc Driver Methods*

Description

Implementations of pure virtual functions defined in the DBI package for OdbcDriver objects.

Usage

```
## S4 method for signature 'OdbcDriver'
show(object)

## S4 method for signature 'OdbcDriver,ANY'
dbDataType(dbObj, obj, ...)

## S4 method for signature 'OdbcDriver,list'
dbDataType(dbObj, obj, ...)

## S4 method for signature 'OdbcDriver,data.frame'
dbDataType(dbObj, obj, ...)

## S4 method for signature 'OdbcDriver'
dbIsValid(dbObj, ...)

## S4 method for signature 'OdbcDriver'
dbGetInfo(dbObj, ...)
```

Arguments

object	Any R object
dbObj	A object inheriting from DBIDriver or DBIConnection
obj	An R object whose SQL type we want to determine.
...	Other arguments passed on to methods.

odbcListColumns *List columns in an object.*

Description

Lists the names and types of each column (field) of a specified object.

Usage

```
odbcListColumns(connection, ...)
```

Arguments

connection	A connection object, as returned by dbConnect().
...	Parameters specifying the object.

Details

The object to inspect must be specified as one of the arguments (e.g. `table = "employees"`); depending on the driver and underlying data store, additional specification arguments may be required.

Value

A data frame with `name` and `type` columns, listing the object's fields.

odbcListDataSources *List Available Data Source Names*

Description

List Available Data Source Names

Usage

```
odbcListDataSources()
```

Value

A data frame with two columns.

name Name of the data source

description Data Source description

odbcListDrivers *List Available ODBC Drivers*

Description

List Available ODBC Drivers

Usage

```
odbcListDrivers()
```

Value

A data frame with three columns. If a given driver does not have any attributes the last two columns will be NA.

name Name of the driver

attribute Driver attribute name

value Driver attribute value

odbcListObjects *List objects in a connection.*

Description

Lists all of the objects in the connection, or all the objects which have specific attributes.

Usage

```
odbcListObjects(connection, ...)
```

Arguments

connection	A connection object, as returned by dbConnect().
...	Attributes to filter by.

Details

When used without parameters, this function returns all of the objects known by the connection. Any parameters passed will filter the list to only objects which have the given attributes; for instance, passing schema = "foo" will return only objects matching the schema foo.

Value

A data frame with name and type columns, listing the objects.

odbcListObjectTypes *Return the object hierarchy supported by a connection.*

Description

Lists the object types and metadata known by the connection, and how those object types relate to each other.

Usage

```
odbcListObjectTypes(connection)
```

Arguments

connection	A connection object, as returned by dbConnect().

Details

The returned hierarchy takes the form of a nested list, in which each object type supported by the connection is a named list with the following attributes:

contains A list of other object types contained by the object, or "data" if the object contains data

icon An optional path to an icon representing the type

For instance, a connection in which the top-level object is a schema that contains tables and views, the function will return a list like the following:

```
list(schema = list(contains = list(
                  list(name = "table", contains = "data")
                  list(name = "view", contains = "data"))))
```

Value

The hierarchy of object types supported by the connection.

odbcPreviewObject *Preview the data in an object.*

Description

Return the data inside an object as a data frame.

Usage

```
odbcPreviewObject(connection, rowLimit, ...)
```

Arguments

connection	A connection object, as returned by dbConnect().
rowLimit	The maximum number of rows to display.
...	Parameters specifying the object.

Details

The object to previewed must be specified as one of the arguments (e.g. table = "employees"); depending on the driver and underlying data store, additional specification arguments may be required.

Value

A data frame containing the data in the object.

OdbcResult *Odbc Result Methods*

Description

Implementations of pure virtual functions defined in the DBI package for OdbcResult objects.

Usage

```
## S4 method for signature 'OdbcResult'
dbClearResult(res, ...)

## S4 method for signature 'OdbcResult'
dbFetch(res, n = -1, ...)

## S4 method for signature 'OdbcResult'
dbHasCompleted(res, ...)

## S4 method for signature 'OdbcResult'
dbGetInfo(dbObj, ...)

## S4 method for signature 'OdbcResult'
dbIsValid(dbObj, ...)

## S4 method for signature 'OdbcResult'
dbGetStatement(res, ...)

## S4 method for signature 'OdbcResult'
dbColumnInfo(res, ...)

## S4 method for signature 'OdbcResult'
dbGetRowCount(res, ...)

## S4 method for signature 'OdbcResult'
dbGetRowsAffected(res, ...)
```

```
## S4 method for signature 'OdbcResult'
dbBind(res, params, ...)
```

Arguments

res	An object inheriting from DBIResult.
...	Other arguments passed on to methods.
n	maximum number of records to retrieve per fetch. Use n = -1 or n = Inf to retrieve all pending records. Some implementations may recognize other special values.
dbObj	An object inheriting from DBIObject, i.e. DBIDriver, DBIConnection, or a DBIResult
params	A list of bindings, named or unnamed.

---

odbcSetTransactionIsolationLevel

Set the Transaction Isolation Level for a Connection

Description

Set the Transaction Isolation Level for a Connection

Usage

```
odbcSetTransactionIsolationLevel(conn, levels)
```

Arguments

conn	A DBIConnection object, as returned by dbConnect().
levels	One or more of 'read_uncommitted', 'read_committed', 'repeatable_read', 'serializable'.

See Also

https://docs.microsoft.com/en-us/sql/odbc/reference/develop-app/setting-the-transaction-isolati

Examples

```
## Not run:
  # Can use spaces or underscores in between words.
  odbcSetTransactionIsolationLevel(con, "read uncommitted")

  # Can also use the full constant name.
  odbcSetTransactionIsolationLevel(con, "SQL_TXN_READ_UNCOMMITTED")

## End(Not run)
```

`test_roundtrip` *Test round tripping a simple table*

Description

This tests all the supported data types, including missing values. It first writes them to the database, then reads them back and verifies the data is identical to the original.

Usage

```
test_roundtrip(con = DBItest:::connect(DBItest:::get_default_context()),
  columns = "", invert = TRUE, force_sorted = FALSE)
```

Arguments

`con`	An established DBI connection.
`columns`	Table columns to exclude (default) or include, dependent on the value of `invert`. One of `datetime`, `date`, `binary`, `integer`, `double`, `character`, `logical`.
`invert`	If `TRUE`, change the definition of columns to be inclusive, rather than exclusive.
`force_sorted`	If `TRUE`, a sorted `id` column is added to the sent data, and the received data is sorted by this column before doing the comparison. This is necessary for some databases that do not preserve row order.

Details

This function is not exported and should only be used during tests and as a sanity check when writing new `odbcDataType()` methods.

Examples

```
## Not run:
test_roundtrip(con)

# exclude a few columns
test_roundtrip(con, c("integer", "double"))

# Only test a specific column
test_roundtrip(con, "integer", invert = FALSE)

## End(Not run)
```

Index