Package ‘odbc’

June 5, 2024

Title  Connect to ODBC Compatible Databases (using the DBI Interface)
Version  1.5.0
Description  A DBI-compatible interface to ODBC databases.
License  MIT + file LICENSE
     https://solutions.posit.co/connections/db/
BugReports  https://github.com/r-dbi/odbc/issues
Depends  R (>= 3.6.0)
Imports  bit64, blob (>= 1.2.0), cli, DBI (>= 1.0.0), hms, lifecycle,
         methods, Rcpp (>= 0.12.11), rlang (>= 1.1.0)
Suggests  covr, DBItest, knitr, magrittr, rmarkdown, RSQLite, testthat
         (>= 3.0.0), tibble, withr
LinkingTo  Rcpp
ByteCompile  true
Config/Needs/check  pkgbuild
Config/Needs/website  tidyverse/tidytemplate
Config/testthat/edition  3
Encoding  UTF-8
RoxygenNote  7.3.0
SystemRequirements  GNU make, An ODBC3 driver manager and drivers.
Collate  'RcppExports.R' 'aaa-odbc-data-type.R' 'connection-pane.R'
         'dbi-connection.R' 'odbc-connection.R' 'db.R' 'dbi-driver.R'
         'dbi-result.R' 'dbi-table.R' 'dbi.R' 'driver-access.R'
         'driver-bigquery.R' 'driver-databricks.R' 'driver-db2.R'
         'driver-hana.R' 'driver-hive.R' 'driver-impala.R'
         'driver-mysql.R' 'driver-oracle.R' 'driver-postgres.R'
         'driver-redshift.R' 'driver-snowflake.R' 'driver-spark.R'
         'driver-sql-server.R' 'driver-sqlite.R' 'driver-teradata.R'
         'driver-vertica.R' 'import-standalone-obj-type.R'
         'import-standalone-types-check.R' 'odbc-config.R'
databricks

'odbc.R' 'utils.R' 'zzz.R'

VignetteBuilder knitr

NeedsCompilation yes

Author Jim Hester [aut],
Hadley Wickham [aut, cre],
Oliver Gjoneski [aut],
lexicalunit [cph] (nanodbc library),
Google Inc. [cph] (cctz library),
Posit Software, PBC [cph, fnd]

Maintainer Hadley Wickham <hadley@posit.co>

Repository CRAN

Date/Publication 2024-06-05 16:40:02 UTC

Contents

databricks ................................................................. 2
DBI-tables ................................................................. 4
dbListTables,OdbcConnection-method .................................. 6
isTempTable ............................................................... 7
odbc .......................................................................... 8
odbcConnectionColumns_,Snowflake,character-method ................. 11
odbcDataType ............................................................. 11
odbcListColumns .......................................................... 12
odbcListConfig ........................................................... 13
odbcListDataSources ...................................................... 14
odbcListDrivers ........................................................... 15
odbcListObjects ........................................................... 17
odbcListObjectTypes ...................................................... 18
odbcPreviewObject ......................................................... 19
odbcSetTransactionIsolationLevel ....................................... 19
quote_value ............................................................... 20
snowflake ................................................................. 21

Index ................................................................. 23

---

databricks

Helper for Connecting to Databricks via ODBC
**Description**

Connect to Databricks clusters and SQL warehouses via the **Databricks ODBC driver**.

In particular, the custom `dbConnect()` method for the Databricks ODBC driver implements a subset of the **Databricks client unified authentication** model, with support for personal access tokens, OAuth machine-to-machine credentials, and OAuth user-to-machine credentials supplied via Posit Workbench or the Databricks CLI on desktop. All of these credentials are detected automatically if present using **standard environment variables**.

**Usage**

```r
databricks()

## S4 method for signature 'DatabricksOdbcDriver'

dbConnect(
  drv,
  httpPath,
  workspace = Sys.getenv("DATABRICKS_HOST"),
  useNativeQuery = TRUE,
  driver = NULL,
  HTTPPath,
  uid = NULL,
  pwd = NULL,
...
)
```

**Arguments**

- `drv` an object that inherits from **DBIDriver**, or an existing **DBICconnection** object (in order to clone an existing connection).
- `httpPath`, `HTTPPath` To query a cluster, use the HTTP Path value found under Advanced Options > JDBC/ODBC in the Databricks UI. For SQL warehouses, this is found under Connection Details instead.
- `workspace` The URL of a Databricks workspace, e.g. "https://example.cloud.databricks.com".
- `useNativeQuery` Suppress the driver’s conversion from ANSI SQL 92 to HiveSQL? The default (TRUE), gives greater performance but means that parameterised queries (and hence `dbWriteTable()`) do not work.
- `driver` The name of the Databricks ODBC driver, or NULL to use the default name.
- `uid`, `pwd` Manually specify a username and password for authentication. Specifying these options will disable automated credential discovery.
- `...` Further arguments passed on to `dbConnect()`.

**Value**

An **OdbcConnection** object with an active connection to a Databricks cluster or SQL warehouse.
### Usage

#### S4 method for signature 'OdbcConnection,character,data.frame'

```r
dbWriteTable(
  conn,
  name,
  value,
  overwrite = FALSE,
  append = FALSE,
  temporary = FALSE,
  row.names = NULL,
  field.types = NULL,
  batch_rows =getOption("odbc.batch_rows", NA),
  ...
)
```

#### S4 method for signature 'OdbcConnection,Id,data.frame'

```r
dbWriteTable(
  conn,
  name,
  value,
  overwrite = FALSE,
  append = FALSE,
  temporary = FALSE,
  row.names = NULL,
  field.types = NULL,
  batch_rows =getOption("odbc.batch_rows", NA),
  ...
)
```

#### S4 method for signature 'OdbcConnection,SQL,data.frame'

```r
```
dbWriteTable(
  conn,
  name,
  value,
  overwrite = FALSE,
  append = FALSE,
  temporary = FALSE,
  row.names = NULL,
  field.types = NULL,
  batch_rows = getOption("odbc.batch_rows", NA),
  ...
)

## S4 method for signature 'OdbcConnection'
dbAppendTable(
  conn,
  name,
  value,
  batch_rows = getOption("odbc.batch_rows", NA),
  ..., 
  row.names = NULL
)

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

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.
field.types: Additional field types used to override derived types.
batch_rows: The number of rows to retrieve. Defaults to NA, which is set dynamically to the minimum of 1024 and the size of the input. Depending on the database, driver, dataset and free memory setting this to a lower value may improve performance.

...: Other arguments used by individual methods.
con: A database connection.
table: The table name, passed on to `dbQuoteIdentifier()`. Options are:
- a character string with the unquoted DBMS table name, e.g. "table_name".
- a call to `Id()` with components to the fully qualified table name, e.g. `Id(schema = "my_schema", table = "table_name")`
- a call to `SQL()` with the quoted and fully qualified table name given verbatim, e.g. `SQL("'my_schema'."table_name'")`

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

```r
## 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)
```

---

**Description**

`dbListTables()` provides names of remote tables accessible through this connection; `dbListFields()` provides names of columns within a table.
isTempTable

Usage

```r
## S4 method for signature 'OdbcConnection'
dbListTables(
  conn,
  catalog_name = NULL,
  schema_name = NULL,
  table_name = NULL,
  table_type = NULL,
  ...
)
```

Arguments

- `conn` A `DBIConnection` object, as returned by `dbConnect()`.
- `catalog_name`, `schema_name`, `table_name`
  Catalog, schema, and table names.
  By default, `catalog_name`, `schema_name` and `table_name` will automatically
  escape underscores to ensure that you match exactly one table. If you want to
  search for multiple tables using wild cards, you will need to use `odbcConnectionTables()`
  directly instead.
- `table_type` The type of the table to return, the default returns all table types.
- `...` Other parameters passed on to methods.

Value

A character vector of table or field names respectively.

### isTempTable

`isTempTable` *Helper method used to determine if a table identifier is that of a temporary table.*

Description

Currently implemented only for select back-ends where we have a use for it (SQL Server, for example). Generic, in case we develop a broader use case.

Usage

```r
isTempTable(conn, name, ...)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>conn</td>
<td>OdbcConnection</td>
</tr>
<tr>
<td>name</td>
<td>Table name</td>
</tr>
<tr>
<td>...</td>
<td>additional parameters to methods</td>
</tr>
</tbody>
</table>

odbc

Connect to a database via an ODBC driver

Description

The `dbConnect()` method documented here is invoked when `DBI::dbConnect()` is called with the first argument `odbc()`. Connecting to a database via an ODBC driver is likely the first step in analyzing data using the `odbc` package; for an overview of package concepts, see the Overview section below.

Usage

```r
odbc()
```

## S4 method for signature 'OdbcDriver'

```r
dbConnect(
  drv,
  dsn = NULL,
  ...,
  timezone = "UTC",
  timezone_out = "UTC",
  encoding = "",
  bigint = c("integer64", "integer", "numeric", "character"),
  timeout = 10,
  driver = NULL,
  server = NULL,
  database = NULL,
  uid = NULL,
  pwd = NULL,
  dbms.name = NULL,
  attributes = NULL,
  interruptible = getOption("odbc.interruptible", interactive()),
  .connection_string = NULL
)
```

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| drv      | An OdbcDriver, from `odbc()`.
| dsn      | The data source name. For currently available options, see the name column of `odbcListDataSources()` output. |
... Additional ODBC keywords. These will be joined with the other arguments to form the final connection string.

Note that ODBC parameter names are case-insensitive so that (e.g.) DRV and drv are equivalent. Since this is different to R and a possible source of confusion, odbc will error if you supply multiple arguments that have the same name when case is ignored.

Any values containing a leading or trailing space, a =, ;, {, or } are likely to require quoting. Use `quote_value()` for a fairly standard approach or see your driver documentation for specifics.

timezone The server time zone. Useful if the database has an internal time zone that is not 'UTC'. If the database is in your local time zone, set this argument to `Sys.timezone()`. See `OlsonNames()` for a complete list of available time zones on your system.

timezone_out The time zone returned to R. If you want to display datetime values in the local timezone, set to `Sys.timezone()`.

encoding The text encoding used on the Database. If the database is not using UTF-8 you will need to set the encoding to get accurate re-encoding. 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.

timeout Time in seconds to timeout the connection attempt. Setting a timeout of Inf indicates no timeout. Defaults to 10 seconds.

driver The ODBC driver name or a path to a driver. For currently available options, see the name column of `odbcListDrivers()` output.

server The server hostname. Some drivers use Servername as the name for this argument. Not required when configured for the supplied dsn.

database The database on the server. Not required when configured for the supplied dsn.

uid The user identifier. Some drivers use username as the name for this argument. Not required when configured for the supplied dsn.

pwd The password. Some drivers use password as the name for this argument. Not required when configured for the supplied dsn.

dbms.name The database management system name. This should normally be queried automatically by the ODBC driver. This name is used as the class name for the OdbcConnection object returned from `dbConnect()`. However, if the driver does not return a valid value, it can be set manually with this parameter.

attributes A list of connection attributes that are passed prior to the connection being established. See `ConnectionAttributes`.

interruptible Logical. If TRUE calls to SQLExecute and SQLExecuteDirect can be interrupted when the user sends SIGINT (ctrl-c). Otherwise, they block. Defaults to TRUE in interactive sessions, and FALSE otherwise. It can be set explicitly either by manipulating this argument, or by setting the global option odbc.interruptible.
.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.

Connection strings

Internally, dbConnect() creates a connection string using the supplied arguments. Connection string keywords are driver-dependent; the arguments documented here are common, but some drivers may not accept them.

Alternatively to configuring DSNs and driver names with the driver manager, you can pass a complete connection string directly as the .connection_string argument. The Connection Strings Reference is a useful resource that has example connection strings for a large variety of databases.

Overview

The odbc package is one piece of the R interface to databases with support for ODBC:

The package supports any Database Management System (DBMS) with ODBC support. Support for a given DBMS is provided by an ODBC driver, which defines how to interact with that DBMS using the standardized syntax of ODBC and SQL. Drivers can be downloaded from the DBMS vendor or, if you’re a Posit customer, using the professional drivers. To manage information about each driver and the data sources they provide access to, our computers use a driver manager. Windows is bundled with a driver manager, while MacOS and Linux require installation of one; this package supports the unixODBC driver manager.

In the R interface, the DBI package provides a front-end while odbc implements a back-end to communicate with the driver manager. The odbc package is built on top of the nanodbc C++ library.

Interfacing with DBMSs using R and odbc involves three high-level steps:

1. Configure drivers and data sources: the functions odbcListDrivers() and odbcListDataSources() help to interface with the driver manager.
2. Connect to a database: The dbConnect() function, called with the first argument odbc(), connects to a database using the specified ODBC driver to create a connection object.
3. Interface with connections: The resulting connection object can be passed to various functions to retrieve information on database structure (dbListTables()), iteratively develop queries (dbSendQuery(), dbColumnInfo()), and query data objects (dbFetch()).

Learn more

To learn more about databases:

- "Best Practices in Working with Databases" documents how to use the odbc package with various popular databases.
- The pyodbc "Drivers and Driver Managers" Wiki provides further context on drivers and driver managers.
- Microsoft’s "Introduction to ODBC" is a thorough resource on the ODBC interface.
Connecting to Snowflake via ODBC

Description

\texttt{odbcConnectionColumns()}:

If the catalog, or the schema arguments are NULL, attempt to infer by querying for \texttt{CURRENT\_DATABASE()} and \texttt{CURRENT\_SCHEMA()}. We do this to aid with performance, as the \texttt{SQLColumns} method is more performant when restricted to a particular DB/schema.

Usage

```r
## S4 method for signature 'Snowflake,character'
dbExistsTableForWrite(conn, name, ..., catalog_name = NULL, schema_name = NULL)
```

Arguments

- \texttt{conn}
  - A \texttt{DBIConnection} object, as returned by \texttt{dbConnect()}.
- \texttt{name}
  - The table name, passed on to \texttt{dbQuoteIdentifier()}. Options are:
    - a character string with the unquoted DBMS table name, e.g. "table\_name".
    - a call to \texttt{Id()} with components to the fully qualified table name, e.g. \texttt{Id(schema = "my\_schema", table = "table\_name")}
    - a call to \texttt{SQL()} with the quoted and fully qualified table name given verbatim, e.g. \texttt{SQL("my\_schema"."table\_name")}
- \texttt{...}
  - Other parameters passed on to methods.
- \texttt{catalog\_name, schema\_name}
  - Catalog and schema names.

Return the corresponding ODBC data type for an R object

Description

This is used when creating a new table with \texttt{dbWriteTable()}. Databases with default methods defined are:

- MySQL
- PostgreSQL
- SQL Server
- Oracle
- SQLite
• Spark
• Hive
• Impala
• Redshift
• Vertica
• BigQuery
• Teradata
• Access
• Snowflake

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. The object type for your method 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.

Value

Corresponding SQL type for the obj.

Description

List columns in an 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.

```r
odbcListConfig

List locations of ODBC configuration files
```

Description

On MacOS and Linux, odbc uses the unixODBC driver manager to manage information about driver and data sources. This helper returns the filepaths where the driver manager will look for that information.

This function is a wrapper around the command line call `odbcinst -j`.

Windows does not use `.ini` configuration files; this function will return a 0-length vector on Windows.

Usage

```
odbcListConfig()
```

See Also

The `odbcListDrivers()` and `odbcListDataSources()` helpers return information on the contents of `odbcinst.ini` and `odbc.ini` files, respectively.

Learn more about unixODBC and the `odbcinst` utility [here](#).

Examples

```
configs <- odbcListConfig()

file.edit(configs[1])
```
odbcListDataSources  

List Configured Data Source Names

Description

Collect information about the available data source names (DSNs). A DSN must be both installed and configured with the driver manager to be included in this list. Configuring a DSN just sets up a lookup table (e.g. in odbc.ini) to allow users to pass only the DSN to `dbConnect()`.

DSNs that are not configured with the driver manager can still be connected to with `dbConnect()` by providing DSN metadata directly.

Usage

odbcListDataSources()

Value

A data frame with two columns:

- **name**  Name of the data source. The entries in this column can be passed to the dsn argument of `dbConnect()`.
- **description**  Data source description.

Configuration

This function interfaces with the driver manager to collect information about the available data source names.

For MacOS and Linux, the odbc package supports the unixODBC driver manager. unixODBC looks to the odbc.ini configuration file for information on DSNs. Find the location(s) of your odbc.ini file(s) with `odbcinst -j`.

In this example odbc.ini file:

```
[MySQL]
Driver = MySQL Driver
Database = test
Server = 127.0.0.1
User = root
password = root
Port = 3306
```

...the data source name is MySQL, which will appear in the name column of this function’s output. To pass the DSN as the dsn argument to `dbConnect()`, pass it as a string, like "MySQL". Driver = MySQL Driver references the driver name in `odbcListDrivers()` output.

Windows is bundled with an ODBC driver manager.

When a DSN is configured with a driver manager, information on the DSN will be automatically passed on to `dbConnect()` when its dsn argument is set.
For example, with the MySQL data source name configured, and the driver name MySQL Driver appearing in `odbcListDrivers()` output, the code:

```r
con <-
  dbConnect(
    odbc::odbc(),
    Driver = "MySQL Driver",
    Database = "test",
    Server = "127.0.0.1",
    User = "root",
    password = "root",
    Port = 3306
  )

...can be written:

con <- dbConnect(odbc::odbc(), dsn = "MySQL")
```

In this case, `dbConnect()` will look up the information defined for MySQL in the driver manager (in our example, `odbc.ini`) and automatically pass the needed arguments.

**See Also**

`odbcListDrivers()`

---

**Description**

Collect information about the configured driver names. A driver must be both installed and configured with the driver manager to be included in this list. Configuring a driver name just sets up a lookup table (e.g. in `odbcinst.ini`) to allow users to pass only the driver name to `dbConnect()`.

Driver names that are not configured with the driver manager (and thus do not appear in this function’s output) can still be used in `dbConnect()` by providing a path to a driver directly.

**Usage**

```r
odbcListDrivers(
  keep =getOption("odbc.drivers_keep"),
  filter =getOption("odbc.drivers_filter")
)
```
Arguments

keep, filter A character vector of driver names to keep in or remove from the results, respectively. If NULL, all driver names will be kept, or none will be removed, respectively. The odbc.drivers_keep and odbc.drivers_filter options control the argument defaults.

Driver names are first processed with keep, then filter. Thus, if a driver name is in both keep and filter, it won’t appear in output.

Value

A data frame with three columns.

name Name of the driver. The entries in this column can be passed to the driver argument of dbConnect() (as long as the driver accepts the argument).

attribute Driver attribute name.

driver attribute value.

value Driver attribute value.

If a driver has multiple attributes, there will be one row per attribute, each with the same driver name. If a given driver name does not have any attributes, the function will return one row with the driver name, but the last two columns will be NA.

Configuration

This function interfaces with the driver manager to collect information about the available driver names.

For MacOS and Linux, the odbc package supports the unixODBC driver manager. unixODBC looks to the odbcinst.ini configuration file for information on driver names. Find the location(s) of your odbcinst.ini file(s) with odbcinst -j.

In this example odbcinst.ini file:

```
[MySQL Driver]
Driver=/opt/homebrew/Cellar/mysql/8.2.0_1/lib/libmysqlclient.dylib
```

Then the driver name is MySQL Driver, which will appear in the name column of this function’s output. To pass the driver name as the driver argument to dbConnect(), pass it as a string, like "MySQL Driver".

Windows is bundled with an ODBC driver manager.

In this example, function output would include 1 row: the name column would read "MySQL Driver", attribute would be "Driver", and value would give the file path to the driver. Additional key-value pairs under the driver name would add additional rows with the same name entry.

When a driver is configured with a driver manager, information on the driver will be automatically passed on to dbConnect() when its driver argument is set. For an example, see the same section in the odbcListDataSources() help-file. Instead of configuring driver information with a driver manager, it is also possible to provide a path to a driver directly to dbConnect().
odbcListObjects

See Also

odbcListDataSources()

Examples

odbcListDrivers()

---

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 re-
quired.

Value
A data frame containing the data in the object.

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'.

quote_value

See Also


Examples

```r
## 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)
```

Description

When connecting to a database using odbc, all the arguments are concatenated into a single connection string that looks like `name1=value1;name2=value2`. That means if your value contains `=` or `;` then it needs to be quoted. Other rules mean that you need to quote any values that starts or ends with white space, or contains `{` or `}`.

This function quotes a string in a way that should work for most drivers, but unfortunately there doesn’t seem to be an approach that works everywhere. If this function doesn’t work for you, you’ll need to carefully read the docs for your driver.

Usage

```r
quote_value(x)
```

Arguments

- `x` A string to quote.

Value

A quoted string, wrapped in `I()`.

Examples

```r
quote_value("abc")
quote_value("ab\'c")
```

# Real usage is more likely to look like:
```r
## Not run:
library(DBI)
```
snowflake

con <- dbConnect(
    odbc::odbc(),
    dsn = "reallycooldatabase",
    password = odbc::quote_value(Sys.getenv("MY_PASSWORD"))
)

## End(Not run)

snowflake

**Helper for connecting to Snowflake via ODBC**

### Description

Connect to a Snowflake account via the Snowflake ODBC driver.

In particular, the custom `dbConnect()` method for the Snowflake ODBC driver detects ambient OAuth credentials on platforms like Snowpark Container Services or Posit Workbench.

### Usage

```r
snowflake()
```

#### S4 method for signature 'Snowflake'

```r
dbConnect(
    drv,
    account = Sys.getenv("SNOWFLAKE_ACCOUNT"),
    driver = NULL,
    warehouse = NULL,
    database = NULL,
    schema = NULL,
    uid = NULL,
    pwd = NULL,
    ...
)
```

### Arguments

- **drv**: an object that inherits from `DBIDriver`, or an existing `DBIConnection` object (in order to clone an existing connection).
- **account**: A Snowflake account identifier, e.g. "testorg-test_account".
- **driver**: The name of the Snowflake ODBC driver, or NULL to use the default name.
- **warehouse**: The name of a Snowflake compute warehouse, or NULL to use the default.
- **database**: The name of a Snowflake database, or NULL to use the default.
- **schema**: The name of a Snowflake database schema, or NULL to use the default.
- **uid, pwd**: Manually specify a username and password for authentication. Specifying these options will disable ambient credential discovery.
- **...**: Further arguments passed on to `dbConnect()`.  

---

```r
```
Value

An OdbcConnection object with an active connection to a Snowflake account.

Examples

```r
## Not run:
# Use ambient credentials.
DBI::dbConnect(odbc::snowflake())

# Use browser-based SSO (if configured). Only works on desktop.
DBI::dbConnect(
  odbc::snowflake(),
  account = "testorg-test_account",
  authenticator = "externalbrowser"
)

# Use a traditional username & password.
DBI::dbConnect(
  odbc::snowflake(),
  account = "testorg-test_account",
  uid = "me",
  pwd = rstudioapi::askForPassword()
)

## End(Not run)
```
Index

bit64::integer64, 9
ConnectionAttributes, 9
databricks, 2
DatabricksOdbcDriver-class (databricks), 2
dbAppendTable,OdbcConnection-method (DBI-tables), 4
dbColumnInfo(), 10
dbConnect (odbc), 8
dbConnect(), , 9–11, 14–16, 19, 21
dbConnect,DatabricksOdbcDriver-method (databricks), 2
dbConnect,OdbcDriver-method (odbc), 8
dbConnect,Snowflake-method (snowflake), 21
dbDataType(), 6
dbExistsTableForWrite,Snowflake,character-method (odbcConnectionColumns_,Snowflake,character-method), 11
dbFetch(), 10
DBI-tables, 4
DBI::dbConnect(), 5, 8
DBIConnection, 3, 7, 11, 19, 21
DBIDriver, 3, 21
dbListTables(), 10
dbListTables,OdbcConnection-method, 6
dbQuoteIdentifier(), 6, 11
dbSendQuery(), 10
dbWriteTable,OdbcConnection,character,data.frame-method (DBI-tables), 4
dbWriteTable,OdbcConnection,Id,data.frame-method (DBI-tables), 4
dbWriteTable,OdbcConnection,SQL,data.frame-method (DBI-tables), 4

iconvlist(), 9
Id(), 6, 11
isTempTable, 7

iseTempTable,OdbcConnection,Id-method (isTempTable), 7
iseTempTable,OdbcConnection,SQL-method (isTempTable), 7

odbc, 8
OdbcConnection, 5
odbcConnectionColumns_,Snowflake,character-method, 11
odbcDataType, 11
odbcDataType,ACCESS-method (odbcDataType), 11
odbcDataType,ANY-method (odbcDataType), 11
odbcDataType,BigQuery-method (odbcDataType), 11
odbcDataType,Hive-method (odbcDataType), 11
odbcDataType,Impala-method (odbcDataType), 11
odbcDataType,Microsoft SQL Server-method (odbcDataType), 11
odbcDataType,MySQL-method (odbcDataType), 11
odbcDataType,Oracle-method (odbcDataType), 11
odbcDataType,PostgreSQL-method (odbcDataType), 11
odbcDataType,Redshift-method (odbcDataType), 11
odbcDataType,Snowflake-method (odbcDataType), 11
odbcDataType,Spark SQL-method (odbcDataType), 11
odbcDataType,SQLite-method (odbcDataType), 11
odbcDataType,Teradata-method (odbcDataType), 11
odbcDataType,Vertica Database-method (odbcDataType), 11
odbcListColumns, 12
odbcListConfig, 13
odbcListDataSources, 14
odbcListDataSources(), 8, 10, 13, 16, 17
odbcListDrivers, 15
odbcListDrivers(), 9, 10, 13–15
odbcListObjects, 17
odbcListObjectTypes, 18
odbcPreviewObject, 19
odbcSetTransactionIsolationLevel, 19
OlsonNames(), 9
quote_value, 20
quote_value(), 9
snowflake, 21
SQL(), 6, 11
sqlCreateTable, OdbcConnection-method
  (DBI-tables), 4
Sys.timezone(), 9