Package ‘odbc’

January 23, 2024

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

**Version**  1.4.2

**Description**  A DBI-compatible interface to ODBC databases.

**License**  MIT + file LICENSE

**URL**  [https://odbc.r-dbi.org](https://odbc.r-dbi.org), [https://github.com/r-dbi/odbc](https://github.com/r-dbi/odbc), [https://solutions.posit.co/connections/db/](https://solutions.posit.co/connections/db/)

**BugReports**  [https://github.com/r-dbi/odbc/issues](https://github.com/r-dbi/odbc/issues)

**Depends**  R (>= 3.6.0)

**Imports**  bit64, blob (>= 1.2.0), DBI (>= 1.0.0), hms, lifecycle, methods, Rcpp (>= 0.12.11), rlang (>= 0.2.0)

**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' 'connection-pane.R' 'dbi-connection.R'

'odbc-connection.R' 'db.R' 'dbi-driver.R' 'dbi-result.R'

'db-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'


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 2024-01-22 23:20:02 UTC

R topics documented:

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

Index 22

---

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 sub-
set 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

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 DBIConnection 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 de-
   fault (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.

Examples

## Not run:
DBI::dbConnect(
  odbc::databricks(),
  httpPath = "sql/protocolv1/o/4425955464597947/1026-023828-vn51jugj"
)

## End(Not run)
DBI-tables  Convenience functions for reading/writing DBMS tables

Description
Convenience functions for reading/writing DBMS tables

Usage

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

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

```r
## S4 method for signature 'OdbcConnection,SQL,data.frame'
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.
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.
batch_rows The number of rows to retrieve. Defaults to NA, which is set dynamically to 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.
**Usage**

```r
## S4 method for signature 'OdbcConnection'
# dbListTables()

# Example:
con <- dbConnect(odbc::odbc())
dbListTables(con)
```

**Description**

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

```
schema_name = NULL,
table_name = NULL,
table_type = NULL,
...
)

## S4 method for signature 'OdbcConnection,character'
dblistFields(
  conn,
  name,
  catalog_name = NULL,
  schema_name = NULL,
  column_name = 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.
name 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 verba-
     tim, e.g. `SQL("'my_schema'.'table_name'")`
column_name The name of the column to return, the default returns all columns.

Value

A character vector of table or field names respectively.

---

isTempTable Helper method used to determine if a table identifier is that of a tem-
porary 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

isTempTable(conn, name, ...)

## S4 method for signature 'OdbcConnection,Id'
isTempTable(conn, name, ...)

## S4 method for signature 'OdbcConnection,SQL'
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 \texttt{dbConnect()} method documented here is invoked when \texttt{DBI::dbConnect()} is called with the first argument \texttt{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

odbc()

## S4 method for signature 'OdbcDriver'
dbConnect(
  drv,
  dsn = NULL,
  ..., timezone = "UTC",
  timezone_out = "UTC",
  encoding = "",
  bigint = c("integer64", "integer", "numeric", "character"),
  timeout = 10,
  driver = NULL,
  server = NULL,
```r

database = NULL,
uid = NULL,
pwd = NULL,
dbms.name = NULL,
attributes = NULL,
.connection_string = NULL
)

Arguments

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 timezone that
is not 'UTC'. If the database is in your local timezone, 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 argu-
ment. 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.
```
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
An S4 object of connection attributes that are passed prior to the connection being established. See `ConnectionAttributes`.

.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.

---

### 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
- Oracle
- SQLite
- Spark
- Hive
- Impala
- Redshift
- Vertica
- BigQuery
- Teradata
- Access

#### 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.

```r
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)
  )
}
```

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.

---

**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

```r
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
```r
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".

**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:

```r
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")
)"
odbcListDrivers

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.

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.
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:

```r
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.

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

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)
```

### quote_value

Describe special character when connecting

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
quote_value(x)

Arguments
- x A string to quote.

Value
A quoted string, wrapped in I().

Examples
quote_value("abc")
quote_value("ab\'c")

# Real usage is more likely to look like:
## Not run:
library(DBI)
con <- dbConnect(
    odbc::odbc(),
    dsn = "reallycooldatabase",
    password = odbc::quote_value(Sys.getenv("MY_PASSWORD"))
)

## End(Not run)
Index

bit64::integer64, 9

ConnectionAttributes, 10

databricks, 2
DatabricksOdbcDriver-class, (databricks), 2
dbAppendTable, OdbcConnection-method (DBI-tables), 4
dbColumnInfo(), 10
dbConnect (odbc), 8
dbConnect(), 3, 7, 10, 14–16, 19
dbConnect, DatabricksOdbcDriver-method (databricks), 2
dbConnect, OdbcDriver-method (odbc), 8
dbDataType(), 6
dbFetch(), 10
DBI::dbConnect(), 5, 8
DBIConnection, 3, 7, 19
DBIDriver, 3
dbListFields, OdbcConnection, character-method
(dblistTables, OdbcConnection-method), 6
dbListTables(), 10
dbListTables, OdbcConnection-method, 6
dbQuoteIdentifier, 6, 7
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, 7
isTempTable, 7
isTempTable, OdbcConnection, Id-method (isTempTable), 7

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

odbc, 8
OdbcConnection, 5
odbcDataType, 11
odbcListColumns, 12
odbcListConfig, 13
odbcListDataSources()(), 9, 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

SQL(), 6, 7
sqlCreateTable, OdbcConnection-method (DBI-tables), 4
Sys.timezone(), 9