Package ‘odbc’

June 29, 2023

Title  Connect to ODBC Compatible Databases (using the DBI Interface)
Version  1.3.5
Description  A DBI-compatible interface to ODBC databases.
License  MIT + file LICENSE
URL  https://r-dbi.github.io/odbc/, https://github.com/r-dbi/odbc,
https://db.rstudio.com
BugReports  https://github.com/r-dbi/odbc/issues
Depends  R (>= 3.2.0)
Imports  bit64,
blob (>= 1.2.0),
DBI (>= 1.0.0),
hms,
methods,
Rcpp (>= 0.12.11),
rlang (>= 0.2.0)
Suggests  covr,
DBItest,
magrittr,
RSQLite,
testthat,
tibble
LinkingTo  Rcpp
ByteCompile  true
Config/Needs/website  tidyverse/tidytemplate
Config/Needs/check  pkgbuild
Encoding  UTF-8
Roxygen  list(markdown = TRUE)
RoxygenNote  7.2.3
SystemRequirements  GNU make, An ODBC3 driver manager and drivers.
Collate  ‘odbc.R’
‘Driver.R’
‘Connection.R’
‘DataTypes.R’
‘RcppExports.R’
odbc-package

'A Result.R'
'Table.R'
'Viewer.R'
'db.R'
'hidden.R'
'utili.R'
'zzz.R'

R topics documented:

odbc-package .......................................................... 2
dbConnect,OdbcDriver-method ..................................... 3
dbListFields,OdbcConnection,character-method ................... 5
dbListTables,OdbcConnection-method ............................... 6
dbListTables-methods .................................................. 7
dbQuoteString-methods ............................................... 8
odbc ................................................................. 8
odbc-tables ............................................................ 8
OdbcConnection .......................................................... 11
odbcConnectionActions ................................................ 13
odbcConnectionCatalogs ............................................... 13
odbcConnectionColumns ............................................. 14
odbcConnectionIcon ................................................... 16
odbcConnectionSchemas ............................................. 16
odbcConnectionTables ............................................... 17
odbcConnectionTableTypes .......................................... 18
odbcDataType .......................................................... 19
OdbcDriver ............................................................. 20
odbcListColumns ...................................................... 21
odbcListDataSources ................................................ 21
odbcListDrivers ....................................................... 22
odbcListObjects ....................................................... 22
odbcListObjectTypes ................................................ 23
odbcPreviewObject .................................................... 24
odbcPreviewQuery ..................................................... 24
OdbcResult .............................................................. 25
odbcSetTransactionIsolationLevel ................................ 26
sqlCreateTable-methods ............................................. 26
SUPPORTED_CONNECTION_ATTRIBUTES ............................... 27

Index 28

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

Description

A DBI-compatible interface to ODBC databases.
Author(s)

**Maintainer:** Hadley Wickham <hadley@rstudio.com>

Authors:

- Jim Hester
- Oliver Gjoneski

Other contributors:

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

See Also

Useful links:

- [https://r-dbi.github.io/odbc/](https://r-dbi.github.io/odbc/)
- [https://github.com/r-dbi/odbc](https://github.com/r-dbi/odbc)
- [https://db.rstudio.com](https://db.rstudio.com)
- Report bugs at [https://github.com/r-dbi/odbc/issues](https://github.com/r-dbi/odbc/issues)

---

**dbConnect, OdbcDriver-method**

*Connect to a ODBC compatible database*

**Description**

Connect to a ODBC compatible database

**Usage**

```r
## 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,
  database = NULL,
  uid = NULL,
  pwd = NULL,
  dbms.name = NULL,
  attributes = 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.
- **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.
- **server**: The server hostname.
- **database**: The database on the server.
- **uid**: The user identifier.
- **pwd**: The password to use.
- **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 `OdbcConnect` 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.

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](https://www.connectionstrings.com) is also a useful resource of example connection strings for a variety of databases.
## 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()`.
- **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 verbatim, e.g. `SQL("my_schema"."table_name")`
- **catalog_name**: The name of the catalog to return, the default returns all catalogs.
- **schema_name**: The name of the schema to return, the default returns all schemas.
- **column_name**: The name of the column to return, the default returns all columns.
- **...**: Other parameters passed on to methods.

### Details

% can be used as a wildcard in any of the search parameters to match 0 or more characters. _ can be used to match any single character.

### Value

`dbListFields()` returns a character vector that enumerates all fields in the table in the correct order. This also works for temporary tables if supported by the database. The returned names are suitable for quoting with `dbQuoteIdentifier()`.

### Failure modes

If the table does not exist, an error is raised. Invalid types for the name argument (e.g., character of length not equal to one, or numeric) lead to an error. An error is also raised when calling this method for a closed or invalid connection.
Specification

The name argument can be

- a string
- the return value of `dbQuoteIdentifier()`
- a value from the table column from the return value of `dbListObjects()` where `is_prefix` is `FALSE`

A column named `row_names` is treated like any other column.

See Also

`dbColumnInfo()` to get the type of the fields.

Other DBIConnection generics: `DBIConnection-class`, `dbAppendTable()`, `dbCreateTable()`, `dbDataType()`, `dbDisconnect()`, `dbExecute()`, `dbExistsTable()`, `dbGetException()`, `dbGetInfo()`, `dbGetQuery()`, `dbIsReadOnly()`, `dbIsValid()`, `dbListObjects()`, `dbListResults()`, `dbListTables()`, `dbReadTable()`, `dbRemoveTable()`, `dbSendQuery()`, `dbSendStatement()`, `dbWriteTable()`

Examples

```r
con <- dbConnect(RSQLite::SQLite(), "::memory:")
dbWriteTable(con, "mtcars", mtcars)
dbListFields(con, "mtcars")
dbDisconnect(con)
```

---

**Description**

Returns the unquoted names of remote tables accessible through this connection. This should include views and temporary objects, but not all database backends (in particular `RMariaDB` and `RMySQL`) support this.

**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  The name of the catalog to return, the default returns all catalogs.
schema_name   The name of the schema to return, the default returns all schemas.
table_name    The name of the table to return, the default returns all tables.
table_type    The type of the table to return, the default returns all table types.
...           Other parameters passed on to methods.

Details

% can be used as a wildcard in any of the search parameters to match 0 or more characters. _ can be used to match any single character.

Value

dbListTables() returns a character vector that enumerates all tables and views in the database. Tables added with dbWriteTable() are part of the list. As soon a table is removed from the database, it is also removed from the list of database tables.
The same applies to temporary tables if supported by the database.
The returned names are suitable for quoting with dbQuoteIdentifier().

Failure modes

An error is raised when calling this method for a closed or invalid connection.

See Also

The ODBC documentation on Pattern Value Arguments for further details on the supported syntax.

Examples

con <- dbConnect(RSQLite::SQLite(), "::memory:")
dbListTables(con)
dbWriteTable(con, "mtcars", mtcars)
dbListTables(con)
dbDisconnect(con)

dbListTables-methods  ~ Methods for Function dbListTables in Package DBI ~

dbListTables-methods  ~ Methods for Function dbListTables in Package DBI ~
dbQuoteString-methods  ~~ Methods for Function dbQuoteString in Package DBI ~~

Description

~~ Methods for function dbQuoteString in package DBI ~~

Methods

signature(conn = "DBIConnection", x = "ANY")
signature(conn = "DBIConnection", x = "character")
signature(conn = "DBIConnection", x = "SQL")
signature(conn = "Hive", x = "character")

odbc  

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,
  batch_rows = getOption("odbc.batch_rows", NA),
  
)
```

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

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

## S4 method for signature 'OdbcConnection'
```
dbAppendTable(conn, name, value, ..., row.names = NULL)
```

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

## 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.

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.

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

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

```
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, params = NULL, ..., immediate = FALSE)

## S4 method for signature 'OdbcConnection,character'
dbSendStatement(conn, statement, params = NULL, ..., immediate = FALSE)

## 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'

dbQuoteIdentifier(conn, x, ...)

## S4 method for signature 'OdbcConnection,SQL'

dbQuoteIdentifier(conn, x, ...)

## 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, params = NULL, ...)

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

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

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

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

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

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

### 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.

- **params**
  - Query parameters to pass to `dbBind()`. See `dbBind()` for details.

- **immediate**
  - If `TRUE`, SQLExecDirect will be used instead of SQLPrepare, and the `params` argument is ignored.

- **obj**
  - An R object whose SQL type we want to determine.

- **x**
  - A character vector, `SQL` or `Id` object to quote as identifier.

- **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 verbatim, e.g. `SQL("my_schema":"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

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.

odbcConnectionCatalogs

Description
This function returns a listing of available catalogs.

Usage
odbcConnectionCatalogs(conn)

## S4 method for signature 'OdbcConnection'
odbcConnectionCatalogs(conn)

Arguments
conn OdbcConnection
odbcConnectionColumns

Description

For a given table this function returns detailed information on all fields / columns. The expectation is that this is a relatively thin wrapper around the ODBC SQLColumns function call, with some of the field names renamed / re-ordered according to the return specifications below.

Usage

odbcConnectionColumns(conn, name, ...)

## S4 method for signature 'OdbcConnection,Id'
odbcConnectionColumns(conn, name, column_name = NULL)

## S4 method for signature 'OdbcConnection,character'
odbcConnectionColumns(
  conn, 
  name, 
  catalog_name = NULL, 
  schema_name = NULL, 
  column_name = NULL
)

## S4 method for signature 'OdbcConnection,SQL'
odbcConnectionColumns(conn, name, ...)

## S4 method for signature 'Oracle,character'
odbcConnectionColumns(
  conn, 
  name, 
  catalog_name = NULL, 
  schema_name = NULL, 
  column_name = NULL
)

Arguments

- conn: OdbcConnection
- name: table we wish to get information on
- ...: additional parameters to methods
- column_name: The name of the column to return, the default returns all columns.
- catalog_name: character catalog where the table is located
- schema_name: character schema where the table is located
Details

In `dbWriteTable()` we make a call to this method to get details on the fields of the table we are writing to. In particular the columns `data_type`, `column_size`, and `decimal_digits` are used. An implementation is not necessary for `dbWriteTable()` to work.

`odbcConnectionColumns` is routed through the SQLColumns ODBC method. This function, together with remaining catalog functions (SQLTables, etc), by default (`SQL_ATTR_METADATA_ID == false`) expect either ordinary arguments (OA) in the case of the catalog, or pattern value arguments (PV) in the case of schema/table name. For these, quoted identifiers do not make sense, so we unquote identifiers prior to the call.

Query, rather than use SQLColumns ODBC API for ORACLE since when using the API we bind a BIGINT to one of the column results. Oracle’s OEM driver is unable to handle.

Value

data.frame with columns

- name
- field.type - equivalent to type_name in SQLColumns output
- table_name
- schema_name
- catalog_name
- data_type
- column_size
- buffer_length
- decimal_digits
- numeric_precision_radix
- column_default
- sql_data_type
- sql_datetime_subtype
- char_octet_length
- ordinal_position
- nullable

See Also

The ODBC documentation on SQLColumns for further details.

The ODBC documentation on Arguments to catalog functions.
### odbcConnectionIcon

*Get an icon representing a connection.*

**Description**

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

**Usage**

```r
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.

### odbcConnectionSchemas

This function returns a listing of available schemas.

**Usage**

```r
odbcConnectionSchemas(conn, catalog_name)
```

#### S4 method for signature 'OdbcConnection,ANY'

```r
odbcConnectionSchemas(conn, catalog_name = NULL)
```

#### S4 method for signature 'OdbcConnection,character'

```r
odbcConnectionSchemas(conn, catalog_name)
```

**Arguments**

- `conn` OdbcConnection
- `catalog_name` Catalog where we are looking to list schemas.

**Details**

Currently, for a generic connection the `catalog_name` argument is ignored.
**Description**

This function returns a listing of tables accessible to the connected user. The expectation is that this is a relatively thin wrapper around the ODBC SQLTables function call, albeit returning a subset of the fields.

**Usage**

```r
odbcConnectionTables(conn, name, ...)
## S4 method for signature 'OdbcConnection,Id'
odbcConnectionTables(conn, name, table_type = NULL)

## S4 method for signature 'OdbcConnection,character'
odbcConnectionTables(
  conn,
  name,
  catalog_name = NULL,
  schema_name = NULL,
  table_type = NULL
)

## S4 method for signature 'OdbcConnection,ANY'
odbcConnectionTables(
  conn,
  name = NULL,
  catalog_name = NULL,
  schema_name = NULL,
  table_type = NULL
)

## S4 method for signature 'Oracle,character'
odbcConnectionTables(
  conn,
  name,
  catalog_name = NULL,
  schema_name = NULL,
  table_type = NULL
)
```

**Arguments**

- `conn`  
  OdbcConnection
- `name`  
  table we wish to search for
- `...`  
  additional parameters to methods
table_type  List tables of this type, for example ’VIEW’. See odbcConnectionTableTypes for a listing of available table types for your connection.
catalog_name  character catalog where we wish to query for available tables
schema_name  character schema where we wish to query for available tables.

Details

It is important to note that, similar to the ODBC/API call, this method also accomodates pattern-value arguments for the catalog, schema, and table name arguments.

If extending this method, be aware that package:odbc internally uses this method to satisfy both DBI::dbListTables and DBI::dbExistsTable methods. ( The former also advertises pattern value arguments )

Query, rather than use SQLTables ODBC API for performance reasons on Oracle. Main functional difference between the implementation of SQLTables ( OEM driver ) and the query below is that the OEM implementation also looks through the synonyms. Given the performance reports, we sacrifice the synonym look-through for better execution time.

Value

data.frame with columns
  • table_catalog
  • table_schema
  • table_name
  • table_remarks

See Also

The ODBC documentation on SQLTables for further details.
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**

```r
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)
  )
}
```

---

### OdbcDriver

#### Odbc Driver Methods

**Description**

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

**Usage**

```r
## 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, ...)
```
odbcListColumns

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.

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 the available data sources on your system. See the DSN Configuration files section of the package README for details on how to install data sources for the most common databases.

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 the available drivers on your system. See the Installation section of the package README for details on how to install drivers for the most common databases.

Usage

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

Arguments

- **keep**  
  A character vector of driver names to keep in the results, if NULL (the default) will keep all drivers.

- **filter**  
  A character vector of driver names to filter from the results, if NULL (the default) will not filter any drivers.

Value

A data frame with three columns. If a given driver does not have any attributes the last two columns will be `NA`. Drivers can be excluded from being returned by setting the `odbc.drivers.filter` option.

- **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

```r
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:

```r
list(schema = 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.

odbcPreviewQuery  Create a preview query.

Description
Optimize against the rowLimit argument. S3 since some back-ends do not parse the LIMIT syntax.
Internal, not expected that users would interact with this method.

Usage
odbcPreviewQuery(connection, rowLimit, name)

## S3 method for class 'OdbcConnection'
odbcPreviewQuery(connection, rowLimit, name)

## S3 method for class 'Microsoft SQL Server'
odbcPreviewQuery(connection, rowLimit, name)

## S3 method for class 'Oracle'
odbcPreviewQuery(connection, rowLimit, name)
## OdbcResult

### Arguments

- **connection**: A connection object, as returned by `dbConnect()`.
- **rowLimit**: The maximum number of rows to display.
- **name**: Name of the object to be previewed.

### OdbcResult Methods

**Description**

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

**Usage**

```r
## 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'
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, ..., batch_rows = getOption("odbc.batch_rows", NA))
```

### 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.

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.

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
## 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)

sqlCreateTable-methods

~~ Methods for Function sqlCreateTable in Package DBI ~~

Description
~~ Methods for function sqlCreateTable in package DBI ~~
### Supported Connection Attributes

**Description**

These (pre) connection attributes are supported and can be passed as part of the `dbConnect` call in the named list `attributes` parameter:

**Details**

- `azure_token`: This should be a string scalar; in particular Azure Active Directory authentication token. Only for use with Microsoft SQL Server and with limited support away from the OEM Microsoft driver.

**Examples**

```r
## Not run:
conn <- dbConnect(
  odbc::odbc(),
  dsn = "my_azure_mssql_db",
  Encrypt = "yes",
  attributes = list("azure_token" = .token)
)

## End(Not run)
```
## Index

**datasets**
- SUPPORTED_CONNECTION_ATTRIBUTES, 27

**methods**
- dbListTables-methods, 7
- dbQuoteString-methods, 8
- sqlCreateTable-methods, 26

- bit64::integer64, 4

- ConnectionAttributes, 4

- ConnectionAttributes
  - (SUPPORTED_CONNECTION_ATTRIBUTES), 27

- dbAppendTable, 6
- dbAppendTable,OdbcConnection-method
  - (odbc-tables), 8
- dbBegin,OdbcConnection-method
  - (OdbcConnection), 11
- dbBind(), 12
- dbBind,OdbcResult-method
  - (OdbcResult), 25
- dbClearResult,OdbcResult-method
  - (OdbcResult), 25
- dbColumnInfo(), 6
- dbColumnInfo,OdbcResult-method
  - (OdbcResult), 25
- dbCommit,OdbcConnection-method
  - (OdbcConnection), 11
- dbConnect
  - (dbConnect,OdbcDriver-method), 3
  - (dbConnect(), 5, 7, 12, 26)
- dbConnect,OdbcDriver-method, 3
- dbCreateTable, 6
- dbDataType, 6
- dbDataType(), 10
- dbDataType,OdbcConnection,ANY-method
  - (OdbcConnection), 11
- dbDataType,OdbcConnection,data.frame-method
  - (OdbcConnection), 11
- dbDataType,OdbcDriver,ANY-method
  - (OdbcDriver), 20
- dbDataType,OdbcDriver,data.frame-method
  - (OdbcDriver), 20
- dbDataType,OdbcDriver,list-method
  - (OdbcDriver), 20
- dbDisconnect, 6
- dbDisconnect,OdbcConnection-method
  - (OdbcConnection), 11
- dbExecute, 6
- dbExistsTable, 6
- dbExistsTable,OdbcConnection,character-method
  - (OdbcConnection), 11
- dbExistsTable,OdbcConnection,Id-method
  - (OdbcConnection), 11
- dbExistsTable,OdbcConnection,SQL-method
  - (OdbcConnection), 11
- dbFetch,OdbcResult-method
  - (OdbcResult), 25
- dbGetException, 6
- dbGetInfo, 6
- dbGetInfo,OdbcConnection-method
  - (OdbcConnection), 11
- dbGetInfo,OdbcDriver-method
  - (OdbcDriver), 20
- dbGetQuery, 6
- dbGetQuery,OdbcConnection,character-method
  - (OdbcConnection), 11
- dbGetRowCount,OdbcResult-method
  - (OdbcResult), 25
- dbGetRowsAffected,OdbcResult-method
  - (OdbcResult), 25
- dbGetStatement,OdbcResult-method
  - (OdbcResult), 25
- dbHasCompleted,OdbcResult-method
  - (OdbcResult), 25
- DBI::dbConnect(), 10
- DBIConnection, 4, 5, 7, 12, 21, 25, 26
- DBIDriver, 4, 12, 21, 25
- DBIObject, 12, 25
- DBIResult, 12, 25
- dbIsReadOnly, 6
- dbIsValid, 6
- dbIsValid,OdbcConnection-method
  - (OdbcConnection), 11

---

28
INDEX

29

dbIsValid,OdbcDriver-method (OdbcDriver), 20
dbIsValid,OdbcResult-method (OdbcResult), 25
dbListFields (dbListFields,OdbcConnection,character-method), 5
dbListFields,OdbcConnection,character-method (dbListFields), 5
dbListObjects, 6
dbListObjects(), 6
dbListResults, 6
dbListTables, 6

dbListTables (dbListTables,OdbcConnection-method), 6
dbListTables,OdbcConnection-method, 6
dbListTables,Teradata-method, 6
dbListTables,6

dbListTables-methods, 7
dbQuoteIdentifier(), 5, 6, 10, 12

dbQuoteIdentifier,OdbcConnection,character-method (dbQuoteIdentifier), 11

dbQuoteIdentifier,OdbcConnection,SQL-method (dbQuoteIdentifier), 11
dbQuoteString,DBIConnection,ANY-method (dbQuoteString), 8
dbQuoteString,DBIConnection,character-method (dbQuoteString), 8
dbQuoteString,DBIConnection,SQL-method (dbQuoteString), 8
dbQuoteString,Hive,character-method (dbQuoteString), 8
dbQuoteString-methods, 8

dbReadTable, 6
dbRemoveTable, 6
dbRemoveTable,OdbcConnection,character-method (dbRemoveTable), 6
dbRollback,OdbcConnection-method (dbRollback), 11
dbSendQuery, 6

dbSendQuery,OdbcConnection,character-method (dbSendQuery), 11

dbSendStatement, 6
dbSendStatement,OdbcConnection,character-method (dbSendStatement), 11

dbWriteTable, 6

dbWriteTable(), 7, 15
dbWriteTable,OdbcConnection,character,character-method (dbWriteTable), 8
dbWriteTable,OdbcConnection,Id,character-method (dbWriteTable), 8
dbWriteTable,OdbcConnection,Id,character-data.frame-method (dbWriteTable), 8
dbWriteTable,OdbcConnection,SQL,character-data.frame-method (dbWriteTable), 8
dbWriteTable,OdbcConnection,SQL,character-data.frame-method (dbWriteTable), 8
dbWriteTable,OdbcConnection,SQL,character-data.frame-method (dbWriteTable), 8

dbWriteTable,OdbcConnection,SQL,character-data.frame-method (dbWriteTable), 8
dbWriteTable,OdbcConnection,SQL,character-data.frame-method (dbWriteTable), 8

29
odbcListObjectTypes, 23
odbcPreviewObject, 24
odbcPreviewQuery, 24
OdbcResult, 25
OdbcResult-class (OdbcResult), 25
odbcSetTransactionIsolationLevel, 26
OlsonNames(), 4

show,OdbcConnection-method (OdbcConnection), 11
show,OdbcDriver-method (OdbcDriver), 20
SQL(), 5, 10, 12
sqlCreateTable,DB2/AIX64-method (sqlCreateTable-methods), 26
sqlCreateTable,DBIConnection-method (sqlCreateTable-methods), 26
sqlCreateTable,HDB-method (sqlCreateTable-methods), 26
sqlCreateTable,OdbcConnection-method (odbc-tables), 8
sqlCreateTable,Oracle-method (sqlCreateTable-methods), 26
sqlCreateTable,Teradata-method (sqlCreateTable-methods), 26
sqlCreateTable-methods, 26
sqlData,OdbcConnection-method (odbc-tables), 8
SUPPORTED_CONNECTION_ATTRIBUTES, 27
Sys.timezone(), 4