Package ‘odbc’

June 8, 2018

Title Connect to ODBC Compatible Databases (using the DBI Interface)
Version 1.1.6
Description A DBI-compatible interface to ODBC databases.
License MIT + file LICENSE
URL https://github.com/r-dbi/odbc
BugReports https://github.com/r-dbi/odbc/issues
SystemRequirements C++11, GNU make, An ODBC3 driver manager and drivers.
Roxygen list(markdown = TRUE)
RoxygenNote 6.0.1
LazyData true
ByteCompile true
Depends R (>= 3.2.0)
Imports DBI (>= 1.0.0),
       methods,
       Rcpp (>= 0.12.11),
       blob (>= 1.1.0),
       bit64,
       hms
Suggests tibble,
       DBItest,
       testthat,
       covr,
       magrittr
LinkingTo Rcpp, BH
Collate ‘odbc.R’
       ‘Driver.R’
       ‘Connection.R’
       ‘DataTypes.R’
       ‘RcppExports.R’
       ‘Result.R’
       ‘Table.R’
       ‘Viewer.R’
       ‘db.R’
       ‘hidden.R’
       ‘utils.R’
       ‘zzz.R’
R topics documented:

odbc-package .......................................................... 2
dbConnect,OdbcDriver-method .................................... 3
dbListFields,OdbcConnection,character-method ................. 4
dbListTables,OdbcConnection-method ............................ 5
odbc ................................................................. 6
odbc-tables .......................................................... 6
OdbcConnection ......................................................... 7
odbcConnectionActions ................................................ 9
odbcConnectionIcon ................................................... 10
odbcDataType ........................................................ 10
OdbcDriver ............................................................ 11
odbcListColumns ....................................................... 12
odbcListDataSources .................................................. 13
odbcListDrivers ......................................................... 13
odbcListObjects ......................................................... 14
odbcListObjectTypes .................................................. 14
odbcPreviewObject ..................................................... 15
OdbcResult ............................................................. 16
odbcSetTransactionIsolationLevel .................................. 17
test_roundtrip .......................................................... 17

Index 19

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

Description

A DBI-compatible interface to ODBC databases.

Author(s)

Maintainer: Jim Hester <james.hesterrstudio.com>

Authors:

• Hadley Wickham <hadlerstudio.com>

Other contributors:

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

See Also

Useful links:

• https://github.com/r-dbi/odbc
• Report bugs at https://github.com/r-dbi/odbc/issues
Connect to a ODBC compatible database

Usage

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

Arguments

- `drv`: an object that inherits from `DBIConnection`, or an existing `DBIConnection` object (in order to clone an existing connection).
- `dsn`: The Data Source Name.
- `...`: Additional ODBC keywords, these will be joined with the other arguments to form the final connection string.
- `timezone`: The Server time zone. Useful if the database has an internal timezone that is not 'UTC'. If the database is in your local timezone set to Sys.timezone(). See `olsonNames()` for a complete list of available timezones on your system.
- `encoding`: The text encoding used on the Database. If the database is 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.
- `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.

**dbListFields, OdbcConnection, character-method**

*List field names of a remote table*

**Description**

List field names of a remote table

**Usage**

```r
## 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**  
  a character string with the name of the remote table.
- **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**

a character vector

**See Also**

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

Other DBIConnection generics: `DBIConnection-class, dbDataType, dbDisconnect, dbExecute, dbExistsTable, dbGetException, dbGetInfo, dbGetQuery, dbIsValid, dbListObjects, dbListResults, dbListTables, dbReadTable, dbRemoveTable, dbSendQuery, dbSendStatement, dbWriteTable`
**dbListTables, OdbcConnection-method**

*List remote tables*

**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, including temporary tables if supported by the database. As soon a table is removed from the database, it is also removed from the list of database tables.

The returned names are suitable for quoting with `dbQuoteIdentifier()`. An error is raised when calling this method for a closed or invalid connection.

**Additional arguments**

TBD: `temporary = NA`

This must be provided as named argument. See the "Specification" section for details on their usage.

**See Also**

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

Odbc driver

Description

Driver for an ODBC database.

Usage

odbc()

Examples

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

## End(Not run)

odbc-tables

Convenience functions for reading/writing DBMS tables

Description

Convenience functions for reading/writing DBMS tables

Usage

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

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

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

Arguments

c 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.
**OdbcConnection**

<table>
<thead>
<tr>
<th>temporary</th>
<th>If TRUE, will generate a temporary table statement.</th>
</tr>
</thead>
<tbody>
<tr>
<td>row.names</td>
<td>Either TRUE, FALSE, NA or a string. If TRUE, always translate row names to a column called &quot;row_names&quot;. 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.</td>
</tr>
<tr>
<td>field.types</td>
<td>Additional field types used to override derived types.</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments used by individual methods.</td>
</tr>
<tr>
<td>con</td>
<td>A database connection.</td>
</tr>
<tr>
<td>table</td>
<td>Name of the table. Escaped with <code>dbQuoteIdentifier()</code>.</td>
</tr>
<tr>
<td>fields</td>
<td>Either a character vector or a data frame. A named character vector: Names are column names, values are types. Names are escaped with <code>dbQuoteIdentifier()</code>. Field types are unescaped. A data frame: field types are generated using <code>dbDataType()</code>.</td>
</tr>
</tbody>
</table>

**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**

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

**Usage**

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

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

### `dbDisconnect`

```r
dbDisconnect(conn, ...)
```

### `dbSendQuery`

```r
dbSendQuery(conn, statement, ...)
```

### `dbSendStatement`

```r
dbSendStatement(conn, statement, ...)
```

### `dbDataType`

```r
dbDataType(dbObj, obj, ...)
```

### `dbDataFrame`

```r
dbDataFrame(dbObj, obj, ...)
```

### `dbQuoteIdentifier`

```r
dbQuoteIdentifier(conn, x, ...)
```

### `dbQuoteIdentifierSQL`

```r
dbQuoteIdentifier(conn, x, ...)
```

### `dbExistsTable`

```r
dbExistsTable(conn, name, ...)
```

### `dbRemoveTable`

```r
dbRemoveTable(conn, name, ...)
```

### `dbGetInfo`

```r
dbGetInfo(dbObj, ...)
```

### `dbGetQuery`

```r
dbGetQuery(conn, statement, n = -1, ...)
```

### `dbBegin`

```r
dbBegin(conn, ...)
```

### `dbCommit`

```r
dbCommit(conn, ...)
```

### `dbRollback`

```r
dbRollback(conn, ...)
```

### `dbExistsTableId`

```r
dbExistsTable(dbObj, name, ...)
```

### `dbExistsTableSQL`

```r
dbExistsTable(dbObj, 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.

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

### x
A character vector, SQL or Id object to quote as identifier.

### name
A character string specifying a DBMS table name.

### n
maximum number of records to retrieve per fetch. Use \( n = -1 \) or \( n = \text{Inf} \) to retrieve all pending records. Some implementations may recognize other special values.

---

### odbcConnectionActions
List the actions supported for the connection

---

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

### Usage
`odbcConnectionActions(connection)`

### Arguments
- `connection` A connection object, as returned by `dbConnect()`.

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

- **callback** A function to be invoked to perform the action
- **icon** An optional path to an icon representing the action

### Value
A named list of actions that can be performed on the connection.
odbcConnectionIcon  
*Get an icon representing a connection.*

**Description**

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

**Usage**

`odbcConnectionIcon(connection)`

**Arguments**

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

**Details**

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

**Value**

The path to an icon file on disk.

---

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

**Description**

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

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

**Usage**

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

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

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

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

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

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

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

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

Arguments

<table>
<thead>
<tr>
<th>object</th>
<th>Any R object</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbObj</td>
<td>A object inheriting from DBIDriver or DBICconnection</td>
</tr>
<tr>
<td>obj</td>
<td>An R object whose SQL type we want to determine.</td>
</tr>
<tr>
<td>...</td>
<td>Other arguments passed on to methods.</td>
</tr>
</tbody>
</table>

**odbcListColumns**

List columns in an object.

Description

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

Usage

odbcListColumns(connection, ...)

Arguments

<table>
<thead>
<tr>
<th>connection</th>
<th>A connection object, as returned by dbConnect().</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>Parameters specifying the object.</td>
</tr>
</tbody>
</table>

Details

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

Value

A data frame with name and type columns, listing the object’s fields.
**odbcListDataSources**

*List Available Data Source Names*

**Description**

List Available Data Source Names

**Usage**

```r
odbcListDataSources()
```

**Value**

A data frame with two columns.

- **name** Name of the data source
- **description** Data Source description

---

**odbcListDrivers**

*List Available ODBC Drivers*

**Description**

List Available ODBC Drivers

**Usage**

```r
odbcListDrivers()
```

**Value**

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

- **name** Name of the driver
- **attribute** Driver attribute name
- **value** Driver attribute value
odbcListObjects  
*List objects in a connection.*

**Description**

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

**Usage**

```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**

```r
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

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

Arguments

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

Details

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

Value

A data frame containing the data in the object.
OdbcResult

Odbc Result Methods

Description

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

Usage

```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, ...)
```

Arguments

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

*Set the Transaction Isolation Level for a Connection*

**Description**

Set the Transaction Isolation Level for a Connection

**Usage**

```r
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(conn, "read_uncommitted")

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

## End(Not run)
```

---

**test_roundtrip**

*Test round tripping a simple table*

**Description**

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

**Usage**

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

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

**Details**

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

**Examples**

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

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

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

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

bit64::integer64, 3

dbBegin, OdbcConnection-method (OdbcConnection), 7
dbBind, OdbcResult-method (OdbcResult), 16
dbClearResult, OdbcResult-method (OdbcResult), 16
dbColumnInfo(), 4
dbColumnInfo, OdbcResult-method (OdbcResult), 16
dbCommit, OdbcConnection-method (OdbcConnection), 7
dbConnect (dbConnect, OdbcDriver-method), 3
dbConnect(), 4, 5, 9, 17
dbConnect(), OdbcDriver-method, 3
dbDataType, 4
dbDataType(), 7
dbDataType, OdbcConnection, ANY-method (OdbcConnection), 7
dbDataType, OdbcConnection, data.frame-method (OdbcConnection), 7
dbDataType, OdbcDriver, ANY-method (OdbcDriver), 11
dbDataType, OdbcDriver, data.frame-method (OdbcDriver), 11
dbDataType, OdbcDriver, list-method (OdbcDriver), 11
dbDisconnect, 4
dbDisconnect, OdbcConnection-method (OdbcConnection), 7
dbExecute, 4
dbExistsTable, 4
dbExistsTable, OdbcConnection, character-method (OdbcConnection), 7
dbExistsTable, OdbcConnection, Id-method (OdbcConnection), 7
dbExistsTable, OdbcConnection, SQL-method (OdbcConnection), 7
dbFetch, OdbcResult-method (OdbcResult), 16

dbGetException, 4
dbGetInfo, 4
dbGetInfo, OdbcConnection-method (OdbcConnection), 7
dbGetInfo, OdbcDriver-method (OdbcDriver), 11
dbgetQuery, 4
dbgetQuery, OdbcConnection, character-method (OdbcConnection), 7
dbGetRowCount, OdbcResult-method (OdbcResult), 16
dbGetRowsAffected, OdbcResult-method (OdbcResult), 16
dbGetStatement, OdbcResult-method (OdbcResult), 16
dbHasCompleted, OdbcResult-method (OdbcResult), 16
DBI::dbConnect(), 6
DBIConnection, 3–5, 9, 12, 16, 17
DBIDriver, 3, 9, 12, 16
DBIObject, 9, 16
DBIResult, 9, 16
dbIsValid, 4
dbIsValid, OdbcConnection-method (OdbcConnection), 7
dbIsValid, OdbcDriver-method (OdbcDriver), 11
dbIsValid, OdbcResult-method (OdbcResult), 16
dbListFields (dbListFields, OdbcConnection, character-method), 4
dbListFields, OdbcConnection, character-method, 4
dbListObjects, 4
dbListResults, 4
dbListTables, 4

dbListTables (dbListTables, OdbcConnection-method), 5
dbListTables, OdbcConnection-method, 5

dbQuoteIdentifier(), 7
dbQuoteIdentifier, OdbcConnection, character-method (OdbcConnection), 7
dbQuoteIdentifier, 0dbcConnection, SQL-method
(0dbcConnection), 7

dbReadTable, 4
dbRemoveTable, 4
dbRemoveTable, 0dbcConnection, character-method
(0dbcConnection), 7
dbRollback, 0dbcConnection-method
(0dbcConnection), 7
dbSendQuery, 4
dbSendQuery, 0dbcConnection, character-method
(0dbcConnection), 7
dbSendStatement, 4
dbSendStatement, 0dbcConnection, character-method
(0dbcConnection), 7
dbWriteTable, 4
dbWriteTable, 5
dbWriteTable, 0dbcConnection, character, data.frame-method
(odbc-tables), 6

iconvlist(), 3
Id, 9

odbc, 6
odbc-package, 2
odbc-tables, 6
OdbcConnection, 6, 7
OdbcConnection-class (OdbcConnection), 7
odbcConnectionActions, 9
odbcConnectionIcon, 10
odbcDataType, 10
OdbcDriver, 11
OdbcDriver-class (OdbcDriver), 11
odbcListColumns, 12
odbcListDataSources, 13
odbcListDrivers, 13
odbcListObjects, 14
odbcListObjectTypes, 14
odbcPreviewObject, 15
OdbcResult, 16
OdbcResult-class (OdbcResult), 16
odbcSetTransactionIsolationLevel, 17
OlsonNames(), 3

show, OdbcConnection-method
(0dbcConnection), 7
show, OdbcDriver-method (OdbcDriver), 11
SQL, 9
sqlCreateTable, 0dbcConnection-method
(odbc-tables), 6
sqlData, 0dbcConnection-method
(odbc-tables), 6

test_roundtrip, 17