Package ‘ROracle’

March 4, 2014

Version 1.1-11

Date 2013-09-17

Author Denis Mukhin, David A. James and Jake Luciani

Maintainer Denis Mukhin <denis.x.mukhin@oracle.com>

Title OCI based Oracle database interface for R

Description Oracle database interface (DBI) driver for R.

This is a DBI-compliant Oracle driver based on the OCI.

SystemRequirements Oracle Instant Client or Oracle Database Client

LazyLoad yes

Depends methods, DBI (&gt;= 0.2-5)

Imports utils

License LGPL

URL http://www.oracle.com


NeedsCompilation yes

Repository CRAN

Date/Publication 2014-03-04 10:42:40

R topics documented:

  dbCallProc-methods .................................................... 2
  dbCommit-methods ..................................................... 2
  dbConnect-methods ................................................... 3
  dbDriver-methods ..................................................... 9
  dbGetInfo-methods .................................................... 10
  dbListConnections-methods ........................................... 12
dbCallProc-methods  

Call an SQL stored procedure

Description

Not yet implemented.

Methods

conn  a OraConnection object.

... additional arguments are passed to the implementing method.

See Also

Oracle, dbConnect, dbSendQuery, dbGetQuery, fetch, dbCommit, dbGetInfo, dbReadTable.

dbCommit-methods  

DBMS Transaction Management

Description

Commits or rolls back the current transaction in an Oracle connection

Usage

```r
## S4 method for signature 'OraConnection'
dbCommit(conn, ...)
## S4 method for signature 'OraConnection'
dbRollback(conn, ...)
```

Arguments

- conn  a OraConnection object, as produced by the function dbConnect
- ... currently unused.
**Details**

- **dbCommit**: implementation saves all changes done on that connection. Changes can not be undone once saved permanently.
- **dbRollback**: implementation undo all changes done after last savepoint.

**Side Effects**

- **dbCommit**: saves changes permanently.
- **dbRollback**: undo all changes done after last save point.

**References**


**See Also**

Oracle, dbConnect, dbSendQuery, dbGetQuery, fetch, dbCommit, dbGetInfo, dbReadTable.

**Examples**

```r
## Not run:
drv <- dbDriver("Oracle")
con <- dbConnect(drv, "scott", "tiger")
dbReadTable(con, "EMP")
rs <- dbSendQuery(con, "delete from emp where deptno = 10")
dbReadTable(con, "EMP")
if(dbGetInfo(rs, what = "rowsAffected") > 1)
{
  warning("dubious deletion -- rolling back transaction")
dbRollback(con)
}
dbReadTable(con, "EMP")

## End(Not run)
```

---

**dbConnect-methods**

*Create a Connection Object to an Oracle DBMS*

**Description**

These methods are straight-forward implementations of the corresponding generic functions.
Usage

```r
## S4 method for signature 'OraDriver'
dbConnect(drv, username = "", password = "", dbname = "", prefetch = FALSE,
          bulk_read = 1000L, stmt_cache = 0L, external_credentials = FALSE,
          sysdba = FALSE, ...)

## S4 method for signature 'ExtDriver'
dbConnect(drv, prefetch = FALSE, bulk_read = 1000L, stmt_cache = 0L,
          external_credentials = FALSE, sysdba = FALSE, ...)

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

Arguments

- **drv**: An object of class `OraDriver` or `ExtDriver`.
- **conn**: An `OraConnection` object as produced by `dbConnect`.
- **username**: A character string specifying a user name.
- **password**: A character string specifying a password.
- **dbname**: A character string specifying a connect identifier (for more information refer to Chapter 8, Configuring Naming Methods, of Oracle Database Net Services Administrator’s Guide). This is the same as part of the SQL*Plus connect string that follows the '@' sign. If you are using Oracle Wallet to store username and password, then this string should be the connect string used to create the wallet mapping (for more information, refer to Configuring Clients to Use the External Password Store in Chapter 3 of Oracle Database Security Guide). Connect identifiers for an Oracle TimesTen IMDB instance are supported via the OCI tnsnames or easy connect naming methods. For additional information on TimesTen connections for OCI see chapter 3, TimesTen Support for OCI, of TimesTen In-Memory C Developer’s Guide.
- **prefetch**: A logical value indicating TRUE or FALSE. When set to TRUE, ROracle will use OCI prefetch buffers to retrieve additional data from the server thus saving memory required in RODBI/ROOCI by allocating a single row buffer to fetch the data from OCI. Using prefetch results in a fetch call for every row. By default, prefetch is FALSE and array fetch is used to retrieve the data from the server.
- **bulk_read**: An integer value indicating the number of rows to fetch at a time. The default value is 1000L. When the prefetch option is selected, memory is allocated for prefetch buffers and OCI will fetch that many rows at a time. When prefetch is not used (the default), memory is allocated in RODBI/ROOCI define buffers. Setting this to a large value will result in more memory allocated based on the number of columns in the select list and the types of columns. For a column of type character, define buffers are allocated using the maximum width times the NLS maximum width. Applications should adjust this value based on the query result and a larger value will benefit queries that return a large result. An application can tune this value as needed.
- **stmt_cache**: An integer value indicating the number of statements to cache. It means that cursors are ready to be used without the need to parse the statements again. The
default value is 0L. If stmt_cache value is greater than 0L then prefetch value must be set to TRUE.

external_credentials
A logical value indicating TRUE or FALSE. When set to TRUE, ROracle will begin OCI session authenticated with external credentials on the connection. The default value is FALSE.

sysdba
A logical value indicating TRUE or FALSE. When set to TRUE, ROracle will begin OCI session with SYSDBA privileges on the connection. The default value is FALSE.

... Currently unused.

Details

dbConnect This connection object is used to execute operations on the database.

When prefetch is set to TRUE, it allows the use of the OCI prefetch buffer to retrieve additional data from the server.

The bulk_read argument is used to set an integer value indicating the number of rows to fetch at a time.

The stmt_cache argument is used to enable or disable the statement caching feature. Its value specifies the statement cache size.

The external_credentials argument is used to begin OCI session authenticated with external credentials on the connection.

The sysdba argument is used to begin OCI session with SYSDBA privileges on the connection.

When establishing a connection with an ExtDriver driver, none of the arguments specifying credentials are used. A connection in this mode is a singleton object, that is, all calls to dbConnect return the same connection object.

dbDisconnect This implementation disconnects the connection between R and the database server.

It frees all resources used by the connection object. It frees all result sets associated with this connection object.

Value

dbConnect An object OraConnection whose class extends DBIConnection. This object is used to execute SQL queries on the database.

dbDisconnect A logical value indicating whether the operation succeeded or not.

Side Effects

dbConnect Establishes a connection between R and an Oracle Database server.

dbDisconnect Frees resources used by the connection object.

References

For the Oracle Database documentation see http://www.oracle.com/technetwork/indexes/documentation/index.html.
See Also

Oracle, dbConnect, dbSendQuery, dbGetQuery, fetch, dbCommit, dbGetInfo, dbReadTable.

Examples

## Not run:
## create an Oracle Database instance and create one connection on the
## same machine.
drv <- dbDriver("Oracle")

## use username/password authentication
con <- dbConnect(drv, username = "scott", password = "tiger")

## run a SQL statement by creating first a resultSet object
rs <- dbSendQuery(con, "select * from emp where deptno = 10")

## we now fetch records from the resultSet into a data.frame
data <- fetch(rs)    ## extract all rows
dim(data)

## End(Not run)
## Not run:
## create an Oracle Database instance and create one connection to a
## remote database using the SID in the connect string.
drv <- dbDriver("Oracle")

## refer to Oracle Database Net Services Administrator's Guide for
## details on connect string specification.
host <- "myhost"
port <- 1521
sid <- "mysid"
connect_string <- paste(
  "(DESCRIPTION=",
  "(ADDRESS=(PROTOCOL=tcp)(HOST=", host, ")(PORT=", port, "))",
  "(CONNECT_DATA=(SID=", sid, "))", sep = "")"

## use username/password authentication
con <- dbConnect(drv, username = "scott", password = "tiger",
                 dbname = connect_string)

## run a SQL statement by creating first a resultSet object
rs <- dbSendQuery(con, "select * from emp where deptno = 10")

## we now fetch records from the resultSet into a data.frame
data <- fetch(rs)    ## extract all rows
dim(data)

## End(Not run)
## Not run:
## create an Oracle Database instance and create one connection to a
## remote database using the service name.
drv <- dbDriver("Oracle")
## dbConnect-methods

### refer to Oracle Database Net Services Administrator's Guide for details on connect string specification.

```r
host <- "myhost"
port <- 1521
svc <- "mydb.example.com"
connect.string <- paste(  
  "(DESCRIPTION=",  
  "(ADDRESS=(PROTOCOL=tcp)(HOST="host", PORT="port", "))",  
  "(CONNECT_DATA=(SERVICE_NAME="svc",))", sep="")"
```

### use username/password authentication
```
con <- dbConnect(drv, username = "scott", password = "tiger",  
  dbname = connect.string)
```

### run a SQL statement by creating first a resultSet object
```
rs <- dbSendQuery(con, "select * from emp where deptno = 10")
```

### we now fetch records from the resultSet into a data.frame
```
data <- fetch(rs)  
# extract all rows
```
```
dim(data)
```

### End(Not run)
### Not run:
```
# create an Oracle Database instance and create one connection.
drv <- dbDriver("Oracle")
```

### use Oracle Wallet authentication
```
con <- dbConnect(drv, username = "", password = "",  
  dbname = "<wallet_connect_string>"
```

### run a SQL statement by creating first a resultSet object
```
rs <- dbSendQuery(con, "select * from emp where deptno = 10")
```

### we now fetch records from the resultSet into a data.frame
```
data <- fetch(rs)  
# extract all rows
```
```
dim(data)
```

### End(Not run)
### Not run:
```
# create an Oracle Database instance and create one connection.
drv <- dbDriver("Oracle")
```

### connect to a TimesTen IMDB instance using the easy connect naming method where SampleDb is a direct driver TimesTen DSN
```
con <- dbConnect(drv, username = "scott", password = "tiger",  
  dbname = "localhost/SampleDb:timesTen_direct")
```

### run a SQL statement by creating first a resultSet object
```
rs <- dbSendQuery(con, "select * from dual")
```

### we now fetch records from the resultSet into a data.frame
```
data <- fetch(rs)  
# extract all rows
```
```
dim(data)
```
```r
# End(Not run)
# Not run:
# connect to an extproc (this assumes that the driver has already
# been initialized in the embedded R code by passing an external
# pointer representing the extproc context)
con <- dbConnect(Extproc())

# run a SQL statement by creating first a resultSet object
rs <- dbSendQuery(con, "select * from dual")

# we now fetch records from the resultSet into a data.frame
data <- fetch(rs)  # extract all rows
dim(data)

# End(Not run)
# Not run:
# create an Oracle Database instance and create one connection. 
drv <- dbDriver("Oracle")

# create connection with SYSDBA privileges
con <- dbConnect(drv, username = "scott", password = "tiger", 
                 sysdba = TRUE)

# run a SQL statement by creating first a resultSet object
rs <- dbSendQuery(con, "select * from emp where deptno = 10")

# we now fetch records from the resultSet into a data.frame
data <- fetch(rs)  # extract all rows
dim(data)

# End(Not run)
# Not run:
# create an Oracle Database instance and create one connection.
drv <- dbDriver("Oracle")

# Use OS authentication as an example of external authentication
# Make sure that database user exist to allow an OS authentication

# create connection authenticated with external credentials
con <- dbConnect(drv, username = "", password = "",
                 external_credentials = TRUE)

# Above dbConnect() used OS credentials to connect with database.

# run a SQL statement by creating first a resultSet object
rs <- dbSendQuery(con, "select * from emp where deptno = 10")

# we now fetch records from the resultSet into a data.frame
data <- fetch(rs)  # extract all rows
dim(data)
```

Description
Oracle driver initialization and closing

Usage

```r
## S4 method for signature 'OraDriver'
dbUnloadDriver(drv, ...)
## S4 method for signature 'ExtDriver'
dbUnloadDriver(drv, ...)
```

Arguments

```r
drv                       an object that inherits from OraDriver or ExtDriver as created by dbDriver.
...                      any other arguments are passed to the driver drvName.
```

Details

- **dbDriver** object is a singleton, that is, on subsequent invocations it returns the same initialized object.
- This implementation allows you to connect to multiple host servers and run multiple connections on each server simultaneously.
- When interruptible is set to TRUE, it allows for interrupting long-running queries on the server by executing the query in a thread. Main thread checks for Ctrl-C and issues OCI-Break/OCIReset to cancel the operation on the server. By default interruptible is FALSE.

**dbUnloadDriver** implementation remove communication link between R client and database. It frees all connection and all result sets associated with those connection objects

Value

- **dbDriver** An object OraDriver or ExtDriver whose class extends DBIDriver. This object is used to create connections, using the function `dbConnect`, to one or several Oracle database engines.
- **dbUnloadDriver** free all resources occupied by driver object.

Side Effects

- **dbDriver** The R client part of the database communication is initialized, but note that connecting to the database engine needs to be done through calls to `dbConnect`.
- **dbUnloadDriver** Remove communication link between R client and database.
References

For the Oracle Database documentation see http://www.oracle.com/technetwork/indexes/documentation/index.html.

See Also

Oracle, dbConnect, dbSendQuery, dbGetQuery, fetch, dbCommit, dbGetInfo, dbListTables, dbReadTable.

Examples

```r
## Not run:
# create an Oracle instance
drv <- dbDriver("Oracle")

con <- dbConnect(drv, "scott", "tiger")
res <- dbSendQuery(con, "select * from emp")
fetch(res, n = 5)
fetch(res)
dbClearResult(res)
dbUnloadDriver(drv)

## End(Not run)
```

---

**Description**

These methods are straightforward implementations of the corresponding generic functions.

**Usage**

```r
## S4 method for signature 'OraDriver'
dbGetInfo(dbObj, ...)
## S4 method for signature 'ExtDriver'
dbGetInfo(dbObj, ...)
## S4 method for signature 'OraConnection'
dbGetInfo(dbObj, what, ...)
## S4 method for signature 'OraResult'
dbGetInfo(dbObj, what, ...)
## S4 method for signature 'OraResult'
dbGetStatement(res, ...)
## S4 method for signature 'OraResult'
dbGetRowCount(res, ...)
## S4 method for signature 'OraResult'
dbGetRowsAffected(res, ...)
## S4 method for signature 'OraResult'
```
### dbGetInfo-methods

**dbColumnInfo**

```r
# S4 method for signature 'OraResult'
dbHasCompleted(res, ...)
```

### Arguments

- `dbObj` any object that implements some functionality in the R interface to databases (a driver, a connection or a result set).
- `what` a character string specifying an element of the output list.
- `res` an `OraResult`.
- `...` currently unused.

### Value

information about object.

### References


### See Also

`Oracle`, `dbDriver`, `dbConnect`, `dbSendQuery`, `dbGetQuery`, `fetch`, `dbCommit`, `dbGetInfo`, `dbListTables`, `dbReadTable`.

### Examples

```r
## Not run:
drv <- dbDriver("Oracle")
con <- dbConnect(drv, "scott", "tiger")

rs <- dbSendQuery(con, "select * from emp")
dbGetStatement(rs)
dbHasCompleted(rs)
dbGetInfo(rs)

# DBIDriver info
names(dbGetInfo(drv))

# DBIConnection info
names(dbGetInfo(con))

# DBIResult info
names(dbGetInfo(rs))

## End(Not run)
```
dbListConnections-methods

List items from Oracle objects

Description

These methods are straightforward implementations of the corresponding generic functions.

Usage

```r
## S4 method for signature 'OraDriver'
dbListConnections(drv, ...)  
## S4 method for signature 'ExtDriver'
dbListConnections(drv, ...)  
## S4 method for signature 'OraConnection'
dbListResults(conn, ...)
```

Arguments

- `drv` an OraDriver or ExtDriver.
- `conn` an OraConnection.
- `...` currently unused.

Details

- `dbListConnections` implementation return a list of all associated connections. It shows information about all associated connections.

- `dbListResults` implementation return a list of all associated result sets. It shows information about all associated result sets.

Value

- `dbListConnections` A list of all connections associated with driver.
- `dbListResults` A list of all result sets associated with connection.

References

For the Oracle Database documentation see http://www.oracle.com/technetwork/indexes/documentation/index.html.

See Also

Oracle, dbGetInfo, dbColumnInfo, dbDriver, dbConnect, dbSendQuery
Convenience Functions for Manipulating DBMS Tables

Description

These functions mimic their R counterparts get, assign, exists, remove, objects, and names except that they generate code that gets remotely executed in a database engine.

Usage

```r
## S4 method for signature 'OraConnection,character'
dbReadTable(conn, name, schema = NULL, row.names = NULL, ...)
## S4 method for signature 'OraConnection,character,data.frame'
dbWriteTable(conn, name, value, row.names = FALSE, overwrite = FALSE,
              append = FALSE, ora.number = TRUE, schema = NULL, ...)
## S4 method for signature 'OraConnection,character'
dbExistsTable(conn, name, schema = NULL, ...)
## S4 method for signature 'OraConnection,character'
dbRemoveTable(conn, name, purge = FALSE, schema = NULL, ...)
## S4 method for signature 'OraConnection'
dbListTables(conn, schema = NULL, all = FALSE, full = FALSE, ...)
## S4 method for signature 'OraConnection,character'
dbListFields(conn, name, schema = NULL, ...)
```

Arguments

- `conn` An `OraConnection` database connection object.
- `name` A case-sensitive character string specifying a table name.
- `schema` A case-sensitive character string specifying a schema name (or a vector of character strings for `dbListTables`).
row.names  In the case of dbReadTable, this argument can be a string, an index or a logical vector specifying the column in the DBMS table to be used as row.names in the output data.frame (a NULL specifies that no column should be used as row.names in the output). The default is NULL.

In the case of dbWriteTable, this argument should be a logical value specifying whether the row.names should be output to the output DBMS table; if TRUE, an extra column whose name is "row.names" will be added to the output. The default is FALSE.

value  A data.frame containing the data to write to a table. (See Details section for supported column types.)

overwrite  A logical value specifying whether to overwrite an existing table or not. The default is FALSE.

append  A logical value specifying whether to append to an existing table in the DBMS. The default is FALSE.

ora.number  A logical value specifying whether to create a table with Oracle NUMBER or BINARY_DOUBLE columns while writing numeric data. The default value is TRUE.

purge  A logical value specifying whether to add the PURGE option to the SQL DROP TABLE statement.

all  A logical value specifying whether to look at all schemas.

full  A logical value specifying whether to generate schema names. When argument all is TRUE, the output is a vector containing schema names followed by the table names. Using matrix(..., ncol = 2) on the output produces a matrix where each row corresponds to a table and the columns represent the schema names and table names respectively.

...  currently unused.

Details

Table, schema, and column names are case sensitive, e.g., table names ABC and abc are not the same. All database schema object names should not include double quotes as they are enclosed in double quotes when the corresponding SQL statement is generated.

ROracle methods such as dbReadTable, dbGetQuery, fetch, and dbWriteTable use the following mapping between R and Oracle data types:

- logical and integer map to Oracle INTEGER
- numeric maps to Oracle NUMBER if argument ora.number is TRUE or Oracle BINARY_DOUBLE if FALSE
- character maps to Oracle VARCHAR2(4000)
- Date and POSIXct map to Oracle DATE ROracle - the ROracle package R - the R application POSIXct - the POSIXct class TIMESTAMP TIMESTAMP WITH TIME ZONE TIMESTAMP WITH LOCAL TIME ZONE
- difftime maps to Oracle INTERVAL DAY TO SECOND
- list of raw vectors map to Oracle RAW(2000)
- other R types such as factor are converted to character
ROracle returns values from database columns that are of data type: date, time stamp, time stamp with time zone and time stamp with local time zone data types in R’s POSIXct format. POSIXct refers to a time that is internally stored as the number of seconds since the start of 1970 in UTC. Number of seconds are exchanged from R and ROracle driver in floating point double format. In POSIXct representation R uses the TZ environment variable or maps the OS time zone environment variable to its own, therefore the date will be displayed in this time zone format.

One can insert data into columns of the 4 data types listed above using a string with the correct format or POSIXct representation. String data is passed to the database directly and ROracle relies on database to convert it to date time representation. POSIXct data will be converted to Oracle’s internal DateTime descriptor by adding the beginning of 1970 UTC to the number of seconds obtained from R. Any POSIXct variables in insert statement are bound as SQLT_TIMESTAMP_TZ.

When these data types are selected from Oracle they are converted to POSIXct format by subtracting the start of 1970 UTC. When ROracle client and database have different time zone version files and an application uses the time zone data types it is recommended that they be the same as the region may have changed its zone. ROracle will report an error in this situation as it can cause data corruption or loss of information when operating on these data types.

Columns having date and time stamp data type are fetched by ROracle using the SQLT_TIMESTAMP data type. Columns having time stamp with time zone and time stamp with local time zone data types are fetched using SQLT_TIMESTAMP_TZ data type. Columns of datatype time zone with local time zone undergo conversion to the session time zone that the R application runs in therefore setting the time zone environment TZ in R will affect the data values in this column. ROracle driver does not map the TZ environment variable to session time zone and does not issue an alter DDL to set the session time zone. In order to fetch data from columns with timestamp with time zone and timestamp with local time zone the client and server must have the same time zone data file else an error will be reported.

Example below shows the effect of changing time zone in R environment:

```
R> Sys.timezone()
[1] "PST8PDT"
# Selecting data and displaying it
res <- dbGetQuery(con, selStr)
R> res[,1]
[1] 1 2 3 4 5 6
R> res[,2]
[1] "2012-06-05 00:00:00 PDT" "2012-01-05 07:15:02 PST"
   "2012-01-05 00:00:00 PST" "2011-01-05 00:00:00 PST"
[5] "2013-01-05 00:00:00 PST" "2020-01-05 00:00:00 PST"
R> res[,3]
[1] "2012-06-05 00:00:00 PDT" "2012-01-05 07:15:03 PST"
   "2012-01-05 00:00:00 PST" "2011-01-05 00:00:00 PST"
[5] "2013-01-05 00:00:00 PST" "2020-01-05 00:00:00 PST"
R> res[,4]
[1] "2012-06-05 00:00:00 PDT" "2012-01-05 07:15:03 PST"
   "2012-01-05 00:00:00 PST" "2011-01-05 00:00:00 PST"
[5] "2013-01-05 00:00:00 PST" "2020-01-05 00:00:00 PST"
R> res[,5]
[1] "2012-06-05 00:00:00 PDT" "2012-01-05 07:15:03 PST"
```
Also dbWriteTable always auto commits a current transaction as well as the data it inserts, i.e. it acts as a DDL statement even if appends rows to an already existing table.

**Value**

A data.frame in the case of dbReadTable; a vector in the case of dbListTables and dbListFields; a logical in the case of dbExistsTable indicating whether the table exists; otherwise TRUE when the operation was successful or an exception.

**References**

For the Oracle Database documentation see [http://www.oracle.com/technetwork/documentation/index.html](http://www.oracle.com/technetwork/documentation/index.html). For Datetime Data Types and Time Zone Support in Oracle see [http://docs.oracle.com/cd/E14072_01/server.112/e10729/ch4datetime.htm](http://docs.oracle.com/cd/E14072_01/server.112/e10729/ch4datetime.htm).

**See Also**

Oracle, dbDriver, dbConnect, dbSendQuery, dbGetQuery, fetch, dbCommit, dbGetInfo.

**Examples**

```r
## Not run:
con <- dbConnect(Oracle(), "scott", "tiger")
if (dbExistsTable(con, "FOO", "SCOTT"))
  dbRemoveTable(con, "FOO")

foo <- dbReadTable(con, "EMP")
row.names(foo) <- foo$EMPNO
```
foo <- foo[-1]

dbWriteTable(con, "FOO", foo, row.names = TRUE)
dbWriteTable(con, "FOO", foo, row.names = TRUE, overwrite = TRUE)
dbReadTable(con, "FOO", row.names = 1)

dbGetQuery(con, "delete from foo")
dbWriteTable(con, "FOO", foo, row.names = TRUE, append = TRUE)
dbReadTable(con, "FOO", row.names = 1)
dbRemoveTable(con, "FOO")

dbListTables(con)
dbListFields(con, "EMP")

if (dbExistsTable(con, "RORACLE_TEST", "SCOTT"))
  dbRemoveTable(con, "RORACLE_TEST")

# example of POSIXct usage
# A table is created using:
createTab <- "create table RORACLE_TEST(row_num number, id1 date,
id2 timestamp, id3 timestamp with time zone,
id4 timestamp with local time zone )"

dbGetQuery(con, createTab)
# Insert statement
insStr <- "insert into RORACLE_TEST values(\1, \2, \3, \4, \5)";

# Select statement
selStr <- "select * from RORACLE_TEST";

# Insert time stamp without time values in POSIXct form
x <- 1;
y <- "2012-06-05";
y <- as.POSIXct(y);
dbGetQuery(con, insStr, data.frame(x, y, y, y));

# Insert date & time stamp with time values in POSIXct form
x <- 2;
y <- "2012-01-05 07:15:02";
y <- as.POSIXct(y);
z <- "2012-01-05 07:15:03.123";
z <- as.POSIXct(z);
dbGetQuery(con, insStr, data.frame(x, y, z, z));

# Insert list of date objects in POSIXct form
x <- c(3, 4, 5, 6);
y <- c('2012-01-05', '2011-01-05', '2013-01-05', '2020-01-05');
y <- as.POSIXct(y);
dbGetQuery(con, insStr, data.frame(x, y, y, y));

dbCommit (con)

# Selecting data and displaying it
dbSendQuery-methods

Execute a statement on a given database connection

Description

These methods are straight-forward implementations of the corresponding generic functions except for the execute method which is an ROracle specific DBI extension.

Usage

```r
## S4 method for signature 'OraConnection,character'
dbSendQuery(conn, statement, data = NULL, prefetch = FALSE, bulk_read = 1000L, ...)
## S4 method for signature 'OraConnection,character'
dbGetQuery(conn, statement, data = NULL, prefetch = FALSE, bulk_read = 1000L, ...)
## S4 method for signature 'OraResult'
dbClearResult(res, ...)
## S4 method for signature 'OraConnection'
dbGetException(conn, ...)
execute(res, ...)
## S4 method for signature 'OraResult'
execute(res, data = NULL, ...)
```

Arguments

- `conn` an OraConnection object.
- `statement` a character vector of length 1 with the SQL statement.
- `res` an OraResult object.
- `data` a data.frame specifying bind data
- `prefetch` a logical value indicating TRUE or FALSE. When set to TRUE, ROracle will use OCI prefetch buffers to retrieve additional data from the server thus saving memory required in RODBI/ROOCI by allocating a single row buffer to fetch the data from OCI. Using prefetch results in a fetch call for every row. By default, prefetch is FALSE and array fetch is used to retrieve the data from server.
- `bulk_read` an integer value indicating the number of rows to fetch at a time. Default is 1000L. When the prefetch option is selected, memory is allocated for prefetch buffers and OCI will fetch those many rows at a time. When prefetch is not
used (default) memory is allocated in RODBI/ROOCI define buffers. Setting this to a large value will result in more memory allocated based on the number of columns in the select list and the types of columns. For a column of character, define buffers are allocated using the maximum width times the NLS maximum width. Application should adjust this value based on the query result and a larger value will benefit queries that return a large result. Application can tune this value to its need.

... currently unused.

Details

- **dbGetQuery** implementation does execute query statement and fetch result data from database.
- **dbSendQuery** implementation does execute query statement and return result set to the application. Now application can perform operations on result set.
- **dbClearResult** implementation does free resources occupied by result set.
- **dbGetException** implementation does retrieve error information.
- **execute** implementation does execute query statement.

Value

dbSendQuery an object OraResult whose class extends DBIResult. This object is used to fetch data from database, using the function fetch.

Side Effects

- **dbGetQuery** query statement is executed and data has fetched from database.
- **dbSendQuery** query statement is executed, but note that data needs to fetch through calls to **fetch**.
- **dbClearResult** resources acquired by result set has free.
- **dbGetException** error information retrieved and then cleaned from driver.
- **execute** query statement is executed.

References

For the Oracle Database documentation see http://www.oracle.com/technetwork/indexes/documentation/index.html.

See Also

Oracle, dbDriver, dbConnect, fetch, dbCommit, dbGetInfo, dbReadTable.

Examples

```r
## Not run:
drv <- dbDriver("Oracle")
con <- dbConnect(drv, "scott", "tiger")
res <- dbSendQuery(con, "select * from emp where deptno = :1",
                   data = data.frame(deptno = 10))
```
data <- fetch(res, n = -1)
res2 <- dbSendQuery(con, "select * from emp where deptno = :1",
   data1 = data.frame(deptno = 10), prefetch=TRUE,
   bulk_read=2L)
data1 <- fetch(res2, n = -1)
res3 <- dbSendQuery(con, "select * from emp where deptno = :1",
   data2 = data.frame(deptno = 10), bulk_read=10L)
data2 <- fetch(res3, n = -1)
res4 <- dbSendQuery(con, "select * from emp where ename = :1",
   data3 = data.frame(ename = 'SMITH'))
data3 <- fetch(res4, n = -1)

## End(Not run)

### dbSetDataMappings-methods

Set data mappings between Oracle and R

**Description**

Not yet implemented

**Methods**

- **res** a `OraResult` object as returned by `dbSendQuery`.
- **flds** a `data.frame` with field descriptions as returned by `dbColumnInfo`.
- ... any additional arguments are passed to the implementing method.

**See Also**

`Oracle, dbSendQuery, fetch, dbColumnInfo`.

**Examples**

```r
## Not run:
makeImage <- function(x) {
   .C("make_image", as.integer(x), length(x))
}

res <- dbSendQuery(con, statement)
flds <- dbColumnInfo(res)
flds[3, "Sclass"] <- makeImage

dbSetDataMappings(res, flds)
im <- fetch(res, n = -1)

## End(Not run)
```
Class ExtDriver

Description

An Oracle extproc driver class implementing the R database interface (DBI) API.

Generators

The main generators are dbDriver and Extproc.

Extends

Class "DBIDriver", directly. Class "DBIObject", by class "DBIDriver", distance 2.

Methods

dbConnect signature(drv = "ExtDriver"): ...

dbGetInfo signature(dbObj = "ExtDriver"): ...

dbListConnections signature(drv = "ExtDriver"): ...

dbUnloadDriver signature(drv = "ExtDriver"): ...

summary signature(object = "ExtDriver"): ...

show signature(object = "ExtDriver")

See Also

DBI classes: OraConnection-class OraResult-class

Examples

## Not run:
con <- dbConnect(Extproc())

## End(Not run)
**fetch-methods**

Fetch records from a previously executed query

---

### Description

This method is a straightforward implementation of the corresponding generic function.

### Usage

```r
## S4 method for signature 'OraResult'
fetch(res, n = -1, ...)  
```

### Arguments

- `res`: an `OraResult` object.
- `n`: maximum number of records to retrieve per fetch. Use `n = -1` to retrieve all pending records.
- `...`: currently unused.

### Details

The ROracle implementations retrieves only `n` records, and if `n` is missing it returns all records.

### Value

Number of records fetched from database.

### References


### See Also

`Oracle, dbConnect, dbSendQuery, dbGetQuery, dbClearResult, dbCommit, dbGetInfo, dbReadTable`

### Examples

```r
## Not run:
drv <- dbDriver("Oracle")
con <- dbConnect(drv, "scott", "tiger")
res <- dbSendQuery(con, "select * from emp")

# we now fetch the first 10 records from the resultset into a data.frame
data1 <- fetch(res, n = 10)
dim(data1)

dbHasCompleted(res)
```
# Instantiate an Oracle client from the current R session

**Description**

This function creates and initializes an Oracle client from the current R session. It returns an object that allows you to connect to one or several Oracle servers.

**Usage**

```r
Oracle(interruptible = FALSE)
Extproc(extproc.ctx = NULL)
```

**Arguments**

- `interruptible` logical indicating whether to allow user interrupts on long-running queries.
- `extproc.ctx` An external pointer wrapping extproc context.

**Details**

This object is a singleton, that is, on subsequent invocations it returns the same initialized object.

This implementation allows you to connect to multiple host servers and run multiple connections on each server simultaneously.

When `interruptible` is set to TRUE, it allows for interrupting long-running queries on the server by executing the query in a thread. Main thread checks for Ctrl-C and issues OCIBreak/OCIReset to cancel the operation on the server. By default `interruptible` is FALSE.

**Value**

An object of class OraDriver for Oracle or ExtDriver for Extproc whose class extends DBIDriver. This object is used to create connections, using the function `dbConnect`, to one or several Oracle database engines.

**Side Effects**

The R client part of the database communication is initialized, but note that connecting to the database engine needs to be done through calls to `dbConnect`.
Oracle user authentication

In order to establish a connection to an Oracle server users need to provide a user name, a password, and possibly a connect identifier (for more informations refer to chapter 8 (Configuring Naming Methods) of Oracle Database Net Services Administrator’s Guide). This is the same as part of the SQL*Plus connect string that follows the ‘@’ sign.

Connections to an Oracle TimesTen IMDB instance are established using the OCI tnsnames or easy connect naming methods. For additional information on TimesTen connections for OCI see chapter 3 (TimesTen Support for Oracle Call Interface) of the Oracle TimesTen In-Memory C Developer’s Guide.

Transactions

The current implementation directly supports transaction commits and rollbacks on a connection-wide basis through calls to $\texttt{dbCommit}$ and $\texttt{dbRollback}$. Save points are not yet directly implemented, but you may be able to define them and rollback to them through calls to dynamic SQL with $\texttt{dbGetQuery}$.

Notice that Oracle (and ANSI/ISO compliant DBMS) transactions are implicitly started when data definition SQL are executed (create table, etc.), which helper functions like $\texttt{dbWriteTable}$ may execute behind the scenes. You may want or need to commit or roll back your work before issuing any of these helper functions.

References

For the Oracle Database documentation see http://www.oracle.com/technetwork/documentation/index.html.

Author(s)

David A. James and Denis Mukhin

See Also

On database managers:
$\texttt{dbDriver}$ $\texttt{dbUnloadDriver}$ $\texttt{dbListConnections}$
On connections:
$\texttt{dbConnect}$ $\texttt{dbDisconnect}$ $\texttt{dbSendQuery}$ $\texttt{dbGetQuery}$ $\texttt{dbGetException}$ $\texttt{dbListResults}$
Convenience methods: $\texttt{dbListTables}$ $\texttt{dbReadTable}$ $\texttt{dbWriteTable}$ $\texttt{dbExistsTable}$ $\texttt{dbRemoveTable}$ $\texttt{dbListFields}$
On transaction management:
$\texttt{dbCommit}$ $\texttt{dbRollback}$
On queries and result objects:
$\texttt{fetch}$ $\texttt{dbClearResult}$ $\texttt{dbColumnInfo}$ $\texttt{dbGetStatement}$ $\texttt{dbHasCompleted}$ $\texttt{dbGetRowsAffected}$ $\texttt{dbGetRowCount}$
On meta-data:
$\texttt{show summary}$ $\texttt{dbGetInfo}$
Examples

```r
## Not run:
## create a Oracle instance and create one connection.
ora <- Oracle()    ## or dbDriver("Oracle")
con <- dbConnect(ora, username = "scott", password = "tiger",
                 dbname = "inst1")

## if you are connecting to a local database
con <- dbConnect(ora, username = "scott", password = "tiger")

## execute a statement and fetch its output in chunks of no more
## than 5000 rows at a time
rs <- dbSendQuery(con, "select * from emp where deptno = 10")
while (!dbHasCompleted(rs)) {
  df <- fetch(rs, n = 5000)
  ## process df
}
dbClearResult(rs)    ## done with this query

## execute and fetch a statement with bind data
df <- dbGetQuery(con, "select * from emp where deptno = :1",
                 data = data.frame(deptno = 10))

## create a copy of emp table
dbGetQuery(con, "create table foo as select * from emp")

## execute and bind an INSERT statement
my.data = data.frame(empno = c(8001, 8002),
                     ename = c('MUKHIN', 'ABOYOUN'))
more.data = data.frame(empno = c(8003),
                       ename = c('JAMES'))
rs <- dbSendQuery(con, "insert into foo (empno, ename) values (:1, :2)",
                  data = my.data)

## execute with more data
execute(rs, data = more.data)
dbClearResult(rs)    ## done with this query

## ok, everything looks fine
dbCommit(con)

## a concise description of the driver
summary(ora)

## done with this connection
dbDisconnect(con)

## End(Not run)
```

OraConnection-class  
Class OraConnection
Description

An Oracle connection class implementing the R database interface (DBI) API.

Generators

The method `dbConnect` is the main generator.

Extends

Class "DBIConnection", directly. Class "DBIObject", by class "DBIConnection", distance 2.

Methods

- `dbDisconnect` signature(conn = "OraConnection"): ...
- `dbSendQuery` signature(conn = "OraConnection", statement = "character", prefetch = FALSE, bulk_read = 1): ...
- `dbGetQuery` signature(conn = "OraConnection", statement = "character", prefetch = FALSE, bulk_read = 1): ...
- `dbGetException` signature(conn = "OraConnection"): ...
- `dbListResults` signature(conn = "OraConnection"): ...
- `dbListTables` signature(conn = "OraConnection"): ...
- `dbReadTable` signature(conn = "OraConnection", name = "character"): ...
- `dbWriteTable` signature(conn = "OraConnection", name = "character", value = "data.frame"): ...
- `dbExistsTable` signature(conn = "OraConnection", name = "character"): ...
- `dbRemoveTable` signature(conn = "OraConnection", name = "character"): ...
- `dbListFields` signature(conn = "OraConnection", name = "character"): ...
- `dbCommit` signature(conn = "OraConnection"): ...
- `dbRollback` signature(conn = "OraConnection"): ...
- `dbGetInfo` signature(dbObj = "OraConnection"): ...
- `summary` signature(object = "OraConnection"): ...
- `show` signature(object = "OraConnection")

See Also

DBI classes: OraDriver-class OraConnection-class OraResult-class

Examples

```r
## Not run:
ora <- dbDriver("Oracle")
## connecting without a connect string
con <- dbConnect(ora, "scott", "tiger")
## connecting with a connection string with SID
```
host <- "myhost"
port <- 1521
sid <- "mysid"
connect.string <- paste(
  "(DESCRIPTION=",
  "(ADDRESS=('"'port='""))(PORT='"'host='""))",
  "(CONNECT_DATA=('"'sid='""))(sep='"')"
)

## use username/password authentication
con <- dbConnect(drv, username = "scott", password = "tiger",
                 dbname = connect.string)

## connecting with a connection string with service name
host <- "myhost"
port <- 1521
svc <- "mydb.example.com"
connect.string <- paste(
  "(DESCRIPTION=",
  "(ADDRESS=('"'port='"'host='""))(PORT='"'host='""))",
  "(CONNECT_DATA=('"'svc='""))(sep='"')"
)

## use username/password authentication
con <- dbConnect(drv, username = "scott", password = "tiger",
                 dbname = connect.string)

## Please refer to "Oracle Database Net Services Administrator's Guide", which
## has the topic "Connect Identifier and Connect Descriptor Syntax
## Characteristics"

dbListTables(con)

## End(Not run)

OraDriver-class

\textbf{Class} OraDriver

\textbf{Description}

An Oracle driver class implementing the R database interface (DBI) API.

\textbf{Generators}

The main generators are \texttt{dbDriver} and \texttt{Oracle}.

\textbf{Extends}

Class "DBIDriver", directly. Class "DBIObject", by class "DBIDriver", distance 2.
Methods

\begin{itemize}
\item \textbf{dbConnect} signature(drv = "OraDriver"): ...
\item \textbf{dbGetInfo} signature(dbObj = "OraDriver"): ...
\item \textbf{dbListConnections} signature(drv = "OraDriver"): ...
\item \textbf{dbUnloadDriver} signature(drv = "OraDriver"): ...
\item \textbf{summary} signature(object = "OraDriver"): ...
\item \textbf{show} signature(object = "OraDriver")
\end{itemize}

See Also

DBI classes: \texttt{OraConnection-class OraResult-class}

Examples

\begin{verbatim}
## Not run:
ora <- dbDriver("Oracle")
con <- dbConnect(ora, "scott", "tiger")

## End(Not run)
\end{verbatim}

\begingroup
\renewcommand{\arraystretch}{1.2}
\begin{tabular}{ll}
\hline
\texttt{OraResult-class} & \textit{Class OraResult} \\
\hline
\end{tabular}
\endgroup

Description

An Oracle query results class. This class encapsulates the result of a SQL statement.

Generators

The main generator is \texttt{dbSendQuery}.

Extends

Class "DBIResult", directly. Class "DBIObject", by class "DBIResult", distance 2.

Methods

\begin{itemize}
\item \textbf{dbClearResult} signature(res = "OraResult"): ...
\item \textbf{dbColumnInfo} signature(res = "OraResult"): ...
\item \textbf{dbGetInfo} signature(dbObj = "OraResult"): ...
\item \textbf{dbGetStatement} signature(res = "OraResult"): ...
\item \textbf{dbGetRowCount} signature(res = "OraResult"): ...
\item \textbf{dbGetRowsAffected} signature(res = "OraResult"): ...
\item \textbf{dbHasCompleted} signature(res = "OraResult"): ...
\end{itemize}
`fetch` signature(res = "OraResult", n = "numeric"): ...
`fetch` signature(res = "OraResult", n = "missing"): ...
`execute` signature(res = "OraResult"): ...
`summary` signature(object = "OraResult"): ...
`show` signature(object = "OraResult")

See Also

DBI classes: `OraDriver-class` `OraConnection-class` `OraResult-class`

Examples

```r
## Not run:
ora <- dbDriver("Oracle")
con <- dbConnect(ora, "scott", "tiger")
res <- dbSendQuery(con, "select * from emp")
fetch(res, n = 2)
fetch(res)
dbColumnInfo(res)
dbClearResult(res)

## End(Not run)
```

---

**summary-methods**

*Summarize an Oracle object*

**Description**

These methods are straight-forward implementations of the corresponding generic functions.

**Usage**

```r
## S4 method for signature 'OraDriver'
summary(object, ...)
## S4 method for signature 'ExtDriver'
summary(object, ...)
## S4 method for signature 'OraConnection'
summary(object, ...)
## S4 method for signature 'OraResult'
summary(object, ...)
```

**Arguments**

- `object` a driver, connection or result set object.
- `...` currently unused.
Value
description of object.

References
For the Oracle Database documentation see http://www.oracle.com/technetwork/indexes/documentation/index.html.

See Also
Oracle, dbConnect, dbSendQuery, dbGetQuery, dbClearResult, dbCommit, dbGetInfo, dbGetInfo.

Examples
```r
## Not run:
drv <- dbDriver("Oracle")
con <- dbConnect(drv, "scott", "tiger")
res <- dbSendQuery(con, "select * from emp")

summary(drv)
summary(con)
summary(res)
show(drv)
show(con)
show(res)

## End(Not run)
```
Index

*Topic classes
  ExtDriver-class, 21
  OraConnection-class, 25
  OraDriver-class, 27
  OraResult-class, 28

*Topic database
  dbCallProc-methods, 2
  dbCommit-methods, 2
  dbConnect-methods, 3
  dbDriver-methods, 9
  dbGetInfo-methods, 10
  dbListConnections-methods, 12
  dbReadTable-methods, 13
  dbSendQuery-methods, 18
  dbSetDataMappings-methods, 20
  ExtDriver-class, 21
  fetch-methods, 22
  Oracle, 23
  OraConnection-class, 25
  OraDriver-class, 27
  OraResult-class, 28
  summary-methods, 29

*Topic interface
  dbCallProc-methods, 2
  dbCommit-methods, 2
  dbConnect-methods, 3
  dbDriver-methods, 9
  dbGetInfo-methods, 10
  dbListConnections-methods, 12
  dbReadTable-methods, 13
  dbSendQuery-methods, 18
  dbSetDataMappings-methods, 20
  ExtDriver-class, 21
  fetch-methods, 22
  Oracle, 23
  OraConnection-class, 25
  OraDriver-class, 27
  OraResult-class, 28
  summary-methods, 29

*Topic methods
  dbCallProc-methods, 2
  dbCommit-methods, 2
  dbConnect-methods, 3
  dbDriver-methods, 9
  dbGetInfo-methods, 10
  dbListConnections-methods, 12
  dbReadTable-methods, 13
  dbSendQuery-methods, 18
  dbSetDataMappings-methods, 20
  fetch-methods, 22
  summary-methods, 29

  dbCallProc (dbCallProc-methods), 2
  dbCallProc, OraConnection-method (dbCallProc-methods), 2
  dbCallProc-methods, 2
  dbClearResult, 22, 24, 30
  dbClearResult (dbSendQuery-methods), 18
  dbClearResult, OraResult-method (dbSendQuery-methods), 18
  dbColumnInfo, 12, 20, 24
  dbColumnInfo (dbGetInfo-methods), 10
  dbColumnInfo, OraResult-method (dbGetInfo-methods), 10
  dbCommit, 2, 3, 6, 10, 11, 16, 19, 22, 24, 30
  dbCommit (dbCommit-methods), 2
  dbCommit, OraConnection-method (dbCommit-methods), 2
  dbCommit-methods, 2
  dbConnect, 2, 3, 6, 9–12, 16, 19, 22–24, 26, 30
  dbConnect (dbConnect-methods), 3
  dbConnect, ExtDriver-method (dbConnect-methods), 3
  dbConnect, OraDriver-method (dbConnect-methods), 3
  dbConnect-methods, 3
  dbDisconnect, 24
  dbDisconnect (dbConnect-methods), 3
INDEX

(dbListConnections-methods), 12
.dbListConnections, OraDriver-method
(dbListConnections-methods), 12
dbListConnections-methods, 9
dbListConnections, OraConnection-method
(dbListConnections-methods), 12
dbListConnections-methods, 13
dbListConnections, OraConnection, character-method
(dbListConnections-methods), 13
dbListResults-methods
(dbListConnections-methods), 12
dbListResults, OraConnection-method
(dbListConnections-methods), 12
dbListTables, 10, 11, 24
dbListTables, OraConnection-method
(dbListConnections-methods), 13
dbListTables-methods, 13
dbReadTable-methods, 13
dbReadTable, 2, 3, 6, 10–12, 16, 19, 22, 24, 30
dbReadTable, OraConnection-method
(dbReadTable-methods), 13
dbReadTable-methods, 13
dbRemoveTable-methods, 13
dbRemoveTable, 24
dbRemoveTable, OraConnection-method
(dbReadTable-methods), 13
dbRemoveTable-methods, 13
dbRollback-methods, 2
dbRollback, OraConnection-method
(dbCommit-methods), 2
dbRollback-methods, 2
dbSendQuery, 2, 3, 6, 10–12, 16, 20, 22, 24, 28, 30
dbSendQuery-methods, 18
dbSendQuery, OraConnection, character-method
(dbSendQuery-methods), 18
dbSendQuery-methods, 18
dbSetDataMappings-methods
(dbSetDataMappings-methods), 20
dbSendQuery-methods, 18
dbSendQuery-methods, 18
dbSetDataMappings-methods, 20
dbSendQuery-methods, 20
dbUnloadDriver-methods, 20
dbUnloadDriver, 24
dbUnloadDriver, OraDriver-method
(dbDriver-methods), 9
dbUnloadDriver-methods, 9
dbUnloadDriver, OraDriver-method
(dbDriver-methods), 9
dbWriteTable-methods, 24

dbHasCompleted-methods, 24
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
dbHasCompleted-methods, 10
db_write_table (db_read_table-methods), 13
execute (db_send_query-methods), 18
execute, OraResult-method (db_send_query-methods), 18
Extdriver-class, 21
Extproc, 21
Extproc (Oracle), 23
fetch, 2, 3, 6, 10, 11, 16, 19, 20, 24
fetch (fetch-methods), 22
fetch, OraResult-method (fetch-methods), 22
fetch-methods, 22
Oracle, 2, 3, 6, 10–12, 16, 19, 20, 22, 23, 27, 30
OraConnection-class, 25
OraDriver-class, 27
OraResult-class, 28
show, 24
show, Extdriver-method (summary-methods), 29
show, OraConnection-method (summary-methods), 29
show, OraDriver-method (summary-methods), 29
show, OraResult-method (summary-methods), 29
summary, 24
summary, Extdriver-method (summary-methods), 29
summary, OraConnection-method (summary-methods), 29
summary, OraDriver-method (summary-methods), 29
summary, OraResult-method (summary-methods), 29
summary-methods, 29