Package 'ROracle'

February 2, 2012

Version 1.1-1

Date 2012-02-01

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

License LGPL

URL http://www.oracle.com

Collate dbi.R oci.R zzz.R

Repository CRAN

Date/Publication 2012-02-02 07:44:10

R topics documented:

`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 roll backs the current transaction in an Oracle connection

Methods

conn a `OraConnection` object, as produced by the function `dbConnect`.

... currently unused.

See Also

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

Examples

```
## 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.

Methods

drv an object of class OraDriver.

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 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.

... any optional arguments.

Side Effects

A connection between R and an Oracle server is established.

References

For the Oracle Database documentaion 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 instance and create one connection.
drv <- dbDriver("Oracle")
con <- dbConnect(drv, username = "scott", password = "tiger")

## run an 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)
```

---

dbDriver-methods *Oracle implementation of the Database Interface (DBI) classes and drivers*

Description

Oracle driver initialization and closing

Methods

drvName character name of the driver to instantiate.

drv an object that inherits from `OraDriver` as created by `dbDriver`.

... any other arguments are passed to the driver `drvName`.

See Also

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

Examples

```
## 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)

## End(Not run)
```

dbGetInfo-methods *Database interface meta-data*

Description

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

Methods

dbObj any object that implements some functionality in the R interface to databases (a driver, a connection or a result set).

res an OraResult.

what an element of the output list.

... currently not being used.

See Also

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

Examples

```
## 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 straight-forward implementations of the corresponding generic functions.

Methods

drv an OraDriver.

conn an OraConnection.

... currently not used.

See Also

Oracle, dbGetInfo, dbColumnInfo, dbDriver, dbConnect, dbSendQuery

Examples

```
## Not run:
drv <- dbDriver("Oracle")
con1 <- dbConnect(drv, "scott", "tiger")
res1 <- dbSendQuery(con1, "select * from emp where deptno = 10")
res2 <- dbSendQuery(con1, "select * from emp where deptno = 20")
con2 <- dbConnect(drv, "scott", "tiger")
res3 <- dbSendQuery(con2, "select * from dept")

## get all active statements
for(con in dbListConnections(drv))
  for (res in dbListResults(con))
    print(dbGetStatement(res))

## End(Not run)
```

dbReadTable-methods *Convenience functions for manipulating DBMS tables*

Description

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

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.

Methods

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).

all a boolean specifying whether to look at all schemas.

full a boolean specifying whether to generate schema names. When 'all' is set to TRUE the output is a vector containing schema names followed by the table names. One can do matrix(..., ncol = 2) on the output. This way each row of the matrix will correspond to a table with the first column being a schema name and the second - the tabel name.

value a data.frame (or coercible to data.frame). Vectors of types logical, integer, numeric, and character are supported natively; all other vectors must be coercible to character. logical and integer vectors map to INTEGER columns, numeric to NUMBER (or BINARY_DOUBLE if ora.number is set to FALSE), and character to VARCHAR2(4000).

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 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.

overwrite a logical specifying whether to overwrite an existing table or not. Its default is FALSE.

append a logical specifying whether to append to an existing table in the DBMS. Its default is FALSE.

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

purge a logical specifying whether to add PURGE option to the DROP TABLE statement.

... any optional arguments.

Note

Note that the data.frame returned by dbReadTable only has primitive data, e.g., it does not coerce character data to factors. This is similar to the output produced by fetch and dbGetQuery.

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.

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.

See Also

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

Examples

```
## 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")

## End(Not run)
```

---

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.

Methods

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

... additional parameters.

See Also

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

Examples

```
## 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)

## 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

```
## 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)
```

fetch-methods *Fetch records from a previously executed query*

Description

This method is a straight-forward implementation of the corresponding generic function.

Details

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

Methods

res an OraResult object.

n maximum number of records to retrieve per fetch. Use n = -1 to retrieve all pending records.

... currently not used.

See Also

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

Examples

```
## 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)

# let's get all remaining records
data2 <- fetch(res, n = -1)

## End(Not run)
```

Oracle *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

```
Oracle()
```

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.

Value

An object `OraDriver` 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.

Transactions

The current implementation directly supports transaction commits and rollbacks on a connection-wide basis through calls to `dbCommit` and `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 `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 `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 documentaion see http://www.oracle.com/technetwork/indexes/documentation/index.html.

Author(s)

David A. James and Denis Mukhin

See Also

On database managers:

dbDriver dbUnloadDriver dbListConnections

On connections:

dbConnect dbDisconnect dbSendQuery dbGetQuery dbGetException dbListResults

Convenience methods: dbListTables dbReadTable dbWriteTable dbExistsTable dbRemoveTable dbListFields

On transaction management:

dbCommit dbRollback

On queries and result objects:

fetch dbClearResult dbColumnInfo dbGetStatement dbHasCompleted dbGetRowsAffected dbGetRowCount

On meta-data:

show summary dbGetInfo

Examples

```
## 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(depno = 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")`: ...

dbGetQuery `signature(conn = "OraConnection", statement = "character")`: ...

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

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

## End(Not run)
```

`OraDriver-class` *Class OraDriver*

Description

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

Generators

The main generators are `dbDriver` and `Oracle`.

Extends

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

Methods

dbConnect `signature(drv = "OraDriver")`: ...
dbGetInfo `signature(dbObj = "OraDriver")`: ...
dbListConnections `signature(drv = "OraDriver")`: ...
dbUnloadDriver `signature(drv = "OraDriver")`: ...
summary `signature(object = "OraDriver")`: ...
show `signature(object = "OraDriver")`

See Also

DBI classes: OraConnection-class OraResult-class

Examples

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

## End(Not run)
```

OraResult-class *Class OraResult*

Description

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

Generators

The main generator is dbSendQuery.

Extends

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

Methods

dbClearResult signature(res = "OraResult"): ...

dbColumnInfo signature(res = "OraResult"): ...

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

dbGetStatement signature(res = "OraResult"): ...

dbGetRowCount signature(res = "OraResult"): ...

dbGetRowsAffected signature(res = "OraResult"): ...

dbHasCompleted signature(res = "OraResult"): ...

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

```
## 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.

Index