Package ‘MonetDB.R’
March 21, 2016

Version  1.0.1
Title  Connect MonetDB to R
Author  Hannes Muehleisen [aut, cre], Anthony Damico [aut], Thomas Lumley [ctb]
Maintainer  Hannes Muehleisen <hannes@ewi.nl>
Imports  DBI (>= 0.3.1), digest (>= 0.6.4), methods, codetools
Enhances  dplyr (>= 0.3.0), MonetDBLite
Description  Allows to pull data from MonetDB into R. Includes a DBI implementation and a dplyr backend.
License  MPL (== 2.0)
URL  http://www.monetdb.org
SystemRequirements  MonetDB, available from http://www.monetdb.org or
MonetDBLite R package
NeedsCompilation  yes
Repository  CRAN
Date/Publication  2016-03-21 23:07:24

R topics documented:

control ................................................................. 2
dbSendUpdate ......................................................... 3
dbTransaction ......................................................... 4
mc ................................................................. 5
mdbapply ............................................................. 6
MonetDB.R .............................................................. 7
monetdb.read.csv ................................................... 8
monetdb.liststatus .................................................. 9
MonetDBLite .......................................................... 10
monetdbRtype ......................................................... 10
monetdb_queryinfo ............................................... 11
mq ............................................................. 11
sqlite-compatibility ............................................. 12
src_monetdb .......................................................... 13
Description

The MonetDB server can be controlled from the R shell using the functions described below. The general process is to generate a MonetDB database directory and startup script using `monetdb.server.setup`, then pass the path to the startup script to `monetdb.server.start`. This function will return the process id of the database server, which in turn can be passed to `monetdb.server.stop` to stop the database server again. The process ID of a running MonetDB server can also be queried using `monetdb.server.getpid`, which takes a DBI connection as a parameter. A better alternative to `monetdb.server.stop` is `monetdb.server.shutdown`, which takes a DBI connection to shut down the server.

All of these external server process control functions are deprecated in favor of MonetDBLite.

Usage

```r
monetdb.server.setup(database.directory, monetdb.program.path, 
dbname = "demo", dbport = 50000)
monetdb.server.start(bat.file)
monetdb.server.getpid(con)
monetdb.server.stop(correct.pid, wait = TRUE)
monetdb.server.shutdown(con)
```

Arguments

database.directory
Path to the directory where the initialization script and all data will be stored. Must be empty or non-existent.

monetdb.program.path
Path to the MonetDB installation

dbname
Database name to be created

dbport
TCP port for MonetDB to listen for connections. This port should not conflict with other running programs on your local computer. Two databases with the same port number cannot be accessed at the same time

bat.file
Path to the MonetDB startup script. This path is returned by `monetdb.server.setup`

correct.pid
Process ID of the running MonetDB server. This number is returned by `monetdb.server.start`

wait
Wait for the server to shut down or return immediately

con
A DBI connection to MonetDB

Value

`monetdb.server.setup` returns the path to a MonetDB startup script, which can used many times

`monetdb.server.start` returns the process id of the MonetDB database server
**dbSendUpdate**

*Send a data-altering SQL statement to the database.*

**Description**

`dbSendUpdate` is used to send a data-altering statement to a MonetDB database, e.g. `CREATE TABLE` or `INSERT`. As a convenience feature, a placeholder (?) character can be used in the SQL statement, and bound to parameters given in the varargs group before execution. This is especially useful when scripting database updates, since the parameters will be automatically quoted.

The `dbSendUpdateAsync` function is used when the database update is called from finalizers, to avoid very esoteric concurrency problems. Here, the update is not guaranteed to be immediately run. Also, the method returns immediately.

**Usage**

```r
dbSendUpdate( conn, statement, ..., async=FALSE )
```

**Arguments**

- `conn` A MonetDB.R database connection. Created using `dbConnect` with the MonetDB.R database driver.
- `statement` A SQL statement to be sent to the database, e.g. 'UPDATE' or 'INSERT'.
- `...` Parameters to be bound to '?' characters in the query, similar to JDBC.
- `async` Behave like `dbSendUpdateAsync`? Defaults to FALSE.

**Value**

Returns TRUE if the update was successful.

**See Also**

- `dbSendQuery`
dbTransaction

Create, commit or abort a database transaction.

Description

dbTransaction is used to switch the data from the normal auto-commiting mode into transactional mode. Here, changes to the database will not be permanent until dbCommit is called. If the changes are not to be kept around, you can use dbRollback to undo all the changes since dbTransaction was called.

Usage

dbTransaction(conn, ...)

Arguments

conn A MonetDB.R database connection. Created using dbConnect with the MonetDB.R database driver.

... Future use.

Value

Returns TRUE if the transaction command was successful.

Examples

```
## Not run:
# connect to MonetDB
conn <- dbConnect(MonetDB.R(), "monetdb://localhost/acs")
# create table
dbSendUpdate(conn, "CREATE TABLE foo(a INT, b VARCHAR(100))")
# insert value, bind parameters to placeholders in statement
dbSendUpdate(conn, "INSERT INTO foo VALUES(?,?), 42, "bar")
```

```
## End(Not run)
```
Description

\( \text{mc}(\ldots) \) provides a short way of connecting to a MonetDB database. It is equivalent to \text{dbConnect(MonetDB.R(),\ldots)}

Usage

\[
\text{mc(dbname="demo", user="monetdb", password="monetdb", host="localhost", port=50000, timeout=86400, wait=FALSE,language="sql",\ldots)}
\]

Arguments

- \text{dbname} \hspace{2cm} \text{Database name}
- \text{user} \hspace{2cm} \text{Username for database}
- \text{password} \hspace{2cm} \text{Password for database}
- \text{host} \hspace{2cm} \text{Host name of database server}
- \text{port} \hspace{2cm} \text{TCP Port number of database server}
- \text{timeout} \hspace{2cm} \text{Database connection and query timeout}
- \text{wait} \hspace{2cm} \text{Wait for DB to become available or not}
- \text{language} \hspace{2cm} \text{Database language to be used (probably "sql")}
- \ldots \hspace{2cm} \text{Unused}

Value

Returns a DBI connection to the specified MonetDB database.

See Also

- \text{dbConnect}

Examples

\[
\begin{align*}
\text{## Not run:} \\
\text{con <- mc(dbname="demo",hostname="localhost")}
\end{align*}
\]

\[
\begin{align*}
\text{## End(Not run)}
\end{align*}
\]
**mdbapply**  
*Apply a R function to a MonetDB table.*

---

**Description**

`mdbApply` is used to switch the data from the normal auto-committing mode into transactional mode. Here, changes to the database will not be permanent until `dbCommit` is called. If the changes are not to be kept around, you can use `dbRollback` to undo all the changes since `dbTransaction` was called.

**Usage**

`mdbapply(conn, table, fun, ...)`

**Arguments**

- **conn**  
  A MonetDB.R database connection. Created using `dbConnect` with the `MonetDB.R` database driver.

- **table**  
  A MonetDB database table. Can also be a view or temporary table.

- **fun**  
  A R function to be run on the database table. The function gets passed a single `data.frame` argument which represents the database table. The function needs to return a single vector (for now).

- **...**  
  Other parameters to be passed to the function

**Value**

Returns the result of the function applied to the database table.

**Examples**

```r
## Not run:
conn <- dbConnect(MonetDB.R(), "demo")
data(mtcars)
dbWriteTable(conn, "mtcars", mtcars)

mpgplus42 <- mdbapply(conn, "mtcars", "double", function(d) {
d$mtpg + 42
})
```

## End(Not run)
MonetDB.R

DBI database connector for MonetDB

Description

MonetDB.R creates a new DBI driver that can be used to connect and interact with MonetDB.

Usage

MonetDB.R()

details

The MonetDB.R function creates the R object which can be used to a call dbConnect which actually creates the connection. Since it has no parameters, it is most commonly used inline with the dbConnect call.

This package aims to provide a reasonably complete implementation of the DBI. A number of additional methods are provided: dbSendUpdate for database-altering statements, dbSendUpdateAsync for cleanup operations and monetdb.read.csv for database CSV import.

Value

Returns a driver object that can be used in calls to dbConnect.

See Also

dbConnect for documentation how to invoke the driver monetdb.server.setup to set up and start a local MonetDB server from R

Examples

```r
## Not run:
library(DBI)
conn <- dbConnect(MonetDB.R::MonetDB(), dbname = "demo")
dbWriteTable(conn, "iris", iris)
dbListTables(conn)
dbGetQuery(conn, "SELECT COUNT(*) FROM iris")
d <- dbReadTable(conn, "iris")

## End(Not run)
```
monetdb.read.csv

Import a CSV file into MonetDB

Description
Instruct MonetDB to read a CSV file, optionally also create the table for it.

Usage
```r
monetdb.read.csv (conn, files, tablename, header=TRUE, locked=FALSE, best.effort=FALSE, na.strings="", nrow.check=500, delim="", newline = "\n", quote = "\"", create=TRUE, col.names=NULL, lower.case.names=FALSE, sep=delim, ...)
```

Arguments
- **conn**: A MonetDB.R database connection. Created using `dbConnect` with the `MonetDB.R` database driver.
- **files**: A single string or a vector of strings containing the absolute file names of the CSV files to be imported.
- **tablename**: Name of the database table the CSV files should be imported in. Created if necessary.
- **header**: Whether or not the CSV files contain a header line.
- **locked**: Whether or not to disable transactions for import. Setting this to TRUE can greatly improve the import performance.
- **best.effort**: Use best effort flag when reading csv files and continue importing even if parsing of fields/lines fails.
- **na.strings**: Which string value to interpret as NA value.
- **nrow.check**: Amount of rows that should be read from the CSV when the table is being created to determine column types.
- **delim**: Field separator in CSV file.
- **newline**: Newline in CSV file, usually \n for UNIX-like systems and \r\r on Windows.
- **quote**: Quote character(s) in CSV file.
- **create**: Create table before importing?
- **lower.case.names**: Convert all column names to lowercase in the database?
- **col.names**: Optional column names in case the ones from CSV file should not be used
- **sep**: alias for `delim`
- **...**: Additional parameters. Currently not in use.

Value
Returns the number of rows imported if successful.
See Also

dbWriteTable in DBIConnection-class

Examples

```r
## Not run:
library(DBI)
# connect to MonetDB
conn <- dbConnect(MonetDB::MonetDB(), dbname = "demo")
# write test data to temporary CSV file
file <- tempfile()
write.table(iris, file, sep="",)
# create table and import CSV
MonetDB::monetdb.read.csv(conn, file, "iris")

## End(Not run)
```

monetdb.liststatus  Get list of available databases from monetdb

Description

The `monetdbd` daemon can be used to manage multiple MonetDB databases in UNIX-like systems. This function connects to it and retrieves information about the available databases. Please note that `monetdbd` has to be configured to allow TCP control connections first. This can be done by setting a passphrase, e.g. "examplepassphrase" (`monetdb set passphrase=examplepassphrase /path/to/dbfarm`) and then switching on remote control (`monetdb set control=true /path/to/dbfarm`).

Usage

`monetdb.liststatus(passphrase, host="localhost", port=50000L, timeout=86400L)`

Arguments

- `passphrase`  monetdb passphrase, see description
- `host`  hostname to connect to
- `port`  TCP port where `monetdbd` listens
- `timeout`  Connection timeout (seconds)

Value

A `data.frame` that contains various information about the available databases.

Examples

```r
## Not run:
print(monetdb.liststatus("mypasshprase")$dbname)

## End(Not run)
```
MonetDBLite  

*MonetDBLite DBI driver*

**Description**

MonetDBLite creates a new DBI driver to interact with MonetDBLite.

**Usage**

```r
MonetDBLite()
```

**Details**

The `MonetDBLite` function creates the R object which can be used to a call `dbConnect` which actually creates the connection. Since it has no parameters, it is most commonly used inline with the `dbConnect` call.

**Value**

Returns a MonetDBLite driver object that can be used in calls to `dbConnect`.

**Examples**

```r
## Not run:
library(DBI)
conn <- dbConnect(MonetDB.R::MonetDBLite())
## End(Not run)
```

---

**monetdbRtype**  

*Get the name of the R data type for a database type.*

**Description**

For a database data type, get the name of the R data type it is being translated to.

**Usage**

```r
monetdbRtype(dbType)
```

**Arguments**

- **dbType**: A database type string such as CHAR or INTEGER.

**Value**

String containing the R data type for the DB data type, e.g. character or numeric.
### monetdb_queryinfo

**monetdb_queryinfo**

Get information about the result set of a query without actually executing it. This is mainly needed for dplyr compatibility.

### Description

`monetdb_queryinfo(...)` is used to get the expected result set structure (# rows, # columns, column names, column types etc.) without actually running the query.

### Usage

```
monetdb_queryinfo(conn, query)
```

### Arguments

- **conn**: Database name
- **query**: SQL SELECT query to get information about

### Value

Environment with various entries, e.g.

- **cols** – number of columns
- **rows** – number of rows
- **types** – vector of column type from database (e.g. "VARCHAR" or "INT")
- **names** – vector of column names
- **tables** – vector of table names

### Examples

```
## Not run:
monetdb_queryinfo("demo","SELECT 1")

## End(Not run)
```

### mq

**mq**

Connect to a database, run a single SELECT query, and disconnect again.

### Description

`mq(....)` provides a short way to connect to a MonetDB database, run a single SELECT query, and disconnect again.
Usage

mq(dbname, query, ...)

Arguments

dbname Database name
query SQL SELECT query to run
... Other options for dbConnect

Value

Returns a data frame that contains the result of the passed query or an error if something went wrong.

See Also

dbConnect mc

Examples

## Not run:
mq("demo","SELECT 1")

## End(Not run)

---

sqlite-compatibility  Compatibility functions for RSQlite

Description

Some functions that RSQlite has and that we support to allow MonetDBLite being used as a drop-in replacement.

Usage

isIdCurrent(dbObj, ...)
initExtension(dbObj, ...)

Arguments

dbObj A MonetDB.R database connection. Created using dbConnect with the MonetDB.R database driver.
... Additional parameters. Currently not in use.
Description

Use `src_monetdb` to connect to an existing MonetDB database, and `tbl` to connect to tables within that database. Please note that the ORDER BY, LIMIT and OFFSET keywords are not supported in the query when using `tbl` on a connection to a MonetDB database. If you are running a local database, you only need to define the name of the database you want to connect to.

Usage

```r
src_monetdb(dbname, host = "localhost", port = 50000L, user = "monetdb",
password = "monetdb", con=FALSE, ...)
```

```r
## S3 method for class 'src_monetdb'
tbl(src, from, ...)
```

Arguments

- `dbname`: Database name
- `host, port`: Host name and port number of database (defaults to localhost:50000)
- `user, password`: User name and password (if needed)
- `con`: Existing DBI connection to MonetDB to be re-used
- `...`: for the src, other arguments passed on to the underlying database connector, `dbConnect`.
- `src`: a MonetDB src created with `src_monetdb`.
- `from`: Either a string giving the name of table in database, or SQL described a derived table or compound join.

Examples

```r
## Not run:
library(dplyr)
# Connection basics
# To connect to a database first create a src:
my_db <- MonetDB::src_monetdb(dbname="demo")
# Then reference a tbl within that src
my_tbl <- tbl(my_db, "my_table")
```

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

*Topic interface
  dbSendUpdate, 3
  MonetDB.R, 7
  monetdb.read.csv, 8
control, 2

db_analyze.MonetDBConnection (src_monetdb), 13

db_create_index.MonetDBConnection (src_monetdb), 13

db_insert_into.MonetDBConnection (src_monetdb), 13

db_query_fields.MonetDBConnection (src_monetdb), 13

db_query_fields.MonetDBEmbeddedConnection (src_monetdb), 13

db_query_rows.MonetDBConnection (src_monetdb), 13

db_query_rows.MonetDBEmbeddedConnection (src_monetdb), 13

db_save_query.MonetDBConnection (src_monetdb), 13

dbCommit.MonetDBConnection-method (dbTransaction), 4

dbConnect, 3–8, 10, 12

dbRollback.MonetDBConnection-method (dbTransaction), 4

dbSendQuery, 3

dbSendUpdate, 3, 7

dbSendUpdate.MonetDBConnection.character-method (dbSendUpdate), 3

dbSendUpdateAsync, 7

dbSendUpdateAsync (dbSendUpdate), 3

dbSendUpdateAsync.MonetDBConnection.character-method (dbSendUpdate), 3

dbTransaction, 4

dbTransaction.MonetDBConnection-method (dbTransaction), 4

initExtension (sqlite-compatibility), 12

initExtension.MonetDBConnection-method (sqlite-compatibility), 12

isIdCurrent (sqlite-compatibility), 12

isIdCurrent.MonetDBConnection-method (sqlite-compatibility), 12

isIdCurrent.MonetDBResult-method (sqlite-compatibility), 12

mc, 5, 12

mdbapply, 6

mdbapply.MonetDBConnection-method (mdbapply), 6

monet.read.csv (monetdb.read.csv), 8

MonetDB (MonetDB.R), 7

monetdb.liststatus
  (monetdb.liststatus), 9

MonetDB.R, 3, 4, 6, 7, 8, 12

monetdb.read.csv, 7, 8

monetdb.server getpid (control), 2

monetdb.server.setup, 7

monetdb.server.setup (control), 2

monetdb.server.shutdown (control), 2

monetdb.server.start (control), 2

monetdb.server.stop (control), 2

monetdb_queryinfo, 11

monetdb.liststatus, 9

MonetDBLite, 10

monetdblite (MonetDBLite), 10

MonetDBR (MonetDB.R), 7

MonetdbRtype, 10

MonetR (MonetDB.R), 7

mq, 11

RMonetDBLite (MonetDBLite), 10

rmonetdblite (MonetDBLite), 10

sample_frac.tbl_monetdb (src_monetdb), 13

sample_n.tbl_monetdb (src_monetdb), 13
INDEX

sql_subquery.MonetDBConnection (src_monetdb), 13
sqlite-compatibility, 12
src_desc.src_monetdb (src_monetdb), 13
src_monetdb, 13
src_translate_env.src_monetdb (src_monetdb), 13

tbl.src_monetdb (src_monetdb), 13