Package 'MonetDBLite'

July 27, 2018

Version 0.6.0

Title In-Process Version of 'MonetDB'

Description An in-process version of 'MonetDB', a SQL database designed for analytical tasks. Similar to 'SQLite', the database runs entirely inside the 'R' shell.

License MPL (== 2.0)

URL https://github.com/hannesmuehleisen/MonetDBLite-R

BugReports https://github.com/hannesmuehleisen/MonetDBLite-R/issues

Depends R (>= 3.2.0)

Imports DBI (>= 0.6), digest (>= 0.6.4), methods, codetools

Suggests assertthat, testthat, survey, nycflights13, RSQLite, dbplyr, dplyr, gdata, callr, devtools, DBItest, bit64

Collate mapi.R dbi.R dbapply.R dplyr.R control.R embedded.R zzz.R

Encoding UTF-8

NeedsCompilation yes

Author Hannes Mühleisen [aut, cre] (<https://orcid.org/0000-0001-8552-0029>),
Mark Raasveldt [ctb],
Thomas Lumley [ctb],
MonetDB B.V. [cph],
The Regents of the University of California [cph],
Kungliga Tekniska Hogskolan [cph],
Free Software Foundation, Inc. [cph]

Maintainer Hannes Mühleisen <hannes@cwi.nl>

Repository CRAN

Date/Publication 2018-07-27 09:40:03 UTC

R topics documented:

`control` *Control an external MonetDB server from the R shell.*

Description

The external 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 external 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 querid 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 discouraged in favor of embedded `MonetDBLite::MonetDBLite()` functions.

Unlike an embedded instance, initiating an external server process requires MonetDB home page installed on the user's system.

Usage

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

Arguments

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

`monetdb.program.path`
: Path to the MonetDB installation

dbname	Database name to be created
dbport	TCP port for external 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 external MonetDB startup script. This path is returned by monetdb.server.setup
correct.pid	Process ID of the running external MonetDB server. This number is returned by monetdb.server.start
wait	Wait for the server to shut down or return immediately
conn	A DBI connection to external MonetDB

Value

monetdb.server.setup returns the path to an external MonetDB startup script, which can used many times monetdb.server.start returns the process id of the external MonetDB database server

Examples

```
## Not run:
library(DBI)
startscript <- monetdb.server.setup("/tmp/database","/usr/local/monetdb/", "db1", 50001)
pid <- monetdb.server.start(startscript)
monetdb.server.stop(pid)
con <- dbConnect(MonetDB.R(), "monetdb://localhost:50001/db1")

## End(Not run)
```

---

dbSendUpdate *Send a data-altering SQL statement to the database. (DEPRECATED)*

Description

Note: This function has been deprecated and will be removed in a future release! 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

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

Arguments

conn	A MonetDBLite database connection. Created using dbConnect with the MonetDBLite 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

---

mc *Shorthand connection constructor for external MonetDB*

Description

mc(...) provides a short way of connecting to an external MonetDB database. It is equivalent to dbConnect(MonetDB.R(),...)

Usage

```
mc(dbname="demo", user="monetdb", password="monetdb", host="localhost", port=50000,
  timeout=60, wait=FALSE,language="sql",...)
```

Arguments

dbname	Database name
user	Username for database
password	Password for database
host	Host name of database server
port	TCP Port number of database server
timeout	Database connection and query timeout
wait	Wait for DB to become available or not
language	Database language to be used (probably "sql")
...	Unused

Value

Returns a DBI connection to the specified external MonetDB database.

See Also

dbConnect

Examples

```
## Not run:
  con <- mc(dbname="demo",hostname="localhost")

## End(Not run)
```

---

mdbapply *Apply a R function to an external MonetDB table.*

Description

dbApply uses the R UDF facilities in standalone MonetDB to apply the passed function to a table.

Usage

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

Arguments

conn	An external MonetDB.R database connection. Created using dbConnect with the MonetDB.R external database driver.
table	An external MonetDB database table. Can also be a view or temporary table.
fun	A R function to be run on the external 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

```
## Not run:
library(DBI)
con <- dbConnect(MonetDB.R(), "demo")
data(mtcars)
dbWriteTable(con, "mtcars", mtcars)

mpgplus42 <- mdbapply(con, "mtcars", "double", function(d) {
d$mpg + 42
})

## End(Not run)
```

ml *Shorthand connection constructor for embedded MonetDB*

Description

`ml(...)` provides a short way of connecting to an embedded MonetDB database. It is equivalent to `dbConnect(MonetDBLite(),...)`

Usage

```
ml(...)
```

Arguments

`...`	Parameters passed directly to `dbConnect()`

Value

Returns a DBI connection to the specified embedded MonetDB database.

See Also

`dbConnect`

Examples

```
library(DBI)
dbdir <- file.path( tempdir() , "ml" )
con <- ml(dbdir)
dbDisconnect(con, shutdown = TRUE)
```

MonetDB.R *DBI database connector for external MonetDB database*

Description

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

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.

All of the MonetDBLite::MonetDB.R() external server connection functions are discouraged in favor of embedded MonetDBLite::MonetDBLite() functions.

This package aims to provide a reasonably complete implementation of the DBI.

Value

Returns a driver object that can be used in calls to dbConnect with an external MonetDB database.

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

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

## End(Not run)
```

---

monetdb.read.csv *Import a CSV file into MonetDBLite*

Description

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

Usage

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

Arguments

conn	A MonetDBLite database connection. Created using dbConnect with the MonetDBLite 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.
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

```
# initiate a MonetDBLite server
library(DBI)
dbdir <- file.path( tempdir() , 'readcsv' )
con <- dbConnect( MonetDBLite::MonetDBLite() , dbdir )

# write test data to temporary CSV file
file <- tempfile()
write.table(iris, file, sep=",", row.names=FALSE)

# create table and import CSV
```

```
monetdb.read.csv(con, file, "iris")

dbDisconnect(con, shutdown=TRUE)
```

`monetdbd.liststatus` *Get list of available databases from external monetdbd*

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" (`monetdbd set passphrase=examplepassphrase /path/to/dbfarm`) and then switching on remote control (`monetdbd set control=true /path/to/dbfarm`).

Usage

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

Arguments

`passphrase`	`monetdbd` 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

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

## End(Not run)
```

MonetDBLite *MonetDBLite DBI driver*

Description

MonetDBLite creates a new DBI driver to interact with MonetDBLite

Usage

```
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

```
library(DBI)
con <- dbConnect(MonetDBLite::MonetDBLite())
dbDisconnect(con, shutdown=TRUE)
```

monetdblite_shutdown *Shutdown MonetDBLite*

Description

monetdblite_shutdown terminates the running MonetDBLite instance

Usage

```
monetdblite_shutdown()
```

Details

This provides an alternative to dbDisconnect(con, shutdown=TRUE) when no connection is available.

Value

Returns TRUE.

Examples

```
library(DBI)
con <- dbConnect(MonetDBLite::MonetDBLite())
MonetDBLite::monetdblite_shutdown()
```

`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

```
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`.

`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 MonetDBLite database connection. Created using `dbConnect` with the `MonetDBLite` database driver.
`...`	Additional parameters. Currently not in use.

src_monetdb *dplyr integration from MonetDBLite*

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

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

src_monetdblite(dbdir = tempdir(), ...)
```

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.
dbdir	a directory to start MonetDBLite in

Examples

```
library(dplyr)
# To connect to a database first create a src:
dbdir <- file.path(tempdir(), "dplyrdir")
my_db <- MonetDBLite::src_monetdblite(dbdir)

# copy some data to DB
my_iris  <- copy_to(my_db, iris)

# create table object
my_iris2 <- tbl(my_db, 'iris')

# now you can call regular dplyr methods on table object

# ...

# shut down the database
MonetDBLite::monetdblite_shutdown()
```

Index