Package 'MonetDB.R'

July 18, 2014

Version 0.9.4

Date 2013-07-18

Title Connect MonetDB to R

Depends DBI (>= 0.2-7), digest (>= 0.6.4), methods

Description Allows to pull data from MonetDB into R

License MPL (== 1.1)

URL http://monetr.r-forge.r-project.org

Maintainer Hannes Muehleisen <hannes@cwi.nl>

SystemRequirements MonetDB installation, available at http://www.monetdb.org

Author Hannes Muehleisen [aut, cre],Thomas Lumley [ctb],Anthony Damico [ctb]

NeedsCompilation yes

Repository CRAN

Date/Publication 2014-07-18 10:44:53

R topics documented:

`control` *Control a MonetDB server from the R shell.*

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.

Usage

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

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 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`

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

Examples

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

## End(Not run)
```

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

```
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

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)
```

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:
conn <- dbConnect(MonetDB.R(), "monetdb://localhost/acs")
dbSendUpdate(conn, "CREATE TABLE foo(a INT,b VARCHAR(100))")
dbTransaction(conn)
dbSendUpdate(conn, "INSERT INTO foo VALUES(?,?)", 42, "bar")
dbCommit(conn)
dbTransaction(conn)
dbSendUpdate(conn, "INSERT INTO foo VALUES(?,?)", 43, "bar")
dbRollback(conn)

# only 42 will be in table foo

## End(Not run)
```

mc *Shorthand connection constructor for MonetDB*

Description

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

Usage

```
mc(dbname="demo", user="monetdb", password="monetdb", host="localhost", port=50000,
  timeout=86400, 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 MonetDB database.

See Also

`dbConnect`

Examples

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

## End(Not run)
```

MonetDB.R *DBI database connector and virtual data object 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

```
## Not run:
conn <- dbConnect(MonetDB.R(), "monetdb://localhost/demo")
dbListTables(conn)
data(iris)
dbWriteTable(conn, "iris", iris)
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

```
 monetdb.read.csv (conn, files, tablename, nrows, header=TRUE,
locked=FALSE, na.strings="", nrow.check=500, delim=",",
newline = "\\n", quote = "\"", ...)
```

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.
`nrows`	Total number of rows to import. Can be an upper bound.
`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.
`na.strings`	Which string value to interpret as `NA` value.
`...`	Additional parameters. Currently not in use.
`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.

Value

Returns the number of rows imported if successful.

See Also

`dbWriteTable` in `DBIConnection-class`

Examples

```
## Not run:
# connect to MonetDB
conn <- dbConnect(MonetDB.R(), "monetdb://localhost/demo")
# write test data to temporary CSV file
data(iris)
file <- tempfile()
write.table(iris, file, sep=",")
# create table and import CSV
monetdb.read.csv(conn, file, "iris", 150)

## End(Not run)
```

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

monetdbGetTransferredBytes

Get and reset MAPI traffic counters

Description

The MonetDB.R connector keeps track on the amount of bytes transferred to and from the database. The counters are reset after this function is called. This is a developer feature and is not very useful to typical users.

Usage

```
monetdbGetTransferredBytes ()
```

Value

Returns a list with two entries, `bytes.in` and `bytes.out`.

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`.

`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` *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)
```

Index