# Package 'RMariaDB'

August 27, 2020

**Title** Database Interface and 'MariaDB' Driver

**Version** 1.0.10

**Description** Implements a 'DBI'-compliant interface to 'MariaDB' (<https://mariadb.org/>) and 'MySQL' (<https://www.mysql.com/>) databases.

**License** GPL-3

**URL** https://rmariadb.r-dbi.org, https://github.com/r-dbi/RMariaDB, https://downloads.mariadb.org/connector-c/

**BugReports** https://github.com/r-dbi/RMariaDB/issues

**Depends** R (>= 2.8.0)

**Imports** bit64, DBI (>= 1.1.0), hms (>= 0.5.0), methods, Rcpp (>= 0.12.4)

**Suggests** DBItest (>= 1.7.0), rprojroot, testthat

**LinkingTo** BH, plogr, Rcpp

**Encoding** UTF-8

**NeedsCompilation** yes

**RoxygenNote** 7.1.1.9000

**SystemRequirements** libmariadb-client-lgpl-dev or libmariadbclient-dev or libmysqlclient-dev, with libssl-dev (deb), mariadb-connector-c-devel or mariadb-devel (rpm), mariadb-connector-c or mysql-connector-c (brew)

**Collate** 'MariaDBConnection.R' 'MariaDBDriver.R' 'MariaDBResult.R' 'RMariaDB.R' 'RcppExports.R' 'coerce.R' 'connect.R' 'default.R' 'export.R' 'names.R' 'query.R' 'quote.R' 'rownames.R' 'table.R' 'transaction.R' 'utils.R' 'zzz.R'

**Author** Kirill Müller [aut, cre] (<https://orcid.org/0000-0002-1416-3412>), Jeroen Ooms [aut] (<https://orcid.org/0000-0002-4035-0289>), David James [aut], Saikat DebRoy [aut], Hadley Wickham [aut],

Jeffrey Horner [aut],
R Consortium [fnd],
RStudio [cph]

**Maintainer** Kirill Müller <krlmlr+r@mailbox.org>

**Repository** CRAN

**Date/Publication** 2020-08-27 11:40:25 UTC

# R topics documented:



---

`RMariaDB-package` *RMariaDB: Database Interface and 'MariaDB' Driver*

---

## Description

Implements a 'DBI'-compliant interface to 'MariaDB' (<https://mariadb.org/>) and 'MySQL' (<https://www.mysql.com/>) databases.

## Author(s)

**Maintainer**: Kirill Müller <krlmlr+r@mailbox.org> (ORCID)

Authors:

- Jeroen Ooms (ORCID)
- David James
- Saikat DebRoy
- Hadley Wickham
- Jeffrey Horner

Other contributors:

- R Consortium [funder]
- RStudio [copyright holder]

## See Also

Useful links:

- https://rmariadb.r-dbi.org
- https://github.com/r-dbi/RMariaDB
- https://downloads.mariadb.org/connector-c/
- Report bugs at https://github.com/r-dbi/RMariaDB/issues

---

`Client-flags` *Client flags*

---

## Description

Use for the `client.flag` argument to `dbConnect()`, multiple flags can be combined with a bitwise or (see Logic). The flags are provided for completeness.

## See Also

The `flags` argument at https://mariadb.com/kb/en/library/mysql_real_connect.

## Examples

```
## Not run:
library(DBI)
library(RMariaDB)
con1 <- dbConnect(MariaDB(), client.flag = CLIENT_COMPRESS)
con2 <- dbConnect(
  MariaDB(),
  client.flag = CLIENT_COMPRESS | CLIENT_SECURE_CONNECTION
)

## End(Not run)
```

---

`dbConnect,MariaDBDriver-method`
*Connect/disconnect to a MariaDB DBMS*

---

## Description

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

## Usage

```
## S4 method for signature 'MariaDBDriver'
dbConnect(
  drv,
  dbname = NULL,
  username = NULL,
  password = NULL,
  host = NULL,
  unix.socket = NULL,
  port = 0,
  client.flag = 0,
  groups = "rs-dbi",
  default.file = NULL,
  ssl.key = NULL,
  ssl.cert = NULL,
  ssl.ca = NULL,
  ssl.capath = NULL,
  ssl.cipher = NULL,
  ...,
  bigint = c("integer64", "integer", "numeric", "character"),
  timeout = 10
)

MariaDB()
```

## Arguments

| | |
|---|---|
| drv | an object of class MariaDBDriver or MariaDBConnection. |
| dbname | string with the database name or NULL. If not NULL, the connection sets the default database to this value. |
| username, password | Username and password. If username omitted, defaults to the current user. If password is omitted, only users without a password can log in. |
| host | string identifying the host machine running the MariaDB server or NULL. If NULL or the string "localhost", a connection to the local host is assumed. |
| unix.socket | (optional) string of the unix socket or named pipe. |
| port | (optional) integer of the TCP/IP default port. |
| client.flag | (optional) integer setting various MariaDB client flags, see Client-flags for details. |
| groups | string identifying a section in the default.file to use for setting authentication parameters (see MariaDB()). |
| default.file | string of the filename with MariaDB client options, only relevant if groups is given. The default value depends on the operating system (see references), on Linux and OS X the files ~/.my.cnf and ~/.mylogin.cnf are used. |
| ssl.key | (optional) string of the filename of the SSL key file to use. |
| ssl.cert | (optional) string of the filename of the SSL certificate to use. |

| | |
|---|---|
| ssl.ca | (optional) string of the filename of an SSL certificate authority file to use. |
| ssl.capath | (optional) string of the path to a directory containing the trusted SSL CA certificates in PEM format. |
| ssl.cipher | (optional) string list of permitted ciphers to use for SSL encryption. |
| ... | Unused, needed for compatibility with generic. |
| bigint | The R type that 64-bit integer types should be mapped to, default is bit64::integer64, which allows the full range of 64 bit integers. |
| timeout | Connection timeout, in seconds. Use Inf or a negative value for no timeout. |

## References

Configuration files: https://mariadb.com/kb/en/library/configuring-mariadb-with-mycnf/

## Examples

```
## Not run:
# Connect to a MariaDB database running locally
con <- dbConnect(RMariaDB::MariaDB(), dbname = "mydb")
# Connect to a remote database with username and password
con <- dbConnect(RMariaDB::MariaDB(), host = "mydb.mycompany.com",
  user = "abc", password = "def")
# But instead of supplying the username and password in code, it's usually
# better to set up a group in your .my.cnf (usually located in your home
directory). Then it's less likely you'll inadvertently share them.
con <- dbConnect(RMariaDB::MariaDB(), group = "test")

# Always cleanup by disconnecting the database
dbDisconnect(con)

## End(Not run)

# All examples use the rs-dbi group by default.
if (mariadbHasDefault()) {
  con <- dbConnect(RMariaDB::MariaDB(), dbname = "test")
  con
  dbDisconnect(con)
}
if (mariadbHasDefault()) {
# connect to a database and load some data
con <- dbConnect(RMariaDB::MariaDB(), dbname = "test")
dbWriteTable(con, "USArrests", datasets::USArrests, temporary = TRUE)

# query
rs <- dbSendQuery(con, "SELECT * FROM USArrests")
d1 <- dbFetch(rs, n = 10)      # extract data in chunks of 10 rows
dbHasCompleted(rs)
d2 <- dbFetch(rs, n = -1)      # extract all remaining data
dbHasCompleted(rs)
dbClearResult(rs)
dbListTables(con)
```

```
# clean up
dbDisconnect(con)
}
```

---

dbDataType,MariaDBConnection-method

*Determine the SQL Data Type of an S object*

---

## Description

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

## Usage

```
## S4 method for signature 'MariaDBConnection'
dbDataType(dbObj, obj, ...)

## S4 method for signature 'MariaDBDriver'
dbDataType(dbObj, obj, ...)
```

## Arguments

| | |
|---|---|
| dbObj | A MariaDBDriver or MariaDBConnection object. |
| obj | R/S-Plus object whose SQL type we want to determine. |
| ... | any other parameters that individual methods may need. |

## Examples

```
dbDataType(RMariaDB::MariaDB(), "a")
dbDataType(RMariaDB::MariaDB(), 1:3)
dbDataType(RMariaDB::MariaDB(), 2.5)
```

---

dbFetch,MariaDBResult-method

*Execute a SQL statement on a database connection.*

---

## Description

To retrieve results a chunk at a time, use dbSendQuery(), dbFetch(), then dbClearResult(). Alternatively, if you want all the results (and they'll fit in memory) use dbGetQuery() which sends, fetches and clears for you. For data manipulation queries (i.e. queries that do not return data, such as UPDATE, DELETE, etc.), dbSendStatement() serves as a counterpart to dbSendQuery(), while dbExecute() corresponds to dbGetQuery().

## Usage

```
## S4 method for signature 'MariaDBResult'
dbFetch(res, n = -1, ..., row.names = FALSE)

## S4 method for signature 'MariaDBConnection,character'
dbSendQuery(conn, statement, params = NULL, ...)

## S4 method for signature 'MariaDBConnection,character'
dbSendStatement(conn, statement, params = NULL, ...)

## S4 method for signature 'MariaDBResult'
dbBind(res, params, ...)

## S4 method for signature 'MariaDBResult'
dbClearResult(res, ...)

## S4 method for signature 'MariaDBResult'
dbGetStatement(res, ...)
```

## Arguments

| | |
|---|---|
| res | A MariaDBResult object. |
| n | Number of rows to retrieve. Use -1 to retrieve all rows. |
| ... | Unused. Needed for compatibility with generic. |
| row.names | Either TRUE, FALSE, NA or a string.<br>If TRUE, always translate row names to a column called "row_names". If FALSE, never translate row names. If NA, translate rownames only if they're a character vector.<br>A string is equivalent to TRUE, but allows you to override the default name.<br>For backward compatibility, NULL is equivalent to FALSE. |
| conn | an MariaDBConnection object. |
| statement | a character vector of length one specifying the SQL statement that should be executed. Only a single SQL statement should be provided. |
| params | A list of query parameters to be substituted into a parameterised query. |

## Examples

```
if (mariadbHasDefault()) {
con <- dbConnect(RMariaDB::MariaDB(), dbname = "test")
dbWriteTable(con, "arrests", datasets::USArrests, temporary = TRUE)

# Run query to get results as dataframe
dbGetQuery(con, "SELECT * FROM arrests limit 3")

# Send query to pull requests in batches
res <- dbSendQuery(con, "SELECT * FROM arrests")
data <- dbFetch(res, n = 2)
```

```
data
dbHasCompleted(res)

dbClearResult(res)
dbDisconnect(con)
}
```

---

mariadb-tables *Read and write MariaDB tables.*

---

## Description

These methods read or write entire tables from a MariaDB database.

## Usage

```
## S4 method for signature 'MariaDBConnection,character'
dbReadTable(conn, name, ..., row.names = FALSE, check.names = TRUE)

## S4 method for signature 'MariaDBConnection,character,data.frame'
dbWriteTable(
  conn,
  name,
  value,
  field.types = NULL,
  row.names = FALSE,
  overwrite = FALSE,
  append = FALSE,
  ...,
  temporary = FALSE
)

## S4 method for signature 'MariaDBConnection,character,character'
dbWriteTable(
  conn,
  name,
  value,
  field.types = NULL,
  overwrite = FALSE,
  append = FALSE,
  header = TRUE,
  row.names = FALSE,
  nrows = 50,
  sep = ",",
  eol = "\n",
  skip = 0,
  quote = "\"",
  temporary = FALSE,
```

```
  ...
)

## S4 method for signature 'MariaDBConnection'
dbListTables(conn, ...)

## S4 method for signature 'MariaDBConnection'
dbListObjects(conn, prefix = NULL, ...)

## S4 method for signature 'MariaDBConnection,character'
dbExistsTable(conn, name, ...)

## S4 method for signature 'MariaDBConnection,character'
dbRemoveTable(conn, name, ..., temporary = FALSE, fail_if_missing = TRUE)
```

## Arguments

| | |
|---|---|
| conn | a MariaDBConnection object, produced by DBI::dbConnect() |
| name | a character string specifying a table name. |
| ... | Unused, needed for compatibility with generic. |
| row.names | Either TRUE, FALSE, NA or a string.<br>If TRUE, always translate row names to a column called "row_names". If FALSE, never translate row names. If NA, translate rownames only if they're a character vector.<br>A string is equivalent to TRUE, but allows you to override the default name.<br>For backward compatibility, NULL is equivalent to FALSE. |
| check.names | If TRUE, the default, column names will be converted to valid R identifiers. |
| value | A data frame. |
| field.types | Optional, overrides default choices of field types, derived from the classes of the columns in the data frame. |
| 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. If appending, then the table (or temporary table) must exist, otherwise an error is reported. Its default is FALSE. |
| temporary | If TRUE, creates a temporary table that expires when the connection is closed. For dbRemoveTable(), only temporary tables are considered if this argument is set to TRUE. |
| header | logical, does the input file have a header line? Default is the same heuristic used by read.table(), i.e., TRUE if the first line has one fewer column that the second line. |
| nrows | number of lines to rows to import using read.table from the input file to create the proper table definition. Default is 50. |
| sep | field separator character |

| | |
|---|---|
| eol | End-of-line separator |
| skip | number of lines to skip before reading data in the input file. |
| quote | the quote character used in the input file (defaults to \".) |
| prefix | A fully qualified path in the database's namespace, or NULL. This argument will be processed with dbUnquoteIdentifier(). If given the method will return all objects accessible through this prefix. |
| fail_if_missing | If FALSE, dbRemoveTable() succeeds if the table doesn't exist. |

## Value

A data.frame in the case of dbReadTable(); otherwise a logical indicating whether the operation was successful.

## Note

The data.frame returned by dbReadTable() only has primitive data, e.g., it does not coerce character data to factors. Temporary tables are ignored for dbExistsTable() and dbListTables() due to limitations of the underlying C API. For this reason, a prior existence check is performed only before creating a regular persistent table; an attempt to create a temporary table with an already existing name will fail with a message from the database driver.

## Examples

```
if (mariadbHasDefault()) {
con <- dbConnect(RMariaDB::MariaDB(), dbname = "test")

# By default, row names are written in a column to row_names, and
# automatically read back into the row.names()
dbWriteTable(con, "mtcars", mtcars[1:5, ], temporary = TRUE)
dbReadTable(con, "mtcars")
dbReadTable(con, "mtcars", row.names = FALSE)
}
```

---

# mariadbClientLibraryVersions

*MariaDB Check for Compiled Versus Loaded Client Library Versions*

---

## Description

This function prints out the compiled and loaded client library versions.

## Usage

```
mariadbClientLibraryVersions()
```

## Value

A named integer vector of length two, the first element representing the compiled library version and the second element representing the loaded client library version.

## Examples

```
mariadbClientLibraryVersions()
```

---

`mariadbHasDefault` *Check if default database is available.*

---

## Description

RMariaDB examples and tests connect to a database defined by the `rs-dbi` group in ~/.my.cnf. This function checks if that database is available, and if not, displays an informative message. `mariadbDefault()` works similarly but throws a testthat skip condition on failure, making it suitable for use in tests.

## Usage

```
mariadbHasDefault()

mariadbDefault()
```

## Examples

```
if (mariadbHasDefault()) {
  db <- dbConnect(RMariaDB::MariaDB(), dbname = "test")
  dbListTables(db)
  dbDisconnect(db)
}
```

---

`result-meta` *Database interface meta-data.*

---

## Description

See documentation of generics for more details.

## Usage

```
## S4 method for signature 'MariaDBResult'
dbColumnInfo(res, ...)

## S4 method for signature 'MariaDBResult'
dbGetRowsAffected(res, ...)

## S4 method for signature 'MariaDBResult'
dbGetRowCount(res, ...)

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

## Arguments

| | |
|---|---|
| res | An object of class MariaDBResult |
| ... | Ignored. Needed for compatibility with generic |

## Examples

```
if (mariadbHasDefault()) {
con <- dbConnect(RMariaDB::MariaDB(), dbname = "test")
dbWriteTable(con, "t1", datasets::USArrests, temporary = TRUE)

rs <- dbSendQuery(con, "SELECT * FROM t1 WHERE UrbanPop >= 80")
rs

dbGetStatement(rs)
dbHasCompleted(rs)
dbColumnInfo(rs)

dbFetch(rs)
rs

dbClearResult(rs)
dbDisconnect(con)
}
```

---

# transactions *DBMS Transaction Management*

---

## Description

Commits or roll backs the current transaction in an MariaDB connection. Note that in MariaDB DDL statements (e.g. CREATE TABLE) cannot be rolled back.

## Usage

```
## S4 method for signature 'MariaDBConnection'
dbBegin(conn, ...)

## S4 method for signature 'MariaDBConnection'
dbCommit(conn, ...)

## S4 method for signature 'MariaDBConnection'
dbRollback(conn, ...)
```

## Arguments

| | |
|---|---|
| conn | a MariaDBConnection object, as produced by `DBI::dbConnect()`. |
| ... | Unused. |

## Examples

```
if (mariadbHasDefault()) {
con <- dbConnect(RMariaDB::MariaDB(), dbname = "test")
df <- data.frame(id = 1:5)

dbWriteTable(con, "df", df, temporary = TRUE)
dbBegin(con)
dbExecute(con, "UPDATE df SET id = id * 10")
dbGetQuery(con, "SELECT id FROM df")
dbRollback(con)

dbGetQuery(con, "SELECT id FROM df")

dbDisconnect(con)
}
```

# Index