Package 'Andromeda'

April 8, 2025

Type Package

Title Asynchronous Disk-Based Representation of Massive Data

Version 1.0.0

Date 2025-04-08

Maintainer Martijn Schuemie <schuemie@ohdsi.org>

Description Storing very large data objects on a local drive, while still making it possible to manipulate the data in an efficient manner.

License Apache License 2.0

VignetteBuilder knitr

URL https://github.com/OHDSI/Andromeda

BugReports https://github.com/OHDSI/Andromeda/issues

Depends dplyr

Imports DBI,
zip,
methods,
dbplyr,
tidyselect,
cli,
rlang,
pillar,
duckdb

Suggests testthat,
stringr,
knitr,
rmarkdown,
tibble,
rJava,
rcmdcheck

LazyData false

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Encoding UTF-8

Contents

`andromeda` *Create an Andromeda object*

Description

By default the `Andromeda` object is created in the systems temporary file location. You can override this by specifying a folder using `options(andromedaTempFolder = "c:/andromedaTemp")`, where `"c:/andromedaTemp"` is the folder to create the Andromeda objects in.

Although in general Andromeda is well-behaved in terms of memory usage, it can consume a lot of memory for specific operations such as sorting and aggregating. By default the memory usage is limited to 75% of the physical memory. However it is possible to set another limit by using `options(andromedaMemoryLimit = 2.5)`, where `2.5` is the number of GB to use at most. One GB is 1,000,000,000 bytes.

Usage

```
andromeda(..., options = list())
```

Arguments

`...`	Named objects. See details for what objects are valid. If no objects are provided, an empty Andromeda is returned.
`options`	A named list of options. Currently the only supported option is 'threads' (see example). All other options are ignored.

Details

Valid objects are data frames, Andromeda tables, or any other dplyr table.

Value

Returns an Andromeda object.

Examples

```
andr <- andromeda(cars = cars, iris = iris)

names(andr)
# [1] 'cars' 'iris'

andr$cars %>% filter(speed > 10) %>% collect()
# # A tibble: 41 x 2
# speed dist
# <dbl> <dbl>
# 1 11 17
# ...

close(andr)

# Use multiple threads for queries
andr <- andromeda(cars = cars, iris = iris, options = list(threads = 8))
```

Andromeda-class *The Andromeda class*

Description

The Andromeda class is an S4 object.

This class provides the ability to work with data objects in R that are too large to fit in memory. Instead, these objects are stored on disk. This is slower than working from memory, but may be the only viable option.

Show the names of the tables in an Andromeda object.

Usage

```
## S4 method for signature 'Andromeda'
show(object)

## S4 method for signature 'Andromeda'
x$name

## S4 replacement method for signature 'Andromeda'
x$name <- value

## S4 replacement method for signature 'Andromeda'
x[[i]] <- value
```

```
## S4 method for signature 'Andromeda'
x[[i]]

## S4 method for signature 'Andromeda'
names(x)

## S4 method for signature 'Andromeda'
length(x)

## S4 method for signature 'Andromeda'
close(con, ...)
```

Arguments

object	An Andromeda object.
x	An Andromeda object.
name	The name of a table in the Andromeda object.
value	A data frame, Andromeda table, or other 'DBI' table.
i	The name of a table in the Andromeda object.
con	An Andromeda object.
...	Included for compatibility with generic close() method.

Value

A vector of names.

Tables

An Andromeda object has zero, one or more tables. The list of table names can be retrieved using the names() method. Tables can be accessed using the dollar sign syntax, e.g. andromeda$myTable, or double-square-bracket syntax, e.g. andromeda[["myTable"]]

Permanence

To mimic the behavior of in-memory objects, when working with data in Andromeda the data is stored in a temporary location on the disk. You can modify the data as you can see fit, and when needed can save the data to a permanent location. Later this data can be loaded to a temporary location again and be read and modified, while keeping the saved data as is.

Inheritance

The Andromeda inherits directly from duckdb_connection As such, it can be used as if it is a duckdb_connection. duckdb is an R wrapper around 'duckdb', a low-weight but powerful single-user SQL database that can run from a single file on the local file system.

See Also

andromeda()

Examples

```
andr <- andromeda(cars = cars, iris = iris)

names(andr)
# [1] 'cars' 'iris'

close(andr)
```

`appendToTable` *Append to an Andromeda table*

Description

Append a data frame, Andromeda table, or result of a query on an `Andromeda` table to an existing `Andromeda` table.

If data from another `Andromeda` is appended, a batch-wise copy process is used, which will be slower than when appending data from within the same `Andromeda` object.

Important: columns are appended based on column name, not on column order. The column names should therefore be identical (but not necessarily in the same order).

Usage

```
appendToTable(tbl, data)
```

Arguments

`tbl`	An `Andromeda` table. This must be a base table (i.e. it cannot be a query result).
`data`	The data to append. This can be either a data.frame or another Andromeda table.

Value

Returns no value. Executed for the side-effect of appending the data to the table.

Examples

```
andr <- andromeda(cars = cars)
nrow(andr$cars)
# [1] 50

appendToTable(andr$cars, cars)
nrow(andr$cars)
# [1] 100

appendToTable(andr$cars, andr$cars %>% filter(speed > 10))
nrow(andr$cars)
# [1] 182

close(andr)
```

batchApply *Apply a function to batches of data in an Andromeda table*

Description

Apply a function to batches of data in an Andromeda table

Usage

```
batchApply(tbl, fun, ..., batchSize = 1e+05, progressBar = FALSE, safe = FALSE)
```

Arguments

tbl	An Andromeda table (or any other 'DBI' table).
fun	A function where the first argument is a data frame.
...	Additional parameters passed to fun.
batchSize	Number of rows to fetch at a time.
progressBar	Show a progress bar?
safe	Create a copy of tbl first? Allows writing to the same Andromeda as being read from.

Details

This function is similar to the lapply() function, in that it applies a function to sets of data. In this case, the data is batches of data from an Andromeda table. Each batch will be presented to the function as a data frame.

Value

Invisibly returns a list of objects, where each object is the output of the user-supplied function applied to a batch

See Also

groupApply()

Examples

```
andr <- andromeda(cars = cars)

fun <- function(x) {
  return(nrow(x))
}

result <- batchApply(andr$cars, fun, batchSize = 25)

result
# [[1]]
# [1] 25
#
# [[2]]
```

```
# [1] 25

close(andr)
```

`batchTest` *Apply a boolean test to batches of data in an Andromeda table and terminate early*

Description

Apply a boolean test to batches of data in an Andromeda table and terminate early

Usage

```
batchTest(tbl, fun, ..., batchSize = 1e+05)
```

Arguments

`tbl`	An `Andromeda` table (or any other 'DBI' table).
`fun`	A function where the first argument is a data frame and returns a logical value.
`...`	Additional parameters passed to `fun`.
`batchSize`	Number of rows to fetch at a time.

Details

This function applies a boolean test function to sets of data and terminates at the first `FALSE`.

Value

Returns `FALSE` if any of the calls to the user-supplied function returned `FALSE`, else returns `TRUE`.

Examples

```
andr <- andromeda(cars = cars)

fun <- function(x) {
  is.unsorted(x %>% select(speed) %>% collect())
}

result <- batchTest(andr$cars, fun, batchSize = 25)

result
# [1] FALSE

close(andr)
```

copyAndromeda *Copy Andromeda*

Description

Creates a complete copy of an Andromeda object. Object attributes are not copied.

Usage

```
copyAndromeda(andromeda, options = list())
```

Arguments

andromeda	The Andromeda object to copy.
options	A list containing Andromeda options. Currently the only supported option is 'threads'. Setting `options = list(threads = 10)` will set the database used by Andromeda to use 10 threads.

Value

The copied Andromeda object.

Examples

```
andr <- andromeda(cars = cars, iris = iris)

andr2 <- copyAndromeda(andr)

names(andr2)
# [1] 'cars' 'iris'

close(andr)
close(andr2)
```

createIndex *Create an index on one or more columns in an Andromeda table*

Description

Create an index on one or more columns in an Andromeda table

Usage

```
createIndex(tbl, columnNames, unique = FALSE, indexName = NULL)
```

Arguments

tbl	An Andromeda table (or any other 'DBI' table).
columnNames	A vector of column names (character) on which the index is to be created.
unique	Should values in the column(s) be enforced to be unique?
indexName	The name of the index. If not provided, a random name will be generated.

Details

Indices can speed up subsequent queries that use the indexed columns, but can take time to create, and will take additional space on the drive.

Value

Invisibly returns the input table.

See Also

`listIndices()`, `removeIndex()`

Examples

```
andr <- andromeda(cars = cars)

createIndex(andr$cars, "speed")

# Will be faster now that speed is indexed:
andr$cars %>%
  filter(speed == 10) %>%
  collect()

close(andr)
```

---

getAndromedaTempDiskSpace

Get the available disk space in Andromeda temp

Description

Attempts to determine how much disk space is still available in the Andromeda temp folder. This function uses Java, so will only work if the `rJava` package is installed.

By default the Andromeda temp folder is located in the system temp space, but the location can be altered using `options(andromedaTempFolder = "c:/andromedaTemp")`, where `"c:/andromedaTemp"` is the folder to create the Andromeda objects in.

Usage

```
getAndromedaTempDiskSpace(andromeda = NULL)
```

Arguments

`andromeda`	Optional: provide an Andromeda object for which to get the available disk space. Normally all Andromeda objects use the same temp folder, but the user could have altered it.

Value

The number of bytes of available disk space in the Andromeda temp folder. Returns NA if unable to determine the amount of available disk space, for example because `rJava` is not installed, or because the user doesn't have the rights to query the available disk space.

Examples

```
# Get the number of available gigabytes:
getAndromedaTempDiskSpace() / 1024^3
#123.456
```

`groupApply` *Apply a function to groups of data in an Andromeda table*

Description

Apply a function to groups of data in an Andromeda table

Usage

```
groupApply(
  tbl,
  groupVariable,
  fun,
  ...,
  batchSize = 1e+05,
  progressBar = FALSE,
  safe = FALSE
)
```

Arguments

`tbl`	An `Andromeda` table (or any other 'DBI' table).
`groupVariable`	The variable to group by
`fun`	A function where the first argument is a data frame.
`...`	Additional parameters passed to fun.
`batchSize`	Number of rows fetched from the table at a time. This is not the number of rows to which the function will be applied. Included mostly for testing purposes.
`progressBar`	Show a progress bar?
`safe`	Create a copy of `tbl` first? Allows writing to the same Andromeda as being read from.

Details

This function applies a function to groups of data. The groups are identified by unique values of the `groupVariable`, which must be a variable in the table.

Value

Invisibly returns a list of objects, where each object is the output of the user-supplied function applied to a group.

See Also

`batchApply()`

Examples

```
andr <- andromeda(cars = cars)

fun <- function(x) {
  return(tibble::tibble(speed = x$speed[1], meanDist = mean(x$dist)))
}

result <- groupApply(andr$cars, "speed", fun)
result <- bind_rows(result)
result
# # A tibble: 19 x 2
# speed meanDist
# <dbl> <dbl>
# 1 4 6
# 2 7 13
# 3 8 16
# ...

close(andr)
```

`isAndromeda` *Check whether an object is an Andromeda object*

Description

Check whether an object is an Andromeda object

Usage

```
isAndromeda(x)
```

Arguments

`x`	The object to check.

Details

Checks whether an object is an Andromeda object.

Value

A logical value.

isAndromedaTable *Is the object an Andromeda table?*

Description

Is the object an Andromeda table?

Usage

```
isAndromedaTable(tbl)
```

Arguments

tbl	A reference to an Andromeda table

Value

TRUE or FALSE

Examples

```
## Not run:
andr <- andromeda(cars = cars)
isAndromedaTable(andr$cars)
close(andr)

## End(Not run)
```

isValidAndromeda *Check whether an Andromeda object is still valid*

Description

Check whether an Andromeda object is still valid

Usage

```
isValidAndromeda(x)
```

Arguments

x	The Andromeda object to check.

Details

Checks whether an Andromeda object is still valid, or whether it has been closed.

Value

A logical value.

Examples

```
andr <- andromeda(cars = cars, iris = iris)

isValidAndromeda(andr)
# TRUE

close(andr)

isValidAndromeda(andr)
# FALSE
```

`listIndices` *List all indices on an Andromeda table*

Description

List all indices on an Andromeda table

Usage

```
listIndices(tbl)
```

Arguments

`tbl`	An `Andromeda` table (or any other 'DBI' table).

Details

Lists any indices that may have been created using the `createIndex()` function.

Value

Returns a tibble listing the indices, indexed columns, and whether the index is unique.

See Also

`createIndex()`, `removeIndex()`

Examples

```
andr <- andromeda(cars = cars)

createIndex(andr$cars, "speed")

listIndices(andr$cars)
# # A tibble: 1 x 5
# indexSequenceId indexName                unique columnSequenceId columnName
#           <int> <chr>                    <lgl>             <int> <chr>
#1              0 idx_ocy8we9j2i7ld0rshgb4 FALSE                 0 speed

close(andr)
```

`loadAndromeda` *Load Andromeda from file*

Description

Load Andromeda from file

Usage

```
loadAndromeda(fileName, options = list())
```

Arguments

`fileName`	The path where the object was saved using `saveAndromeda()`.
`options`	A list containing Andromeda options. Currently the only supported option is 'threads'. Setting `options = list(threads = 10)` will set the database used by Andromeda to use 10 threads.

Value

An `Andromeda` object.

See Also

`saveAndromeda()`

Examples

```
# For this example we create an Andromeda object and save it to
# a temporary file locationL
fileName <- tempfile()
andr <- andromeda(cars = cars)
saveAndromeda(andr, fileName)

# Using loadAndromeda to load the object back:
andr <- loadAndromeda(fileName)

# Don't forget to close Andromeda when you are done:
close(andr)

# Cleaning up the file used in this example:
unlink(fileName)
```

names.tbl_Andromeda *Get the column names of an Andromeda table*

Description

Get the column names of an Andromeda table

Usage

```
## S3 method for class 'tbl_Andromeda'
names(x)
```

Arguments

x	An table in an Andromeda object

Value

A character vector of column names

Examples

```
andr <- andromeda(cars = cars)
names(andr$cars)
# [1] "speed" "dist"
close(andr)
```

names<-,Andromeda-method *Set table names in an Andromeda object*

Description

names(andromedaObject) must be set to a character vector with length equal to the number of tables in the andromeda object (i.e. length(andromedaObject)). The user is responsible for setting valid table names (e.g. not using SQL keywords or numbers as names) This function treats Andromeda table names as case insensitive so if the only difference between the new names and old names is the case then the names will not be changed.

Usage

```
## S4 replacement method for signature 'Andromeda'
names(x) <- value
```

Arguments

x	An Andromeda object
value	A character vector with the same length as the number of tables in x

Examples

```
andr <- andromeda(cars = cars, iris = iris)
names(andr) <- c("CARS", "IRIS")
names(andr)
# [1] "CARS" "IRIS"
close(andr)
```

---

names<-.tbl_Andromeda *Set column names of an Andromeda table*

Description

Set column names of an Andromeda table

Usage

```
## S3 replacement method for class 'tbl_Andromeda'
names(x) <- value
```

Arguments

x	A reference to a table in an andromeda object. (see examples)
value	A character vector of new names that must have length equal to the number of columns in the table.

Examples

```
andr <- andromeda(cars = cars)
names(andr$cars) <- toupper(names(andr$cars))
names(andr$cars)
# [1] "SPEED" "DIST"
close(andr)
```

---

removeIndex *Removes an index from an Andromeda table*

Description

Removes an index from an Andromeda table

Usage

```
removeIndex(tbl, columnNames = NULL, indexName = NULL)
```

Arguments

tbl	An Andromeda table (or any other 'DBI' table).
columnNames	A vector of column names (character) on which the index was created. If not provided, then the indexName argument must be provided.
indexName	The name of the index. If not provided, the columnNames argument must be provided.

Details

Remove an index created using the `createIndex()` function. Either the index name or the column names on which the index was created must be provided.

Value

Invisibly returns the input table.

See Also

`createIndex()`, `listIndices()`

Examples

```
andr <- andromeda(cars = cars)

createIndex(andr$cars, "speed")

# Will be faster now that speed is indexed:
andr$cars %>%
  filter(speed == 10) %>%
  collect()

removeIndex(andr$cars, "speed")

close(andr)
```

`restoreDate` *Restore dates*

Description

This function has been deprecated since Andromeda v0.5 preserves dates.

Usage

```
restoreDate(x)
```

Arguments

`x`	A numeric vector representing dates.

Value

A vector of type `Date`.

See Also

`restorePosixct()`

Examples

```
myData <- data.frame(startDate = as.Date(c("2000-01-01", "2001-01-31", "2004-12-31")))
andr <- andromeda(myData = myData)

andr$myData %>%
  collect() %>%
  mutate(startDate = restoreDate(startDate))
# # A tibble: 3 x 1
# startDate
# <date>
# 1 2000-01-01
# 2 2001-01-31
# 3 2004-12-31

close(andr)
```

---

restorePosixct *Restore timestamps*

Description

This function has been deprecated since Andromeda v0.5 preserves POSIXct datetimes.

Usage

```
restorePosixct(x)
```

Arguments

x	A numeric vector representing timestamps

Value

A vector of type `POSIXct`.

See Also

`restoreDate()`

Examples

```
myData <- data.frame(startTime = as.POSIXct(c("2000-01-01 10:00",
                                              "2001-01-31 11:00",
                                              "2004-12-31 12:00")))
andr <- andromeda(myData = myData)

andr$myData %>%
  collect() %>%
  mutate(startTime = restorePosixct(startTime))
# # A tibble: 3 x 1
# startTime
# <dttm>
```

```
# 1 2000-01-01 10:00:00
# 2 2001-01-31 11:00:00
# 3 2004-12-31 12:00:00

close(andr)
```

---

saveAndromeda *Save Andromeda to file*

Description

Saves the Andromeda object in a zipped file. Note that by default the Andromeda object is automatically closed by saving it to disk. This is due to a limitation of the underlying technology ('duckdb'). To keep the connection open, use maintainConnection = TRUE. This will first create a temporary copy of the Andromeda object. Note that this can be substantially slower.

Usage

```
saveAndromeda(
  andromeda,
  fileName,
  maintainConnection = FALSE,
  overwrite = TRUE
)
```

Arguments

andromeda	An object of class Andromeda.
fileName	The path where the object will be written.
maintainConnection	Should the connection be maintained after saving? If FALSE, the Andromeda object will be invalid after this operation, but saving will be faster.
overwrite	If the file exists, should it be overwritten? If FALSE and the file exists, an error will be thrown.

Value

Returns no value. Executed for the side-effect of saving the object to disk.

See Also

loadAndromeda

loadAndromeda()

Examples

```
andr <- andromeda(cars = cars)

# For this example we'll use a temporary file location:
fileName <- tempfile()

saveAndromeda(andr, fileName)

# Cleaning up the file used in this example:
unlink(fileName)
```

Index