Package 'dbflobr'

October 30, 2021

Title Read and Write Files to SQLite Databases

Version 0.2.1

Description Reads and writes files to SQLite databases <https://www.sqlite.org/index.html> as flobs (a flob is a blob that preserves the file extension).

License MIT + file LICENSE

URL https://github.com/poissonconsulting/dbflobr

BugReports https://github.com/poissonconsulting/dbflobr/issues

Depends R (>= 3.5)

Imports blob, chk, clisymbols, crayon, DBI, flobr, glue, rlang, RSQLite

Suggests covr, knitr, rmarkdown, testthat, withr

VignetteBuilder knitr

Encoding UTF-8

Language en-US

RoxygenNote 7.1.2

NeedsCompilation no

Author Sebastian Dalgarno [aut] (<https://orcid.org/0000-0002-3658-4517>), Joe Thorley [aut] (<https://orcid.org/0000-0002-7683-4592>), Evan Amies-Galonski [aut, cre] (<https://orcid.org/0000-0003-1096-2089>), Poisson Consulting [cph, fnd]

Maintainer Evan Amies-Galonski <evan@poissonconsulting.ca>

Repository CRAN

Date/Publication 2021-10-30 09:50:10 UTC

R topics documented:

`add_blob_column` *Add blob column*

Description

Add named empty blob column to SQLite database

Usage

```
add_blob_column(column_name, table_name, conn)
```

Arguments

`column_name`	A string of the name of the BLOB column.
`table_name`	A string of the name of the existing table.
`conn`	A SQLite connection object.

Value

Modified SQLite database.

Examples

```
conn <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
DBI::dbWriteTable(conn, "Table1", data.frame(IntColumn = c(1L, 2L)))
DBI::dbReadTable(conn, "Table1")
add_blob_column("BlobColumn", "Table1", conn)
DBI::dbReadTable(conn, "Table1")
DBI::dbDisconnect(conn)
```

delete_flob *Delete flob*

Description

Delete a flob from a SQLite database.

Usage

```
delete_flob(column_name, table_name, key, conn)
```

Arguments

column_name	A string of the name of the BLOB column.
table_name	A string of the name of the existing table.
key	A data.frame whose columns and values are used to filter the table to a single row (this in combination with the column_name argument are used to target a single cell within the table to modify).
conn	A SQLite connection object.

Value

An invisible copy of the deleted flob.

Examples

```
flob <- flobr::flob_obj
conn <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
DBI::dbWriteTable(conn, "Table1", data.frame(IntColumn = c(1L, 2L)))
key <- data.frame(IntColumn = 2L)
write_flob(flob, "BlobColumn", "Table1", key, conn, exists = FALSE)
DBI::dbReadTable(conn, "Table1")
delete_flob("BlobColumn", "Table1", key, conn)
DBI::dbReadTable(conn, "Table1")
DBI::dbDisconnect(conn)
```

import_all_flobs *Import all flobs.*

Description

Import flobs to SQLite database from directory. Table and column names are matched to directory names within main directory. Values in file names are matched to table primary key to determine where to write flob.

Usage

```
import_all_flobs(
  conn,
  dir = ".",
  sep = "_-_",
  pattern = ".*",
  sub = FALSE,
  exists = FALSE,
  replace = FALSE
)
```

Arguments

conn	A SQLite connection object.
dir	A string of the path to the directory to import the files from. Files need to be within nested folders like 'table1/column1/a.csv'. This structure is created automatically if save_all_flobs() function is used.
sep	A string of the separator between values in file names.
pattern	A regular expression specifying the pattern file names must match.
sub	A logical scalar specifying whether to import flobs based on their filename (sub = FALSE) or the name of their subdirectory (sub = TRUE) which must only contain 1 file. If sub = NA and replace = TRUE then the names of the subdirectories are used irrespective of whether they include files and existing flobs are deleted if the corresponding subdirectory is empty. If sub = TRUE or sub = NA then recursion is just one subfolder deep.
exists	A logical scalar specifying whether the column must (TRUE) or mustn't (FALSE) already exist or whether it doesn't matter (NA). IF FALSE, a new BLOB column is created.
replace	A flag indicating whether to replace existing flobs (TRUE) or not (FALSE).

Value

An invisible named list indicating directory path, file names and whether files were successfully written to database.

Examples

```
conn <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
DBI::dbGetQuery(conn, "CREATE TABLE Table1 (CharColumn TEXT PRIMARY KEY NOT NULL)")
DBI::dbWriteTable(conn, "Table1", data.frame(CharColumn = c("a", "b")), append = TRUE)
flob <- flobr::flob_obj
write_flob(flob, "BlobColumn", "Table1", data.frame(CharColumn = "a"), conn)
dir <- file.path(tempdir(), "import_all")
save_all_flobs(conn = conn, dir = dir)
import_all_flobs(conn, dir, exists = TRUE, replace = TRUE)
DBI::dbDisconnect(conn)
```

`import_flobs` *Import flobs.*

Description

Import flobs to SQLite database column from directory. Values in file name are matched to table primary key to determine where to write flob.

Usage

```
import_flobs(
  column_name,
  table_name,
  conn,
  dir = ".",
  sep = "_-_",
  pattern = ".*",
  sub = FALSE,
  exists = FALSE,
  recursive = FALSE,
  replace = FALSE
)
```

Arguments

`column_name`	A string of the name of the BLOB column.
`table_name`	A string of the name of the existing table.
`conn`	A SQLite connection object.
`dir`	A string of the path to the directory to import files from.
`sep`	A string of the separator between values in file names.
`pattern`	A regular expression specifying the pattern file names must match.
`sub`	A logical scalar specifying whether to import flobs based on their filename (sub = FALSE) or the name of their subdirectory (sub = TRUE) which must only contain 1 file. If sub = NA and replace = TRUE then the names of the subdirectories are used irrespective of whether they include files and existing flobs are deleted if the corresponding subdirectory is empty. If sub = TRUE or sub = NA then recursion is just one subfolder deep.
`exists`	A logical scalar specifying whether the column must (TRUE) or mustn't (FALSE) already exist or whether it doesn't matter (NA). IF FALSE, a new BLOB column is created.
`recursive`	A flag indicating whether to recurse into file directory (TRUE) or not (FALSE).
`replace`	A flag indicating whether to replace existing flobs (TRUE) or not (FALSE).

Value

An invisible named vector indicating file name and whether the file was successfully written to database.

Examples

```
conn <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
DBI::dbGetQuery(conn, "CREATE TABLE Table1 (CharColumn TEXT PRIMARY KEY NOT NULL)")
DBI::dbWriteTable(conn, "Table1", data.frame(CharColumn = c("a", "b")), append = TRUE)
key <- data.frame(CharColumn = "a", stringsAsFactors = FALSE)[0,,drop = FALSE]
dir <- tempdir()
write.csv(key, file.path(dir, "a.csv"))
import_flobs("BlobColumn", "Table1", conn, dir)
DBI::dbDisconnect(conn)
```

---

read_flob *Read flob*

Description

Read a flob from a SQLite database.

Usage

```
read_flob(column_name, table_name, key, conn, slob = FALSE)
```

Arguments

column_name	A string of the name of the BLOB column.
table_name	A string of the name of the existing table.
key	A data.frame whose columns and values are used to filter the table to a single row (this in combination with the column_name argument are used to target a single cell within the table to modify).
conn	A SQLite connection object.
slob	A logical scalar specifying whether to process as slobs (serialized blobs) instead of flobs. If NA, the function will adapt accordingly.

Value

A flob or blob.

Examples

```
flob <- flobr::flob_obj
conn <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
DBI::dbWriteTable(conn, "Table1", data.frame(IntColumn = c(1L, 2L)))
key <- data.frame(IntColumn = 2L)
write_flob(flob, "BlobColumn", "Table1", key, conn, exists = FALSE)
read_flob("BlobColumn", "Table1", key, conn)
DBI::dbDisconnect(conn)
```

`save_all_flobs` *Save all flobs.*

Description

Rename flobs from a SQLite database and save to directory.

Usage

```
save_all_flobs(
  table_name = NULL,
  conn,
  dir = ".",
  sep = "_-_",
  sub = FALSE,
  replace = FALSE,
  geometry = FALSE
)
```

Arguments

`table_name`	A vector of character strings indicating names of tables to save flobs from. By default all tables are included.
`conn`	A SQLite connection object.
`dir`	A string of the path to the directory to save the files in.
`sep`	A string of the separator used to construct file names from values.
`sub`	A logical scalar specifying whether to save all existing files in a subdirectory of the same name (sub = TRUE) or all possible files in a subdirectory of the same name (sub = NA) or not nest files within a subdirectory (sub = FALSE).
`replace`	A flag specifying whether to replace existing files. If sub = TRUE (or sub = NA) and replace = TRUE then all existing files within a subdirectory are deleted.
`geometry`	A flag specifying whether to search columns named geometry for flobs.

Value

An invisible named list of named vectors of the file names and new file names saved.

Examples

```
flob <- flobr::flob_obj
conn <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
DBI::dbGetQuery(conn, "CREATE TABLE Table1 (IntColumn INTEGER PRIMARY KEY NOT NULL)")
DBI::dbWriteTable(conn, "Table1", data.frame(IntColumn = c(1L, 2L)), append = TRUE)
key <- data.frame(IntColumn = 2L)
write_flob(flob, "BlobColumn", "Table1", key, conn, exists = FALSE)
dir <- tempdir()
save_all_flobs(conn = conn, dir = dir)
DBI::dbDisconnect(conn)
```

`save_flobs` *Save flobs.*

Description

Rename flobs from a SQLite database BLOB column and save to directory.

Usage

```
save_flobs(
  column_name,
  table_name,
  conn,
  dir = ".",
  sep = "_-_",
  sub = FALSE,
  replace = FALSE,
  slob_ext = NULL
)
```

Arguments

`column_name`	A string of the name of the BLOB column.
`table_name`	A string of the name of the existing table.
`conn`	A SQLite connection object.
`dir`	A string of the path to the directory to save the files in.
`sep`	A string of the separator used to construct file names from values.
`sub`	A logical scalar specifying whether to save all existing files in a subdirectory of the same name (sub = TRUE) or all possible files in a subdirectory of the same name (sub = NA) or not nest files within a subdirectory (sub = FALSE).
`replace`	A flag specifying whether to replace existing files. If sub = TRUE (or sub = NA) and replace = TRUE then all existing files within a subdirectory are deleted.
`slob_ext`	A string of the file extension to use if slobs (serialized blobs) are encountered. If slob_ext = NULL slobs will be ignored.

Value

An invisible named vector of the file names and new file names saved.

Examples

```
flob <- flobr::flob_obj
conn <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
DBI::dbGetQuery(conn, "CREATE TABLE Table1 (IntColumn INTEGER PRIMARY KEY NOT NULL)")
DBI::dbWriteTable(conn, "Table1", data.frame(IntColumn = c(1L, 2L)), append = TRUE)
key <- data.frame(IntColumn = 2L)
write_flob(flob, "BlobColumn", "Table1", key, conn, exists = FALSE)
dir <- tempdir()
save_flobs("BlobColumn", "Table1", conn, dir)
DBI::dbDisconnect(conn)
```

---

write_flob *Write flob*

Description

Write a flob to a SQLite database.

Usage

```
write_flob(flob, column_name, table_name, key, conn, exists = NA)
```

Arguments

flob	A flob.
column_name	A string of the name of the BLOB column.
table_name	A string of the name of the existing table.
key	A data.frame whose columns and values are used to filter the table to a single row (this in combination with the column_name argument are used to target a single cell within the table to modify).
conn	A SQLite connection object.
exists	A logical scalar specifying whether the column must (TRUE) or mustn't (FALSE) already exist or whether it doesn't matter (NA). IF FALSE, a new BLOB column is created.

Value

An invisible copy of flob.

Examples

```
flob <- flobr::flob_obj
conn <- DBI::dbConnect(RSQLite::SQLite(), ":memory:")
DBI::dbWriteTable(conn, "Table1", data.frame(IntColumn = c(1L, 2L)))
DBI::dbReadTable(conn, "Table1")
key <- data.frame(IntColumn = 2L)
write_flob(flob, "BlobColumn", "Table1", key, conn, exists = FALSE)
DBI::dbReadTable(conn, "Table1")
DBI::dbDisconnect(conn)
```

Index