

Package ‘qst’

November 20, 2020

Type Package

Title Store Tables in SQL Database

Version 0.1.2

Author Magnus Thor Torfason

Maintainer Magnus Thor Torfason <m@zulutime.net>

Description Provides functions for quickly writing (and reading back) a data.frame to file in 'SQLite' format. The name stands for *Store Tables using 'SQLite*', or alternatively for *Quick Store Tables* (either way, it could be pronounced as *Quest*). For data.frames containing the supported data types it is intended to work as a drop-in replacement for the 'write_*()' and 'read_*()' functions provided by similar packages.

License MIT + file LICENCE

Language en-US

Encoding UTF-8

LazyData true

Imports RSQLite, DBI, dplyr, dbplyr, tibble, magrittr

RoxygenNote 7.1.1

NeedsCompilation no

Repository CRAN

Date/Publication 2020-11-20 10:00:02 UTC

R topics documented:

qst	2
read_qst	2
write_qst	3

Index	4
--------------	----------

qst *Store Tables in SQL Database*

Description

This package provides functions for quickly writing (and reading) back a `data.frame` to file in `sqlite` format. The name stands for *Store Tables using SQLite*, or alternatively for *Quick Store Tables* (either way, it could be pronounced as *Quest*).

For `data.frames` containing the supported data types it is intended to work as a drop-in replacement for the `write_*`() and `read_*`() functions provided by packages such as `fst`, `feather`, `qs`, and `readr` packages (as well as the `writeRDS()` and `readRDS()` functions).

read_qst *Read a data.frame from an SQLite database*

Description

This function reads a `data.frame` from an `SQLite` database. The database has one table, named `data`, containing the data. Additional tables, prefixed with `meta_`, may be added in the future to support additional data types not supported in a native way by `SQLite`.

By specifying `lazy=TRUE`, the `data.frame` will not be read into memory on the read operation, but instead a lazy evaluated `data.frame` will be returned. This results in a near-instantaneous read operation, but subsequent operation will then be done from disk using `SQL` translation when the `data.frame` is passed to other functions or `collect()` is called on it.

Usage

```
read_qst(path, lazy = FALSE)
```

Arguments

<code>path</code>	The path to read from.
<code>lazy</code>	If <code>TRUE</code> , the full <code>data.frame</code> will not be read into memory, but instead a lazy evaluated <code>data.frame</code> will be returned.

Value

A `data.frame` read from the `SQLite` file found at `path`

Examples

```
# Write the cars data set to a file, then read it back
cars_db <- tempfile()
write_qst(cars, cars_db, indexes=list("speed"))
dat <- read_qst(cars_db)
unlink(cars_db)
```

write_qst	<i>Write a data.frame to an SQLite database</i>
-----------	---

Description

This function writes a data.frame to an SQLite database. The database has one table, named data, containing the data. Additional tables, prefixed with meta_, may be added in the future to support additional data types not supported in a native way by SQLite.

Usage

```
write_qst(x, path, ..., unique_indexes = NULL, indexes = NULL)
```

Arguments

x	A data.frame to be written to file. Supported column types are integer, numeric and character.
path	The path to write to.
...	Other parameters passed to methods.
unique_indexes	A list of character vectors. Each element of the list will create a new unique index over the specified column(s). Duplicate rows will result in failure.
indexes	A list of character vectors. Each element of the list will create a new index.

Value

The original data frame passed in x

Examples

```
# Write the cars data set to a file
cars_db <- tempfile()
write_qst(cars, cars_db, indexes=list("speed"))
unlink(cars_db)
```

Index

qst, [2](#)

read_qst, [2](#)

write_qst, [3](#)