

# Package ‘filehashSQLite’

February 19, 2015

**Version** 0.2-4

**Date** 2012-03-23

**Depends** R (>= 2.14.0), methods, filehash (>= 1.0), DBI, RSQLite

**Imports** filehash, DBI

**Title** Simple key-value database using SQLite

**Author** Roger D. Peng <rpeng@jhsph.edu>

**Maintainer** Roger D. Peng <rpeng@jhsph.edu>

**Description** Simple key-value database using SQLite as the backend

**License** GPL (>= 2)

**URL** <http://github.com/rdpeng/filehashsqlite>

**Repository** CRAN

**Date/Publication** 2012-03-23 21:45:30

**NeedsCompilation** no

## R topics documented:

filehashSQLite-class . . . . . 1

**Index** 4

---

filehashSQLite-class *Class "filehashSQLite"*

---

### Description

Create a ‘filehash’ database using SQLite as the backend

### Details

The “filehashSQLite” class represents a “filehash” key-value database using the SQLite DBM as the backend. Objects are stored in a single SQLite database table along with their keys.

## Objects from the Class

Objects can be created by calls of the form `new("filehashSQLite", ...)`. More likely, one will use the functions `dbCreate` and `dbInit` from the `filehash` package.

### Slots

**datafile** character, full path to the file in which the database should be stored

**dbcon** Object of class “`SQLiteConnection`”, a SQLite connection

**drv** ‘SQLite’ driver

**name** character, the name of the database

### Extends

Class “`filehash`”, directly.

### Methods

**dbDelete** `signature(db = "filehashSQLite", key = "character")`: delete a key-value pair from the database

**dbExists** `signature(db = "filehashSQLite", key = "character")`: check the existence of a specific key or vector of keys

**dbFetch** `signature(db = "filehashSQLite", key = "character")`: retrieve the value associated with a specific key

**dbInsert** `signature(db = "filehashSQLite", key = "character")`: insert a key-value pair

**dbList** `signature(db = "filehashSQLite")`: return character vector of keys currently stored in the database

**dbUnlink** `signature(db = "filehashSQLite")`: delete the entire database

**dbMultiFetch** `signature(db = "filehashSQLite", key = "character")`: return (as a named list) the values associated with a vector of keys

### Note

“`filehashSQLite`” databases have a “[” method that can be used to extract multiple elements in an efficient manner. The return value is a list with names equal to the keys passed to “[”. If there are keys passed to “[” that do not exist in the database, a warning is given.

The “`SQLite`” format for `filehash` uses an ASCII serialization of the data which could result in some rounding error for floating point numbers.

Note that if you use keys that are numbers coerced to character vectors, then you may have trouble with them being coerced to numeric. The SQLite database will see these key values and automatically convert them to numbers.

### Author(s)

Roger D. Peng

**Examples**

```
library(filehashSQLite)

dbCreate("myTestDB", type = "SQLite")
db <- dbInit("myTestDB", type = "SQLite")

set.seed(100)
db$a <- rnorm(100)
db$b <- "a character element"

with(db, mean(a))

cat(db$b, "\n")
```

# Index

\*Topic **classes**  
    filehashSQLite-class, [1](#)  
    [,filehashSQLite,character,ANY,ANY-method  
        (filehashSQLite-class), [1](#)  
  
    dbDelete,filehashSQLite,character-method  
        (filehashSQLite-class), [1](#)  
    dbDisconnect,filehashSQLite-method  
        (filehashSQLite-class), [1](#)  
    dbExists,filehashSQLite,character-method  
        (filehashSQLite-class), [1](#)  
    dbFetch,filehashSQLite,character-method  
        (filehashSQLite-class), [1](#)  
    dbInsert,filehashSQLite,character-method  
        (filehashSQLite-class), [1](#)  
    dbList,filehashSQLite-method  
        (filehashSQLite-class), [1](#)  
    dbMultiFetch,filehashSQLite,character-method  
        (filehashSQLite-class), [1](#)  
    dbUnlink,filehashSQLite-method  
        (filehashSQLite-class), [1](#)  
  
    filehashSQLite-class, [1](#)