

Package ‘nodbi’

November 11, 2019

Title 'NoSQL' Database Connector

Description Simplified document database manipulation and analysis, including support for many 'NoSQL' databases, including document databases ('Elasticsearch', 'CouchDB', 'MongoDB'), 'key-value' databases ('Redis'), and (with limitations) SQLite/json1.

Version 0.4.0

License MIT + file LICENSE

LazyData true

URL <https://github.com/ropensci/nodbi>

BugReports <https://github.com/ropensci/nodbi/issues>

Depends R (>= 2.10)

Encoding UTF-8

Language en-US

Imports data.table, jsonlite

Suggests sofa (>= 0.3.0), elastic (>= 1.0.0), mongolite (>= 1.6),
redux, RSQLite (>= 1.1), DBI, testthat

RoxygenNote 6.1.1

X-schema.org-applicationCategory Databases

X-schema.org-keywords database, MongoDB, Redis, Elasticsearch,
CouchDB, SQLite, NoSQL, JSON, documents

X-schema.org-isPartOf <https://ropensci.org>

NeedsCompilation no

Author Scott Chamberlain [aut, cre] (<<https://orcid.org/0000-0003-1444-9135>>),
Rich FitzJohn [aut],
Jeroen Ooms [aut],
Ralf Herold [aut] (<<https://orcid.org/0000-0002-8148-6748>>)

Maintainer Scott Chamberlain <sckott@protonmail.com>

Repository CRAN

Date/Publication 2019-11-11 22:50:02 UTC

R topics documented:

nodbi-package	2
contacts	3
diamonds	3
docdb_create	4
docdb_delete	5
docdb_exists	6
docdb_get	7
docdb_query	9
docdb_update	10
nodbi-defunct	12
src	12
src_couchdb	13
src_elastic	14
src_mongo	15
src_redis	15
src_sqlite	16
Index	17

nodbi-package	<i>Document database connector</i>
---------------	------------------------------------

Description

Supports NoSQL databases (Elasticsearch, CouchDB, MongoDB), key-value databases (Redis), and SQLite with json1 extension as in R package RSQLite.

Author(s)

Scott Chamberlain <sckott@protonmail.com>

Rich FitzJohn <rich.fitzjohn@gmail.com>

Jeroen Ooms <jeroen.ooms@stat.ucla.edu>

Ralf Herold <ralf.herold@mailbox.org>

contacts	<i>contacts</i>
----------	-----------------

Description

contacts

Usage

contacts

Format

A json string with ragged, nested contact details

diamonds	<i>diamonds</i>
----------	-----------------

Description

diamonds

Format

A data frame with 53940 rows and 10 variables:

- price price in US dollars (\\$326-\\$18,823)
- carat weight of the diamond (0.2-5.01)
- cut quality of the cut (Fair, Good, Very Good, Premium, Ideal)
- color diamond colour, from J (worst) to D (best)
- clarity a measurement of how clear the diamond is (I1 (worst), SI1, SI2, VS1, VS2, VVS1, VVS2, IF (best))
- x length in mm (0-10.74)
- y width in mm (0-58.9)
- z depth in mm (0-31.8)
- depth total depth percentage = $z / \text{mean}(x, y) = 2 * z / (x + y)$ (43-79)
- table width of top of diamond relative to widest point (43-95)

Sourcefrom **ggplot2**

docdb_create *Create documents*

Description

Create documents

Usage

```
docdb_create(src, key, value, ...)
```

Arguments

src	source object, result of call to an src function
key	(character) A key (collection for mongo)
value	(data.frame) A single data.frame
...	Ignored

Details

Note that with etcd, you have to prefix a key with a forward slash.

Examples

```
## Not run:
# CouchDB
src <- src_couchdb()
docdb_create(src, key="mtcars2", value=mtcars)
docdb_get(src, "mtcars2")

# Elasticsearch
src <- src_elastic()
if (docdb_exists(src, "mtcars")) docdb_delete(src, "mtcars")
if (docdb_exists(src, "iris")) docdb_delete(src, "iris")
if (docdb_exists(src, "diamonds_small")) docdb_delete(src, "diamonds_small")
docdb_create(src, key = "mtcars", value = mtcars)
docdb_create(src, key = "iris", value = iris)
docdb_create(src, key = "diamonds_small", value = diamonds[1:3000L,])

# Redis
src <- src_redis()
docdb_create(src, key = "mtcars", value = mtcars)
docdb_get(src, "mtcars")
docdb_delete(src, "mtcars")

# MongoDB
src <- src_mongo(collection = "mtcars")
docdb_create(src, key = "mtcars", value = mtcars)
docdb_get(src, "mtcars")
```

```

# SQLite
src <- src_sqlite()
if (docdb_exists(src, "mtcars")) docdb_delete(src, "mtcars")
docdb_create(src, key = "mtcars", value = mtcars)
docdb_get(src, "mtcars")
if (docdb_exists(src, "contacts")) docdb_delete(src, "contacts")
## contacts is a dataset included in this package
contacts_df <- data.frame(contacts, stringsAsFactors = FALSE)
docdb_create(src, key = "contacts", value = contacts_df)
docdb_get(src, "contacts")

## End(Not run)

```

docdb_delete

Delete documents

Description

Delete documents

Usage

```
docdb_delete(src, key, ...)
```

Arguments

src	source object, result of call to src, an object of class docdb_src
key	(character) A key (collection for mongo)
...	Ignored for now

Details

Note that with etcd, you have to prefix a key with a forward slash.

Examples

```

## Not run:
# couchdb
(src <- src_couchdb())
docdb_create(src, "mtcars", mtcars)
docdb_get(src, "mtcars")
docdb_delete(src, "mtcars")

# elasticsearch
src <- src_elastic()
if (docdb_exists(src, "iris")) docdb_delete(src, "iris")
docdb_create(src, "iris", iris)
Sys.sleep(2)

```

```

docdb_get(src, "iris")
docdb_delete(src, "iris")

# Redis
src <- src_redis()
docdb_create(src, key = "mtcars", value = mtcars)
docdb_get(src, "mtcars")
docdb_delete(src, "mtcars")

# mongo
src <- src_mongo(collection = "iris")
docdb_create(src, "iris", iris)
docdb_get(src, "iris")
docdb_delete(src, "iris")

# SQLite
src <- src_sqlite()
docdb_create(src, "iris", iris)
docdb_get(src, "iris")
docdb_delete(src, "iris")

## End(Not run)

```

docdb_exists

Check if a database exists

Description

Check if a database exists

Usage

```
docdb_exists(src, key, ...)
```

Arguments

src	source object, result of call to src
key	(character) A key. ignored for mongo
...	Ignored for now

Details

Note that with etcd, you have to prefix a key with a forward slash.

Value

logical, TRUE or FALSE

Note

no docdb_exists method for MongoDB at this time

Examples

```
## Not run:
# CouchDB
(src <- src_couchdb())
docout <- docdb_create(src, key = "mtcars2", value = mtcars)
docdb_exists(src, "mtcars2")
docdb_exists(src, "asdfadf")

# Elasticsearch
(src <- src_elastic())
if (docdb_exists(src, "iris")) docdb_delete(src, "iris")
docdb_exists(src, "iris")
docdb_create(src, "iris", iris)
docdb_exists(src, "iris")
docdb_exists(src, "adfadf")

# Redis
(src <- src_redis())
docdb_create(src, "mtcars", mtcars)
docdb_exists(src, "mtcars")
docdb_exists(src, "asdfasf")

# MongoDB
src <- src_mongo(collection = "mtcars")
docdb_create(src, key = "mtcars", value = mtcars)
docdb_exists(src, "mtcars")

# SQLite
(src <- src_sqlite())
docdb_create(src, "mtcars", mtcars)
docdb_exists(src, "mtcars")
docdb_exists(src, "yellowcheese")

## End(Not run)
```

docdb_get

Get documents

Description

Get documents

Usage

```
docdb_get(src, key, limit = NULL, ...)
```

Arguments

src	source object, result of call to src
key	(character) A key (collection for mongo)
limit	(integer) number of records/rows to return. by default not passed, so you get all results. Only works for CouchDB, Elasticsearch and MongoDB; ignored for others
...	passed on to functions: <ul style="list-style-type: none"> • CouchDB: passed to <code>sofa::db_alldocs()</code> • Elasticsearch: passed to <code>elastic::Search()</code> • Redis: ignored • MongoDB: ignored • SQLite: ignored

Details

Note that with etcd, you have to prefix a key with a forward slash.

Examples

```
## Not run:
# CouchDB
src <- src_couchdb()
docout <- docdb_create(src, key = "mtcars2", value = mtcars)
docdb_get(src, "mtcars2")
docdb_get(src, "mtcars2", limit = 5)

# Elasticsearch
src <- src_elastic()
if (docdb_exists(src, "iris")) docdb_delete(src, "iris")
docdb_create(src, "iris", iris)
Sys.sleep(2)
docdb_get(src, "iris")
if (docdb_exists(src, "d2")) docdb_delete(src, "d2")
docdb_create(src, "d2", diamonds)
Sys.sleep(3)
docdb_get(src, "d2", limit = 1010)

# Redis
src <- src_redis()
docdb_create(src, "mtcars", mtcars)
docdb_get(src, "mtcars")

# Mongo
src <- src_mongo(collection = "mtcars")
docdb_create(src, "mtcars", mtcars)
docdb_get(src, "mtcars")
docdb_get(src, "mtcars", limit = 4)

# SQLite
```



```

src <- src_sqlite()
docdb_create(src, "mtcars", mtcars)
docdb_get(src, "mtcars")
docdb_get(src, "mtcars", limit = 4L)

## End(Not run)

```

docdb_query

Get documents with a filtering query

Description

Get documents with a filtering query

Usage

```
docdb_query(src, key, query, ...)
```

Arguments

src	source object, result of call to src
key	(character) A key (collection for mongo)
query	various. see Query section below.
...	Additional named parameters passed on to each package: <ul style="list-style-type: none"> • CouchDB: passed on to <code>sofa::db_query()</code> • Elasticsearch: passed on to <code>elastic::Search()</code> • MongoDB: passed on to the <code>\$find</code> method of mongolite

Details

Note that with etcd, you have to prefix a key with a forward slash.

What is expected for each source

- CouchDB: a list, see docs for `sofa::db_query()`
- Elasticsearch: query parameters, see `elastic::Search()`; passed to the query parameter of `elastic::Search`, thus performs a URI based search where the query is passed in the URI instead of the body. In theory you can instead pass in a JSON or list to the body parameter, but if you want to do complicated Elasticsearch queries you may be better of using **elastic** package directly
- MongoDB: query parameters, see **mongolite** docs for help with searches
- SQLite: fields, an optional json string of fields to be returned from anywhere in the tree. Parameter query, a json string In analogy to MongoDB, a comma separated list of expressions provides an implicit AND operation. Nested or otherwise complex queries are not yet supported.

Not supported yet

- Redis

Examples

```
## Not run:
# CouchDB
(src <- src_couchdb())
if (docdb_exists(src, "mtcars2")) docdb_delete(src, "mtcars2")
invisible(docdb_create(src, key = "mtcars2", value = mtcars))
docdb_exists(src, "mtcars2")
(query <- list(cyl = list("$gt" = 6)))
docdb_query(src, "mtcars2", query = query)

# Elasticsearch
src <- src_elastic()
if (docdb_exists(src, "iris")) docdb_delete(src, "iris")
docdb_create(src, "iris", iris)
docdb_exists(src, "iris")
Sys.sleep(2)
docdb_query(src, "iris", query = "setosa")
docdb_query(src, "iris", query = "1.5")
docdb_query(src, "iris", query = "Petal.Width:1.5")

# Mongo
src <- src_mongo(collection = "mtcars")
if (docdb_exists(src, "mtcars")) docdb_delete(src, "mtcars")
docdb_create(src, "mtcars", mtcars)
docdb_query(src, "mtcars", query = '{"mpg":21}')
docdb_query(src, "mtcars", query = '{"mpg":21}', fields = '{"mpg":1, "cyl":1}')
docdb_get(src, "mtcars")

# SQLite
src <- src_sqlite()
docdb_create(src, "mtcars", mtcars)
docdb_query(src, "mtcars", query = "{}", fields = '{"mpg":1, "cyl":1}')
docdb_query(src, "mtcars", query = '{"gear": {"$lte": 4}}', fields = '{"gear": 1}')
# for RSQLite from 2.1.2 using PCRE regular expressions
docdb_query(src, "mtcars", query = '{"_id": {"$regex": "^.+0.*$"}}', fields = '{"gear": 1}')

## End(Not run)
```

docdb_update

Update documents

Description

Update documents

Usage

```
docdb_update(src, key, value, ...)
```

Arguments

src	source object, result of call to src
key	(character) A key (collection for mongo)
value	(data.frame) A single data.frame
...	Ignored

Details

Only CouchDB and sqlite supported for now

Examples

```
## Not run:
# CouchDB
src <- src_couchdb()
docdb_create(src, "mtcars2", mtcars)
docdb_get(src, "mtcars2")

mtcars$letter <- sample(letters, NROW(mtcars), replace = TRUE)
invisible(docdb_update(src, "mtcars2", mtcars))
docdb_get(src, "mtcars2")

# MongoDB
src <- src_mongo(collection = "mtcars")
docdb_create(src, key = "mtcars", value = mtcars)
# - update of carb for each matching gear
value <- data.frame("gear" = c(4, 5),
                    "carb" = c(8.1, 7.9),
                    stringsAsFactors = FALSE)
docdb_update(src, "mtcars", value)
# - update of gear where _id / oid is "2"
value <- data.frame("_id" = "2",
                    "gear" = 9,
                    stringsAsFactors = FALSE,
                    check.names = FALSE)
docdb_update(src, "mtcars", value)
docdb_get(src, "mtcars")

# SQLite
src <- src_sqlite()
docdb_create(src, "mtcars", mtcars)
dfupd <- data.frame("cyl" = c(4, 6),
                    "gear" = c(88, 99),
                    stringsAsFactors = FALSE)
docdb_update(src, "mtcars", dfupd)
docdb_query(src, "mtcars",
            query = '{"gear": {"$gte": 88}}',
```

```

        fields = '{"gear": 1, "cyl": 1}'
dfupd <- data.frame("cyl" = c(8, 6),
                  "somejson" = c('{"gear": 77, "carb": 55}',
                                '{"gear": 66, "newvar": 55}'),
                  stringsAsFactors = FALSE)
docdb_update(src, "mtcars", dfupd)
docdb_query(src, "mtcars",
            query = '{"gear": {"$eq": 66}}',
            fields = '{"gear": 1, "cyl": 1, "carb": 1, "newvar": 1}')

## End(Not run)

```

nodbi-defunct *Defunct functions in nodbi*

Description

- [src_etcd](#): etcd removed, as long as S3 methods for all docdb_* functions

src *Setup database connections*

Description

Setup database connections

Details

There is a `src_*`() function to setup a connection to each of the database backends. Each has their own unique set of parameters.

- MongoDB - [src_mongo\(\)](#)
- CouchDB - [src_couchdb\(\)](#)
- Elasticsearch - [src_elastic\(\)](#)
- Redis - [src_redis\(\)](#)
- SQLite - [src_sqlite\(\)](#)

Documentation details for each database:

- MongoDB - <https://docs.mongodb.com/>
- CouchDB - <http://docs.couchdb.org/>
- Elasticsearch - <https://www.elastic.co/guide/en/elasticsearch/reference/current>
- Redis - <https://redis.io/documentation>
- SQLite/json1 - <https://www.sqlite.org/json1.html>

src_couchdb	<i>Setup a CouchDB database connection</i>
-------------	--

Description

Setup a CouchDB database connection

Usage

```
src_couchdb(host = "127.0.0.1", port = 5984, path = NULL,  
            transport = "http", user = NULL, pwd = NULL, headers = NULL)
```

Arguments

host	(character) host value, default: 127.0.0.1
port	(integer/numeric) Port. Remember that if you don't want a port set, set this parameter to NULL. Default: 5984
path	(character) context path that is appended to the end of the url. e.g., bar in http://foo.com/bar. Default: NULL, ignored
transport	(character) http or https. Default: http
user	(character) Username, if any
pwd	(character) Password, if any
headers	(list) list of named headers

Details

uses **sofa** under the hood; uses [sofa::Cushion](#) for connecting

Examples

```
## Not run:  
src_couchdb()  
  
## End(Not run)
```

`src_elastic`*Setup an Elasticsearch database connection*

Description

Setup an Elasticsearch database connection

Usage

```
src_elastic(host = "127.0.0.1", port = 9200, path = NULL,  
            transport_schema = "http", user = NULL, pwd = NULL,  
            force = FALSE, ...)
```

Arguments

<code>host</code>	(character) the base url, defaults to localhost (<code>http://127.0.0.1</code>)
<code>port</code>	(character) port to connect to, defaults to 9200 (optional)
<code>path</code>	(character) context path that is appended to the end of the url. Default: NULL, ignored
<code>transport_schema</code>	(character) http or https. Default: http
<code>user</code>	(character) User name, if required for the connection. You can specify, but ignored for now.
<code>pwd</code>	(character) Password, if required for the connection. You can specify, but ignored for now.
<code>force</code>	(logical) Force re-load of connection details
<code>...</code>	Further args passed on to <code>elastic::connect()</code>

Details

uses **elastic** under the hood; uses `elastic::connect()` for connecting

Examples

```
## Not run:  
src_elastic()  
  
## End(Not run)
```

src_mongo *Setup a mongoDB database connection*

Description

Setup a mongoDB database connection

Usage

```
src_mongo(collection = "test", db = "test",  
           url = "mongodb://localhost", ...)
```

Arguments

collection	(character) name of collection
db	(character) name of database
url	(character) address of the mongodb server in mongo connection string URI format.
...	additional named params passed on to mongolite::mongo

Details

uses **monoglite** under the hood; uses `mongolite::mongo()` for connecting

Examples

```
## Not run:  
(con <- src_mongo())  
print(con)  
  
## End(Not run)
```

src_redis *Setup an Redis database connection*

Description

Setup an Redis database connection

Usage

```
src_redis(...)
```

Arguments

... Redis connection configuration options. See [redux::redis_config](#) for more information

Details

uses **redux** under the hood; uses [redux::hiredis\(\)](#) for connecting

Examples

```
## Not run:
(con <- src_redis())
class(con)

## End(Not run)
```

src_sqlite

Setup a sqlite database connection

Description

Setup a sqlite database connection

Usage

```
src_sqlite(dbname = ":memory:", ...)
```

Arguments

dbname (character) name of database file, defaults to ":memory:" for an in-memory database, see [RSQLite::SQLite\(\)](#)

... additional named parameters passed on to [RSQLite::SQLite\(\)](#)

Details

uses **RSQLite** under the hood

Examples

```
## Not run:
(con <- src_sqlite())
print(con)

## End(Not run)
```


Index

*Topic **data**

contacts, [3](#)

diamonds, [3](#)

*Topic **package**

nodbi-package, [2](#)

contacts, [3](#)

diamonds, [3](#)

docdb_create, [4](#)

docdb_delete, [5](#)

docdb_exists, [6](#)

docdb_get, [7](#)

docdb_query, [9](#)

docdb_update, [10](#)

elastic::connect(), [14](#)

elastic::Search(), [8](#), [9](#)

mongolite::mongo, [15](#)

mongolite::mongo(), [15](#)

nodbi (nodbi-package), [2](#)

nodbi-defunct, [12](#)

nodbi-package, [2](#)

redux::hiredis(), [16](#)

redux::redis_config, [16](#)

RSQLite::SQLite(), [16](#)

sofa::Cushion, [13](#)

sofa::db_alldocs(), [8](#)

sofa::db_query(), [9](#)

src, [4](#), [12](#)

src_couchdb, [13](#)

src_couchdb(), [12](#)

src_elastic, [14](#)

src_elastic(), [12](#)

src_etcd, [12](#)

src_mongo, [15](#)

src_mongo(), [12](#)

src_redis, [15](#)

src_redis(), [12](#)

src_sqlite, [16](#)

src_sqlite(), [12](#)