Package ‘duckdb’

June 16, 2023

Title DBI Package for the DuckDB Database Management System

Version 0.8.1

Description The DuckDB project is an embedded analytical data management system with support for the Structured Query Language (SQL). This package includes all of DuckDB and a R Database Interface (DBI) connector.

License MIT + file LICENSE


BugReports https://github.com/duckdb/duckdb/issues

Depends DBI, R (>= 3.6.0)

Imports methods, utils

Suggests arrow, bit64, callr, DBItest, dplyr, dbplyr, rlang, testthat, tibble, vctrs, withr

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation yes

Author Hannes Mühleisen [aut, cre] (https://orcid.org/0000-0001-8552-0029),
Mark Raasveldt [aut] (https://orcid.org/0000-0001-5005-6844),
Stichting DuckDB Foundation [cph],
Apache Software Foundation [cph],
PostgreSQL Global Development Group [cph],
The Regents of the University of California [cph],
Cameron Desrochers [cph],
Victor Zverovich [cph],
RAD Game Tools [cph],
Valve Software [cph],
Rich Geldreich [cph],
Tenacious Software LLC [cph],
The RE2 Authors [cph],
Google Inc. [cph],
Facebook Inc. [cph],
Steven G. Johnson [cph],

1
duckdb-package

Jiahao Chen [cph],
Tony Kelman [cph],
Jonas Fonseca [cph],
Lukas Fittl [cph],
Salvatore Sanfilippo [cph],
Art.sy, Inc. [cph],
Oran Agra [cph],
Redis Labs, Inc. [cph],
Melissa O'Neill [cph],
PCG Project contributors [cph]

Maintainer  Hannes Mühleisen <hannes@cwi.nl>
Repository  CRAN
Date/Publication  2023-06-16 11:30:02 UTC

R topics documented:

duckdb-package ................................................... 2
backend-duckdb .................................................. 3
duckdb .......................................................... 3
duckdb_explain-class .......................................... 5
duckdb_get_substrait .......................................... 5
duckdb_get_substrait_json .................................... 6
duckdb_prepare_substrait ..................................... 7
duckdb_prepare_substrait_json ................................ 7
duckdb_read_csv ............................................... 8
duckdb_register ................................................. 9
duckdb_register_arrow ....................................... 10

Index  11

duckdb-package  DuckDB client package for R

Description

R client package for DuckDB: an embeddable SQL OLAP Database Management System.

See Also

duckdb() for connection instructions.

https://duckdb.org/ for the project website.
**backend-duckdb**  
_DuckDB SQL backend for dbplyr_

**Description**

This is a SQL backend for dbplyr tailored to take into account DuckDB’s possibilities. This mainly follows the backend for PostgreSQL, but contains more mapped functions.

**Usage**

```
simulate_duckdb(...)  
translate_duckdb(...)  
```

**Arguments**

```
... Any parameters to be forwarded
```

**Examples**

```r
library(dplyr, warn.conflicts = FALSE)  
con <- DBI::dbConnect(duckdb(), path = "::memory::")  

dbiris <- copy_to(con, iris, overwrite = TRUE)  

dbiris %>%  
  select(Petal.Length, Petal.Width) %>%  
  filter(Petal.Length > 1.5) %>%  
  head(5)  

DBI::dbDisconnect(con, shutdown = TRUE)
```

**duckdb**  
_Connect to a DuckDB database instance_

**Description**

*duckdb()* creates or reuses a database instance.  
*duckdb_shutdown()* shuts down a database instance.  
*dbConnect()* connects to a database instance.  
*dbDisconnect()* closes a DuckDB database connection, optionally shutting down the associated instance.
Usage

duckdb(
    dbdir = DBDIR_MEMORY,
    read_only = FALSE,
    bigint = "numeric",
    config = list()
)

duckdb_shutdown(drv)

## S4 method for signature 'duckdb_driver'

dbConnect(
    drv,
    dbdir = DBDIR_MEMORY,
    ...
)

dbDisconnect(conn, ..., shutdown = FALSE)

Arguments

dbdir Location for database files. Should be a path to an existing directory in the file
system. With the default, all data is kept in RAM
read_only Set to TRUE for read-only operation
bigint How 64-bit integers should be returned, default is double/numeric. Set to integer64 for bit64 encoding.
config Named list with DuckDB configuration flags
drv Object returned by duckdb()
... Ignored
debug Print additional debug information such as queries
timezone_out The time zone returned to R, defaults to "UTC", which is currently the only
timezone supported by duckdb. If you want to display datetime values in the
local timezone, set to Sys.timezone() or "."
tz_out_convert How to convert timestamp columns to the timezone specified in timezone_out.
There are two options: "with", and "force". If "with" is chosen, the timestamp will be returned as it would appear in the specified time zone. If "force" is chosen, the timestamp will have the same clock time as the timestamp in the
database, but with the new time zone.
conn A duckdb_connection object
shutdown Set to TRUE to shut down the DuckDB database instance that this connection refers to.

Value
duckdb() returns an object of class duckdb_driver.
dbDisconnect() and duckdb_shutdown() are called for their side effect.
dbConnect() returns an object of class duckdb_connection.

Examples
drv <- duckdb()
con <- dbConnect(drv)

dbGetQuery(con, "SELECT 'Hello, world!'")

dbDisconnect(con)
duckdb_shutdown(drv)

# Shorter:
con <- dbConnect(duckdb())

dbGetQuery(con, "SELECT 'Hello, world!'")

dbDisconnect(con, shutdown = TRUE)
duckdb_get_substrait_json

Get the Substrait plan for a SQL query in the JSON format. Transforms a SQL query into a vector containing the serialized Substrait query JSON.

**Description**

Get the Substrait plan for a SQL query in the JSON format. Transforms a SQL query into a vector containing the serialized Substrait query JSON.

**Usage**

```r
duckdb_get_substrait_json(conn, query, enable_optimizer = TRUE)
```

**Arguments**

- **conn**: A DuckDB connection, created by `dbConnect()`.
- **query**: The query string in SQL.
- **enable_optimizer**: Optional parameter to enable/disable query-optimizer. By default optimizer is enabled.

**Value**

A vector containing the substrait protobuf JSON.
duckdb_prepare_substrait

Query DuckDB using Substrait Method for interpreting a Substrait BLOB plan as a DuckDB Query Plan. It interprets and executes the query.

Usage

duckdb_prepare_substrait(conn, query, arrow = FALSE)

Arguments

- **conn**: A DuckDB connection, created by dbConnect().
- **query**: The Protobuf-encoded Substrait Query Plan. Qack!
- **arrow**: Whether the result should be in Arrow format.

Value

A DuckDB Query Result.

duckdb_prepare_substrait_json

Query DuckDB using Substrait Method for interpreting a Substrait JSON plan as a DuckDB Query Plan. It interprets and executes the query.

Usage

duckdb_prepare_substrait_json(conn, json, arrow = FALSE)

Arguments

- **conn**: A DuckDB connection, created by dbConnect().
- **json**: The Json Query Plan. Qack!
- **arrow**: Whether the result should be in Arrow format.
duckdb_read_csv

Value

A DuckDB Query Result

duckdb_read_csv: Reads a CSV file into DuckDB

Description

Directly reads a CSV file into DuckDB, tries to detect and create the correct schema for it. This usually is much faster than reading the data into R and writing it to DuckDB.

Usage

duckdb_read_csv(
  conn,
  name,
  files,
  header = TRUE,
  na.strings = "",
  nrow.check = 500,
  delim = ",",
  quote = "\"",
  col.names = NULL,
  lower.case.names = FALSE,
  sep = delim,
  transaction = TRUE,
  ...
)

Arguments

conn: A DuckDB connection, created by dbConnect().
name: The name for the virtual table that is registered or unregistered.
files: One or more CSV file names, should all have the same structure though.
header: Whether or not the CSV files have a separate header in the first line.
na.strings: Which strings in the CSV files should be considered to be NULL.
nrow.check: How many rows should be read from the CSV file to figure out data types.
delim: Which field separator should be used.
quote: Which quote character is used for columns in the CSV file.
col.names: Override the detected or generated column names.
lower.case.names: Transform column names to lower case.
sep: Alias for delim for compatibility.
transaction: Should a transaction be used for the entire operation.
...: Passed on to read.csv()
**duckdb_register**

**Value**

The number of rows in the resulted table, invisibly.

**Examples**

```r
con <- dbConnect(duckdb())
data <- data.frame(a = 1:3, b = letters[1:3])path <- tempfile(fileext = "csv")
write.csv(data, path, row.names = FALSE)
duckdb_read_csv(con, "data", path)dbReadTable(con, "data")
dbDisconnect(con)
```

---

**duckdb_register**

*Register a data frame as a virtual table*

**Description**

`duckdb_register()` registers a data frame as a virtual table (view) in a DuckDB connection. No data is copied.

**Usage**

```r
duckdb_register(conn, name, df, overwrite = FALSE, experimental = FALSE)
duckdb_unregister(conn, name)
```

**Arguments**

- `conn`: A DuckDB connection, created by `dbConnect()`.
- `name`: The name for the virtual table that is registered or unregistered.
- `df`: A `data.frame` with the data for the virtual table.
- `overwrite`: Should an existing registration be overwritten?
- `experimental`: Enable experimental optimizations.

**Details**

`duckdb_unregister()` unregisters a previously registered data frame.

**Value**

These functions are called for their side effect.
Examples

```r
con <- dbConnect(duckdb())

data <- data.frame(a = 1:3, b = letters[1:3])

duckdb_register(con, "data", data)
dbReadTable(con, "data")

duckdb_unregister(con, "data")
dbDisconnect(con)
```

---

duckdb_register_arrow  Register an Arrow data source as a virtual table

Description

duckdb_register_arrow() registers an Arrow data source as a virtual table (view) in a DuckDB connection. No data is copied.

Usage

duckdb_register_arrow(conn, name, arrow_scannable, use_async = NULL)
duckdb_unregister_arrow(conn, name)
duckdb_list_arrow(conn)

Arguments

- `conn`: A DuckDB connection, created by `dbConnect()`.
- `name`: The name for the virtual table that is registered or unregistered.
- `arrow_scannable`: A scannable Arrow-object
- `use_async`: Switched to the asynchronous scanner. (deprecated)

Details

duckdb_unregister_arrow() unregisters a previously registered data frame.

Value

These functions are called for their side effect.
Index

backend-duckdb, 3

dbConnect, duckdb_driver-method (duckdb), 3
dbConnect__duckdb_driver (duckdb), 3
dbDisconnect, duckdb_connection-method (duckdb), 3
dbDisconnect__duckdb_connection (duckdb), 3
duckdb, 3
duckdb(), 2
duckdb-package, 2
duckdb_connection, 5
duckdb_driver, 5
duckdb_explain (duckdb_explain-class), 5
duckdb_explain-class, 5
duckdb_get_substruct, 5
duckdb_get_substruct_json, 6
duckdb_list_arrow (duckdb_register_arrow), 10
duckdb_prepare_substruct, 7
duckdb_prepare_substruct_json, 7
duckdb_read_csv, 8
duckdb_register, 9
duckdb_register_arrow, 10
duckdb_shutdown (duckdb), 3
duckdb_unregister (duckdb_register), 9
duckdb_unregister_arrow (duckdb_register_arrow), 10

print.duckdb_explain (duckdb_explain-class), 5

read.csv(), 8

simulate_duckdb (backend-duckdb), 3
Sys.timezone(), 4

translate_duckdb (backend-duckdb), 3