

Package ‘duckdb’

September 5, 2022

Title DBI Package for the DuckDB Database Management System

Version 0.5.0

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

URL <https://duckdb.org/>, <https://github.com/duckdb/duckdb>

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 7.2.1

SystemRequirements C++11, GCC on Solaris

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],
 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@cwil.nl>

Repository CRAN

Date/Publication 2022-09-05 18:20:06 UTC

R topics documented:

duckdb-package	3
backend-duckdb	3
duckdb	4
duckdb_explain-class	5
duckdb_get_substrait	6
duckdb_get_substrait_json	6
duckdb_prepare_substrait	7
duckdb_read_csv	7
duckdb_register	9
duckdb_register_arrow	10
expr_constant	10
expr_function	11
expr_reference	11
expr_set_alias	12
expr_tostring	12
rel_aggregate	13
rel_alias	13
rel_distinct	14
rel_explain	14
rel_filter	15
rel_from_df	15
rel_inner_join	16
rel_order	17
rel_project	17
rel_set_alias	18
rel_sql	18

Index

20

duckdb-package	<i>DuckDB client package for R</i>
----------------	------------------------------------

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	<i>DuckDB SQL backend for dbplyr</i>
----------------	--------------------------------------

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

```
library(dplyr, warn.conflicts = FALSE)
con <- DBI::dbConnect(duckdb::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,
  ...,
  debug = getOption("duckdb.debug", FALSE),
  read_only = FALSE,
  timezone_out = "UTC",
  tz_out_convert = c("with", "force"),
  config = list(),
  bigint = "numeric"
)
```

```
## S4 method for signature 'duckdb_connection'
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 <code>duckdb()</code>
...	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 <code>Sys.timezone()</code> or "".
tz_out_convert	How to convert timestamp columns to the timezone specified in <code>timezone_out</code> . 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 <code>duckdb_connection</code> 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_explain-class *DuckDB EXPLAIN query tree*

Description

DuckDB EXPLAIN query tree

duckdb_get_substrait *Get the Substrait plan for a SQL query Transforms a SQL query into a raw vector containing the serialized Substrait query blob*

Description

Get the Substrait plan for a SQL query Transforms a SQL query into a raw vector containing the serialized Substrait query blob

Usage

```
duckdb_get_substrait(conn, query)
```

Arguments

conn	A DuckDB connection, created by <code>dbConnect()</code> .
query	The query string in SQL

Value

A raw vector containing the substrait protobuf blob

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

```
duckdb_get_substrait_json(conn, query)
```

Arguments

conn	A DuckDB connection, created by <code>dbConnect()</code> .
query	The query string in SQL

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.

Description

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 <code>dbConnect()</code> .
query	The Protobuf-encoded Substrait Query Plan. Qack!
arrow	Whether the result should be in Arrow format

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

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

Value

The number of rows in the resulted table, invisibly.

Examples

```

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	<i>Register a data frame as a virtual table</i>
-----------------	---

Description

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

Usage

```
duckdb_register(conn, name, df)
```

```
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

Details

duckdb_unregister() unregisters a previously registered data frame.

Value

These functions are called for their side effect.

Examples

```
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")
try(dbReadTable(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.

expr_constant *Create a constant expression*

Description

Create a constant expression

Usage

```
expr_constant(val)
```

Arguments

val	the constant value
-----	--------------------

Value

a constant expression

Examples

```
const_int_expr <- expr_constant(42)
const_str_expr <- expr_constant("Hello, World")
```

expr_function *Create a function call expression*

Description

Create a function call expression

Usage

```
expr_function(name, args)
```

Arguments

name	the function name
args	the a list of expressions for the function arguments

Value

a function call expression

Examples

```
call_expr <- expr_function("ABS", list(expr_constant(-42)))
```

expr_reference *Create a column reference expression*

Description

Create a column reference expression

Usage

```
expr_reference(name, table = "")
```

Arguments

name	the column name to be referenced
table	the optional table name or a relation object to be referenced

Value

a column reference expression

Examples

```
col_ref_expr <- expr_reference("some_column_name")
col_ref_expr2 <- expr_reference("some_column_name", "some_table_name")
```

expr_set_alias	<i>Set the alias for an expression</i>
----------------	--

Description

Set the alias for an expression

Usage

```
expr_set_alias(expr, alias)
```

Arguments

expr	the expression
alias	the alias

Examples

```
expr_set_alias(expr_constant(42), "my_alias")
```

expr_tostring	<i>Convert an expression to a string for debugging purposes</i>
---------------	---

Description

Convert an expression to a string for debugging purposes

Usage

```
expr_tostring(expr)
```

Arguments

expr	the expression
------	----------------

Value

a string representation of the expression

Examples

```
expr_str <- expr_tostring(expr_constant(42))
```

rel_aggregate	<i>Lazily aggregate a DuckDB relation object</i>
---------------	--

Description

Lazily aggregate a DuckDB relation object

Usage

```
rel_aggregate(rel, groups, aggregates)
```

Arguments

rel	the DuckDB relation object
groups	a list of DuckDB expressions to group by
aggregates	a (optionally named) list of DuckDB expressions with aggregates to compute

Value

the now aggregated duckdb_relation object

Examples

```
con <- DBI::dbConnect(duckdb())
rel <- rel_from_df(con, mtcars)
aggrs <- list(avg_hp = expr_function("avg", list(expr_reference("hp"))))
rel2 <- rel_aggregate(rel, list(expr_reference("cyl")), aggrs)
```

rel_alias	<i>Get the internal alias for a DuckDB relation object</i>
-----------	--

Description

Get the internal alias for a DuckDB relation object

Usage

```
rel_alias(rel)
```

Arguments

rel	the DuckDB relation object
-----	----------------------------

Examples

```
con <- DBI::dbConnect(duckdb())
rel <- rel_from_df(con, mtcars)
rel_alias(rel)
```

rel_distinct	<i>Lazily compute a distinct result on a DuckDB relation object</i>
--------------	---

Description

Lazily compute a distinct result on a DuckDB relation object

Usage

```
rel_distinct(rel)
```

Arguments

rel the input DuckDB relation object

Value

a new duckdb_relation object with distinct rows

Examples

```
con <- DBI::dbConnect(duckdb())
rel <- rel_from_df(con, mtcars)
rel2 <- rel_distinct(rel)
```

rel_explain	<i>Print the EXPLAIN output for a DuckDB relation object</i>
-------------	--

Description

Print the EXPLAIN output for a DuckDB relation object

Usage

```
rel_explain(rel)
```

Arguments

rel the DuckDB relation object

Examples

```
con <- DBI::dbConnect(duckdb())
rel <- rel_from_df(con, mtcars)
rel_explain(rel)
```

rel_filter	<i>Lazily filter a DuckDB relation object</i>
------------	---

Description

Lazily filter a DuckDB relation object

Usage

```
rel_filter(rel, exprs)
```

Arguments

rel	the DuckDB relation object
exprs	a list of DuckDB expressions to filter by

Value

the now filtered duckdb_relation object

Examples

```
con <- DBI::dbConnect(duckdb())
DBI::dbExecute(con, 'CREATE MACRO gt(a, b) AS a > b')
rel <- rel_from_df(con, mtcars)
rel2 <- rel_filter(rel, list(expr_function("gt", list(expr_reference("cyl"), expr_constant("6")))))
```

rel_from_df	<i>Convert a R data.frame to a DuckDB relation object</i>
-------------	---

Description

Convert a R data.frame to a DuckDB relation object

Usage

```
rel_from_df(con, df)
```

Arguments

con	a DuckDB DBI connection object
df	the data.frame

Value

the duckdb_relation object wrapping the data.frame

Examples

```
con <- DBI::dbConnect(duckdb::duckdb())
rel <- rel_from_df(con, mtcars)
```

rel_inner_join	<i>Lazily INNER join two DuckDB relation objects</i>
----------------	--

Description

Lazily INNER join two DuckDB relation objects

Usage

```
rel_inner_join(left, right, conds)
```

Arguments

left	the left-hand-side DuckDB relation object
right	the right-hand-side DuckDB relation object
conds	a list of DuckDB expressions to use for the join

Value

a new duckdb_relation object resulting from the join

Examples

```
con <- DBI::dbConnect(duckdb())
DBI::dbExecute(con, 'CREATE MACRO eq(a, b) AS a = b')
left <- rel_from_df(con, mtcars)
right <- rel_from_df(con, mtcars)
cond <- list(expr_function("eq", list(expr_reference("cyl", left), expr_reference("cyl", right))))
rel2 <- rel_inner_join(left, right, cond)
```

rel_order	<i>Lazily reorder a DuckDB relation object</i>
-----------	--

Description

Lazily reorder a DuckDB relation object

Usage

```
rel_order(rel, orders)
```

Arguments

rel	the DuckDB relation object
orders	a list of DuckDB expressions to order by

Value

the now aggregated duckdb_relation object

Examples

```
con <- DBI::dbConnect(duckdb())
rel <- rel_from_df(con, mtcars)
rel2 <- rel_order(rel, list(expr_reference("hp")))
```

rel_project	<i>Lazily project a DuckDB relation object</i>
-------------	--

Description

Lazily project a DuckDB relation object

Usage

```
rel_project(rel, exprs)
```

Arguments

rel	the DuckDB relation object
exprs	a list of DuckDB expressions to project

Value

the now projected duckdb_relation object

Examples

```
con <- DBI::dbConnect(duckdb())
rel <- rel_from_df(con, mtcars)
rel2 <- rel_project(rel, list(expr_reference("cyl"), expr_reference("disp")))
```

rel_set_alias	<i>Set the internal alias for a DuckDB relation object</i>
---------------	--

Description

Set the internal alias for a DuckDB relation object

Usage

```
rel_set_alias(rel, alias)
```

Arguments

rel	the DuckDB relation object
alias	the new alias

Examples

```
con <- DBI::dbConnect(duckdb())
rel <- rel_from_df(con, mtcars)
rel_set_alias(rel, "my_new_alias")
```

rel_sql	<i>Run a SQL query on a DuckDB relation object</i>
---------	--

Description

Run a SQL query on a DuckDB relation object

Usage

```
rel_sql(rel, sql)
```

Arguments

rel	the DuckDB relation object
sql	a SQL query to run, use <code>_</code> to refer back to the relation

Value

the now aggregated duckdb_relation object

Examples

```
con <- DBI::dbConnect(duckdb())
rel <- rel_from_df(con, mtcars)
rel2 <- rel_sql(rel, "SELECT hp, cyl FROM _ WHERE hp > 100")
```

Index

backend-duckdb, 3

dbConnect, duckdb_driver-method
(duckdb), 4

dbConnect__duckdb_driver (duckdb), 4

dbDisconnect, duckdb_connection-method
(duckdb), 4

dbDisconnect__duckdb_connection
(duckdb), 4

duckdb, 4

duckdb(), 3

duckdb-package, 3

duckdb_connection, 5

duckdb_driver, 5

duckdb_explain (duckdb_explain-class), 5

duckdb_explain-class, 5

duckdb_get_substrait, 6

duckdb_get_substrait_json, 6

duckdb_list_arrow
(duckdb_register_arrow), 10

duckdb_prepare_substrait, 7

duckdb_read_csv, 7

duckdb_register, 9

duckdb_register_arrow, 10

duckdb_shutdown (duckdb), 4

duckdb_unregister (duckdb_register), 9

duckdb_unregister_arrow
(duckdb_register_arrow), 10

expr_constant, 10

expr_function, 11

expr_reference, 11

expr_set_alias, 12

expr_tostring, 12

print.duckdb_explain
(duckdb_explain-class), 5

read.csv(), 8

rel_aggregate, 13

rel_alias, 13

rel_distinct, 14

rel_explain, 14

rel_filter, 15

rel_from_df, 15

rel_inner_join, 16

rel_order, 17

rel_project, 17

rel_set_alias, 18

rel_sql, 18

simulate_duckdb (backend-duckdb), 3

Sys.timezone(), 5

translate_duckdb (backend-duckdb), 3