Package: glaredb (via r-universe)

November 17, 2024

Title R Bindings for 'GlareDB'

Version 0.0.4

Description R bindings for 'GlareDB', an analytical database based on 'Apache Arrow' and 'Apache DataFusion' that can connect to various data sources and execute queries.

License AGPL-3

URL https://eitsupi.github.io/r-glaredb/, https://github.com/eitsupi/r-glaredb

BugReports https://github.com/eitsupi/r-glaredb/issues

Imports nanoarrow

Suggests arrow, knitr, patrick, polars, rlang, rmarkdown, testthat (>= 3.0.0)

Config/glaredb/LibVersion 0.0.3

Config/Needs/dev RcppTOML, devtools, desc, dplyr, fs, gert, glue, lintr, readr, styler, stringr

Config/Needs/website pkgdown

Config/testthat/edition 3

Encoding UTF-8

OS_type unix

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SystemRequirements Cargo (Rust's package manager), rustc, protobuf-compiler

Config/pak/sysreqs protobuf-compiler libprotoc-dev

Repository https://r-multiverse.r-universe.dev

RemoteUrl https://github.com/eitsupi/r-glaredb

RemoteRef v0.0.4

RemoteSha 349c5bb3a1fca6a8cefae381094b046c8ce028b2

Contents

`as_glaredb_table` *Create a GlareDB table*

Description

GlareDB table is a class that has a struct similar to arrow::Table innerly and can be converted from/to other classes via `nanoarrow::as_nanoarrow_array_stream()`.

Usage

```
as_glaredb_table(x, ...)

## Default S3 method:
as_glaredb_table(x, ..., schema = NULL)

## S3 method for class 'nanoarrow_array_stream'
as_glaredb_table(x, ...)

## S3 method for class 'RGlareDbExecutionOutput'
as_glaredb_table(x, ...)
```

Arguments

`x`	An object to be coerced to a GlareDB table.
`...`	Additional arguments passed to methods.
`schema`	An optional schema used to enforce conversion to a particular type. Defaults to `infer_nanoarrow_schema()`.

Details

The default method of `as_glaredb_table()` calls `nanoarrow::as_nanoarrow_array_stream()` internally, and all arguments are passed to it.

Value

A GlareDB table.

Examples

```
con <- glaredb_connect()

# Create a GlareDB table from a data frame with a specified schema
dat <- data.frame(a = 1:3, b = letters[1:3]) |>
  as_glaredb_table(
    schema = nanoarrow::na_struct(
      list(
        a = nanoarrow::na_int64(),
        b = nanoarrow::na_large_string()
      )
    )
  )

# Run an SQL query against the connection,
# and convert the result to a GlareDB table
glaredb_sql("SELECT * FROM dat", con) |>
  as_glaredb_table()

# Convert the GlareDB table to an R data frame
dat |>
  as.data.frame()

# Convert the GlareDB table to an arrow Table
if (requireNamespace("arrow", quietly = TRUE)) {
  dat |>
    arrow::as_arrow_table()
}

# Convert the GlareDB table to a polars DataFrame
if (requireNamespace("polars", quietly = TRUE)) {
  dat |>
    polars::as_polars_df()
}
```

`glaredb_connect` *Connect to a GlareDB database*

Description

Connect to a GlareDB database

Usage

```
glaredb_connect(
  data_dir_or_cloud_url = NULL,
  ...,
  spill_path = NULL,
  disable_tls = FALSE,
```

```
  cloud_addr = "https://console.glaredb.com",
  location = NULL,
  storage_options = NULL,
  env = parent.frame()
)
```

Arguments

data_dir_or_cloud_url	A character of path to a local GlareDB database or a cloud URL or NULL. If NULL, a in-memory database is used.
...	Ignored.
spill_path	TODO
disable_tls	TRUE or FALSE to indicating whether to disable TLS.
cloud_addr	A character of a GlareDB cloud URL.
location	TODO
storage_options	Named character vector of storage options or NULL (default).
env	The connected environment, an environment class or NULL (means the global env). GlareDB can resister some class of R objects inside the environment automatically, so you can access the objects inside this environment by the object name in the query. The default, the caller environment is used.

Value

GlareDB connection object

Examples

```
# Create a connection of in-memory database
con <- glaredb_connect()

# The print result shows the connected environment
con

# The connected environment can be accessed by `$.env`
con$.env

# Create a table to the database and insert data
glaredb_execute("CREATE TABLE my_table (a int)", con)
glaredb_execute("INSERT INTO my_table VALUES (1), (2)", con)

# Query the data and assign the result to a variable
res <- glaredb_sql("SELECT * FROM my_table", con)

# Since the result `res` exists in the connected environment,
# it can be resolved by the object name in the query
exists("res", envir = con$.env)
```

```
glaredb_sql("SELECT * FROM res", con) |>
  as_glaredb_table()
```

`glaredb_sql` *Run a query against a GlareDB database*

Description

Run a query against a GlareDB database

Usage

```
glaredb_sql(query, connection = NULL)

glaredb_prql(query, connection = NULL)

glaredb_execute(query, connection = NULL)
```

Arguments

`query`	A character of the query to run. • For `glaredb_sql()` and `glaredb_execute()`, an SQL query. • For `glaredb_prql()`, a PRQL query.
`connection`	A GlareDB connection object or `NULL`. If `NULL`, the default in-memory database is used.

Value

A GlareDB execusion output (Query plan). For `glaredb_execute()`, the value is returned invisibly.

Examples

```
# You can materialize the query result by `as_glaredb_table()` etc.
glaredb_sql("SELECT 'hello from R' as hello") |>
  as_glaredb_table()

glaredb_prql("from [
  {a=5, b=false},
  {a=6, b=true},
]") |>
  as.data.frame()

# `glaredb_execute()` is useful for manipulating the database
glaredb_execute("CREATE TABLE my_table (a int)")
glaredb_execute("INSERT INTO my_table VALUES (1), (2)")

glaredb_sql("SELECT * FROM my_table") |>
  as_glaredb_table()
```

Index