Package: db2pq (via r-universe)

June 3, 2026

Type Package

Title Export Database Tables to 'Parquet'

Version 0.0.4

Description Tools for exporting 'PostgreSQL' tables to 'Parquet' files, with support for chunked writes, column type overrides, and timezone-aware timestamp handling. Includes functions for maintaining a local 'Parquet' data library sourced from 'WRDS' (Wharton Research Data Services), with update-checking based on table metadata, and archive management utilities for versioning local data files. See Gow and Ding (2024) ``Empirical Research in Accounting: Tools and Methods'' <doi:10.1201/9781003456230>.

URL https://github.com/iangow/db2pqr, https://iangow.github.io/db2pqr/

BugReports https://github.com/iangow/db2pqr/issues

Author Ian D. Gow [aut, cre]

Maintainer Ian D. Gow <iandgow@gmail.com>

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports arrow, DBI, keyring, RPostgres, tibble, tools, wrds

Suggests adbcpostgresql, adbcdrivermanager, adbi, duckdb, dplyr, dbplyr, ggplot2, knitr, nanoarrow, processx, quarto, rstudioapi, testthat (>= 3.0.0)

Config/testthat/edition 3

Config/Needs/website pkgdown, quarto

Config/roxygen2/version 8.0.0

Config/pak/sysreqs cmake libicu-dev libsecret-1-dev libssl-dev libpq-dev

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

Date/Publication 2026-06-02 21:29:41 UTC

RemoteUrl https://github.com/iangow/db2pqr

RemoteRef HEAD

RemoteSha f2591ebabf031e17c8a90c27197c3c97263b2191

Contents

`adbc_diagnostics` *Report optional ADBC dependency status*

Description

Report optional ADBC dependency status

Usage

```
adbc_diagnostics()
```

Value

A tibble with installed status and package versions for optional ADBC packages used by `db2pq`.

Examples

```
adbc_diagnostics()
```

`con_to_adbi` *Create an ADBC PostgreSQL connection from an* `RPostgres` *connection*

Description

This helper is intentionally narrow: it only supports `RPostgres` `PqConnection` objects and assumes enough connection information is available to build a PostgreSQL URI. If any required piece is missing, the function fails instead of attempting alternative fallbacks. The returned ADBC connection is separate from the original `RPostgres` connection, so using this helper requires capacity for an additional concurrent database connection.

Usage

```
con_to_adbi(con)
```

Arguments

`con`	An `RPostgres` connection object.

Value

An ADBC connection created by `adbi`.

Examples

```
## Not run:
# Requires a PostgreSQL connection and the adbi and adbcpostgresql packages
pg_con <- DBI::dbConnect(RPostgres::Postgres())
adbi_con <- con_to_adbi(pg_con)
DBI::dbDisconnect(adbi_con)
DBI::dbDisconnect(pg_con)

## End(Not run)
```

db_schema_tables *List PostgreSQL relations in a schema*

Description

List PostgreSQL relations in a schema

Usage

```
db_schema_tables(
  schema,
  views = FALSE,
  con = NULL,
  host = Sys.getenv("PGHOST", "localhost"),
  database = Sys.getenv("PGDATABASE", unset = Sys.getenv("PGUSER")),
  user = Sys.getenv("PGUSER", Sys.info()[["user"]]),
  password = Sys.getenv("PGPASSWORD", ""),
  port = as.integer(Sys.getenv("PGPORT", 5432))
)
```

Arguments

schema	PostgreSQL schema name.
views	If TRUE, include views as well as base tables.
con	Optional existing DBI connection.
host, database, user, password, port	PostgreSQL connection parameters used when con is not supplied.

Value

A character vector of relation names.

Examples

```
## Not run:
# Requires a PostgreSQL connection with the target schema
con <- DBI::dbConnect(RPostgres::Postgres())
db_schema_tables("crsp", con = con)
DBI::dbDisconnect(con)

## End(Not run)
```

`db_schema_to_pq` *Export all tables in a PostgreSQL schema to Parquet*

Description

Export all tables in a PostgreSQL schema to Parquet

Usage

```
db_schema_to_pq(
  schema,
  data_dir = NULL,
  force = FALSE,
  tables = NULL,
  chunk_size = 100000L,
  transfer_method = c("dbi", "adbc"),
  numeric_mode = c("decimal", "float64", "text", "raw"),
  archive = FALSE,
  archive_dir = "archive",
  con = NULL,
  ...
)
```

Arguments

`schema`	PostgreSQL schema name.
`data_dir`	Root directory of the local Parquet data repository.
`force`	If `FALSE`, existing output files are skipped.
`tables`	Optional subset of table names to export.
`chunk_size`	Number of rows fetched and written per chunk.
`transfer_method`	Transfer backend passed to `db_to_pq`.
`numeric_mode`	Numeric handling mode passed to `db_to_pq`.
`archive`	Whether existing output files should be archived before replacement.
`archive_dir`	Archive directory name.
`con`	Optional existing DBI connection.
`...`	Additional arguments passed to `db_to_pq`.

Value

Invisibly returns a named list of output paths.

Examples

```
## Not run:
# Requires a PostgreSQL connection with the target schema
db_schema_to_pq("crsp", data_dir = "~/pq_data")

## End(Not run)
```

---

db_to_pq *Export a PostgreSQL table to Parquet*

Description

Exports a table from PostgreSQL to a Parquet file, with optional row filtering, column filtering, column renaming, Arrow type overrides, and timestamp normalization.

Usage

```
db_to_pq(
  table_name,
  schema,
  host = Sys.getenv("PGHOST", "localhost"),
  database = Sys.getenv("PGDATABASE", unset = Sys.getenv("PGUSER")),
  user = Sys.getenv("PGUSER", Sys.info()[["user"]]),
  password = Sys.getenv("PGPASSWORD", ""),
  port = as.integer(Sys.getenv("PGPORT", 5432)),
  data_dir = NULL,
  out_file = NULL,
  where = NULL,
  obs = NULL,
  keep = NULL,
  drop = NULL,
  rename = NULL,
  alt_table_name = NULL,
  chunk_size = 100000L,
  transfer_method = c("dbi", "adbc"),
  numeric_mode = c("decimal", "float64", "text", "raw"),
  con = NULL,
  metadata = NULL,
  use_comment = TRUE,
  col_types = NULL,
  tz = NULL
)
```

Arguments

table_name	Name of the source PostgreSQL table.
schema	Name of the source PostgreSQL schema.

host, database, user, password, port
: PostgreSQL connection parameters used when con is not supplied.

data_dir
: Root directory of the local Parquet data repository. Defaults to the DATA_DIR environment variable, with interactive setup when needed.

out_file
: Optional full output path. Overrides data_dir, schema, and table_name.

where
: Optional SQL WHERE clause without the WHERE keyword.

obs
: Optional integer row limit.

keep, drop
: Optional character vectors of regex patterns used to select source columns. drop is applied before keep.

rename
: Optional named character vector or list mapping source column names to output column names, e.g. c(conm = "company_name").

alt_table_name
: Optional output basename when out_file is omitted.

chunk_size
: Number of rows fetched and written per chunk.

transfer_method
: Transfer backend: "dbi" for the stable DBI path or "adbc" for the optional Arrow/ADBC path.

numeric_mode
: Numeric handling mode for PostgreSQL numeric columns. The default "decimal" transfers them as text and writes bounded numeric(p, s) columns as Arrow decimal Parquet columns. "float64" casts them to DOUBLE PRECISION before transfer. "text" transfers them as text without recreating decimal types. "raw" keeps the backend default.

con
: Optional existing DBI connection.

metadata
: Optional named list of Parquet schema metadata.

use_comment
: If TRUE (the default), the PostgreSQL table comment is fetched and stored as last_modified in the Parquet schema metadata. Has no effect when the table has no comment. A last_modified key already present in metadata takes precedence.

col_types
: Optional named list of Arrow output type overrides. Names refer to output column names after rename.

tz
: Time zone used to interpret TIMESTAMP WITHOUT TIME ZONE columns.

Value

Invisibly returns the output file path.

Examples

```
## Not run:
# Requires a PostgreSQL connection with the target schema and table
db_to_pq("dsi", "crsp", data_dir = "~/pq_data")

# Limit rows for a quick test
db_to_pq("dsf", "crsp", data_dir = "~/pq_data",
          obs = 1000, keep = c("permno", "date", "ret"))

## End(Not run)
```

`lazy_tbl_to_pq` *Export a lazy* `dbplyr` *table to Parquet*

Description

Renders a lazy `dbplyr` query to SQL and streams the result to a Parquet file using the package's internal SQL-to-Parquet writer. This avoids collecting the full result into memory before writing.

Usage

```
lazy_tbl_to_pq(
  tbl,
  out_file,
  chunk_size = 100000L,
  metadata = NULL,
  col_types = NULL
)
```

Arguments

`tbl`	A lazy table backed by `dbplyr`, such as the result of `dplyr::tbl()` or a pipeline of `dplyr` verbs on a remote table.
`out_file`	Full path to the output Parquet file.
`chunk_size`	Number of rows fetched and written per chunk. Default is `100000`.
`metadata`	Optional named list of schema metadata to embed in the Parquet file.
`col_types`	Optional named list specifying Arrow type overrides. Values may be string type names (for example `"int32"` or `"date"`) or Arrow `DataType` objects.

Value

Invisibly returns `out_file`.

Examples

```
## Not run:
# Requires a PostgreSQL connection with the target schema and table
con <- DBI::dbConnect(RPostgres::Postgres())
qry <- dplyr::tbl(con, DBI::Id(schema = "crsp", table = "dsi")) |>
  dplyr::filter(date >= as.Date("2020-01-01"))

lazy_tbl_to_pq(qry, "~/pq_data/crsp/dsi_recent.parquet")
DBI::dbDisconnect(con)

## End(Not run)
```

pq_archive *Archive a Parquet file into the archive subdirectory*

Description

Moves a Parquet file into an archive subdirectory, appending a timestamp suffix derived from the file's `last_modified` metadata. The archived filename takes the form `<table>_<YYYYMMDDTHHMMSSz>.parquet`.

Usage

```
pq_archive(
  table_name = NULL,
  schema = NULL,
  data_dir = NULL,
  file_name = NULL,
  archive_dir = "archive"
)
```

Arguments

`table_name`	Name of the table. Required if `file_name` is not provided.
`schema`	Schema name. Required if `file_name` is not provided.
`data_dir`	Root directory of the Parquet data repository. Defaults to the `DATA_DIR` environment variable, with interactive setup when needed.
`file_name`	Full path to the Parquet file to archive. If provided, `table_name`, `schema`, and `data_dir` are ignored.
`archive_dir`	Name of the archive subdirectory relative to the schema directory. Defaults to `"archive"`.

Details

Either `file_name` or both `table_name` and `schema` must be provided.

Value

Invisibly returns the path to the archived file, or `NULL` if the source file does not exist.

See Also

`pq_restore`, `wrds_update_pq`

Examples

```
## Not run:
# These examples require an existing local Parquet data repository
pq_archive("company", "comp")
pq_archive(file_name = "~/pq_data/comp/company.parquet")

## End(Not run)
```

`pq_data_dir` *Resolve the Parquet data repository directory*

Description

Resolves the root directory used for Parquet data from an explicit argument or the `DATA_DIR` environment variable. In an interactive session, when neither is available, the helper asks the user to choose or enter a directory and offers to persist it in either project-level `.Renviron` or user-level `~/.Renviron`.

Usage

```
pq_data_dir(data_dir = NULL, prompt = interactive(), fallback = ".")
```

Arguments

`data_dir`	Optional Parquet data repository root.
`prompt`	If `TRUE`, prompt interactively when `data_dir` and `DATA_DIR` are both missing.
`fallback`	Directory returned when prompting is disabled and no explicit or environment value is available.

Value

A directory path as a character string.

Examples

```
# Resolve an explicit path without prompting
pq_data_dir("~/pq_data")
```

pq_last_modified *Get last-modified metadata for Parquet data files*

Description

Retrieves the last_modified metadata embedded in Parquet files by wrds_update_pq.

Usage

```
pq_last_modified(
  table_name = NULL,
  schema = NULL,
  data_dir = NULL,
  archive = FALSE,
  archive_dir = "archive"
)
```

Arguments

table_name	Optional. Name of a specific table.
schema	Optional. Name of the schema (subdirectory under data_dir).
data_dir	Root directory of the Parquet data repository. Defaults to the DATA_DIR environment variable, with interactive setup when needed.
archive	If TRUE, look in the archive subdirectory instead of the main schema directory.
archive_dir	Name of the archive subdirectory. Defaults to "archive".

Details

Behaviour depends on the arguments supplied:

- table_name **provided,** archive = FALSE (default): Returns the raw last_modified string embedded in <data_dir>/<schema>/<table_name>.parquet, or "" if the file has no such metadata.
- table_name **provided,** archive = TRUE: Returns a data frame of all archived files matching table_name in <data_dir>/<schema>/<archive_dir>/.
- **Only** schema **provided (no** table_name**)**: Returns a data frame summarising all Parquet files in the schema directory (or its archive subdirectory if archive = TRUE).
- **Neither** table_name **nor** schema **provided**: Returns a data frame summarising all Parquet files across every schema subdirectory of data_dir.

Value

When table_name is provided and archive = FALSE, a single string. Otherwise a tibble with columns file_name, table, schema, last_mod (a POSIXct in UTC), and last_mod_str.

See Also

wrds_update_pq

Examples

```
## Not run:
# These examples require an existing local Parquet data repository
# Raw metadata string for a single table
pq_last_modified("dsi", "crsp")

# Summary of archived versions of a table
pq_last_modified("company", "comp", archive = TRUE)

# Summary of all tables in a schema
pq_last_modified(schema = "crsp")

# Summary of all tables across all schemas
pq_last_modified(data_dir = "data")

## End(Not run)
```

---

pq_remove *Remove a Parquet file from active or archive storage*

Description

Deletes a Parquet file. By default the active file (in the schema directory) is removed. Set `archive = TRUE` to remove a file from the archive subdirectory instead.

Usage

```
pq_remove(
  table_name = NULL,
  schema = NULL,
  data_dir = NULL,
  file_name = NULL,
  archive = FALSE,
  archive_dir = "archive"
)
```

Arguments

`table_name`	Name of the table. Required if `file_name` is not provided.
`schema`	Schema name. Required if `file_name` is not provided.
`data_dir`	Root directory of the Parquet data repository. Defaults to the `DATA_DIR` environment variable, with interactive setup when needed.

file_name	Full path to the Parquet file to remove. If provided, table_name, schema, data_dir, and archive are ignored.
archive	If TRUE, resolve the file from the archive subdirectory rather than the active schema directory.
archive_dir	Name of the archive subdirectory relative to the schema directory. Defaults to "archive".

Details

Either file_name or both table_name and schema must be provided.

Value

Invisibly returns the path of the removed file, or NULL if the file was not found or could not be deleted.

See Also

pq_archive, pq_restore

Examples

```
## Not run:
# These examples require an existing local Parquet data repository
pq_remove("company", "comp")
pq_remove("company_20251109T072042Z", "comp", archive = TRUE)
pq_remove(file_name = "~/pq_data/comp/company.parquet")

## End(Not run)
```

---

pq_restore *Restore an archived Parquet file to the active schema directory*

Description

Moves an archived Parquet file back into the schema directory, stripping the timestamp suffix to recover the original filename. If a current file already exists at the destination, it is archived first (when archive = TRUE) or the function stops (when archive = FALSE).

Usage

```
pq_restore(
  file_basename,
  schema,
  data_dir = NULL,
  archive = TRUE,
  archive_dir = "archive"
)
```

Arguments

file_basename	Basename of the archived file (with or without the .parquet extension), e.g. "company_20251109T072042Z".
schema	Schema name.
data_dir	Root directory of the Parquet data repository. Defaults to the DATA_DIR environment variable, with interactive setup when needed.
archive	If TRUE (default), any existing file at the destination is archived before the restore. If FALSE, the function stops if the destination already exists.
archive_dir	Name of the archive subdirectory. Defaults to "archive".

Value

Invisibly returns the path to the restored file, or NULL on failure.

See Also

pq_archive, pq_last_modified

Examples

```
## Not run:
# Requires an existing archived file in a local Parquet data repository
pq_restore("company_20251109T072042Z", "comp")

## End(Not run)
```

---

tbl_to_pq *Export a lazy* dbplyr *table to Parquet via ADBC*

Description

Renders a lazy dbplyr query to SQL, derives an ADBC PostgreSQL connection from the backing RPostgres connection, and streams Arrow batches directly to a Parquet file.

Usage

```
tbl_to_pq(
  tbl,
  out_file,
  chunk_size = 100000L,
  metadata = NULL,
  col_types = NULL
)
```

Arguments

tbl	A lazy table backed by dbplyr.
out_file	Full path to the output Parquet file.
chunk_size	Number of rows written per Parquet row group. Default is 100000.
metadata	Optional named list of schema metadata to embed in the Parquet file.
col_types	Optional named list specifying Arrow type overrides. Values may be string type names (for example "int32" or "date") or Arrow DataType objects.

Details

This is an experimental development path kept separate from lazy_tbl_to_pq while we evaluate the ADBC transfer route. The ADBC path opens a second PostgreSQL connection for the transfer, so the database must allow an additional concurrent connection beyond the one already backing tbl.

Value

Invisibly returns out_file.

Examples

```
## Not run:
# Requires a PostgreSQL connection with the ADBC driver installed
con <- DBI::dbConnect(RPostgres::Postgres())
qry <- dplyr::tbl(con, DBI::Id(schema = "crsp", table = "dsi"))
tbl_to_pq(qry, "~/pq_data/crsp/dsi.parquet")
DBI::dbDisconnect(con)

## End(Not run)
```

---

tbl_to_pq_debug *Debug the ADBC chunk fetch path for a lazy* dbplyr *table*

Description

Fetches Arrow chunks one at a time using the experimental ADBC path and records where failures occur. This is intended for diagnosing driver or result-set issues without writing a Parquet file.

Usage

```
tbl_to_pq_debug(tbl, max_chunks = Inf)
```

Arguments

tbl	A lazy table backed by dbplyr.
max_chunks	Maximum number of chunks to inspect. Default is Inf.

Value

A data frame with one row per attempted chunk.

Examples

```
## Not run:
# Requires a PostgreSQL connection and the ADBC driver installed
con <- DBI::dbConnect(RPostgres::Postgres())
qry <- dplyr::tbl(con, DBI::Id(schema = "crsp", table = "dsi"))
tbl_to_pq_debug(qry, max_chunks = 3L)
DBI::dbDisconnect(con)

## End(Not run)
```

---

wrds_check_credentials

Check WRDS PostgreSQL credentials

Description

Attempts a small WRDS PostgreSQL connection using `RPostgres`. The return value is a data frame so it can be printed or inspected in setup scripts.

Usage

```
wrds_check_credentials(
  wrds_id = NULL,
  password = NULL,
  prompt = interactive(),
  save = TRUE,
  passfile = NULL
)
```

Arguments

`wrds_id`	Optional WRDS username override.
`password`	Optional password. If omitted, a matching `.pgpass` entry is preferred. When `.pgpass` has no matching entry, an existing `WRDS_PASSWORD` environment variable is used before an interactive prompt. PostgreSQL may also use `PGPASSWORD` or another libpq-supported mechanism when no password is passed to the connection.
`prompt`	If `TRUE`, prompt interactively for a WRDS PostgreSQL password when no password was supplied and no matching `.pgpass` entry exists.
`save`	If `TRUE`, save a supplied or prompted password to `.pgpass` after a successful WRDS connection test.
`passfile`	PostgreSQL password-file path. Defaults to `PGPASSFILE` when set, otherwise the platform default password-file path.

Value

A one-row tibble with ok, method, and message columns.

Examples

```
wrds_check_credentials(prompt = FALSE, save = FALSE)
```

---

wrds_connect_dbi *Connect to WRDS PostgreSQL with DBI*

Description

Opens a DBI connection to the WRDS PostgreSQL database using RPostgres and the db2pq WRDS authentication flow.

Usage

```
wrds_connect_dbi(
  wrds_id = NULL,
  prompt = interactive(),
  save = TRUE,
  passfile = NULL
)
```

Arguments

wrds_id	Optional WRDS username override.
prompt	If TRUE, prompt interactively for a WRDS PostgreSQL password when no password was supplied and no matching .pgpass entry exists.
save	If TRUE, save a supplied or prompted password to .pgpass after a successful WRDS connection test.
passfile	PostgreSQL password-file path. Defaults to PGPASSFILE when set, otherwise the platform default password-file path.

Details

The username is resolved from wrds_id, WRDS_ID, WRDS_USER, or the keyring entry written by wrds::wrds_set_credentials(). Password handling prefers a matching PostgreSQL .pgpass entry, then WRDS_PASSWORD, then an interactive password prompt when prompt = TRUE. When a supplied, prompted, or WRDS_PASSWORD password is used successfully and save = TRUE, it is saved to .pgpass for future connections.

Value

A DBIConnection object.

Examples

```
if (wrds_credentials_available(prompt = FALSE)) {
  con <- wrds_connect_dbi()
  DBI::dbDisconnect(con)
}
```

`wrds_conninfo` *Build WRDS PostgreSQL connection information*

Description

Build WRDS PostgreSQL connection information

Usage

```
wrds_conninfo(wrds_id = NULL, format = c("uri", "dbi"), sslmode = "require")
```

Arguments

`wrds_id`	Optional WRDS username override.
`format`	Return format. `"uri"` returns a PostgreSQL URI. `"dbi"` returns a list suitable for `DBI::dbConnect(RPostgres::Postgres(), !!!x)`.
`sslmode`	PostgreSQL SSL mode. WRDS requires SSL; the default is `"require"`.

Value

A character URI or a named list of DBI connection arguments.

Examples

```
wrds_conninfo(wrds_id = "example_user")
wrds_conninfo(wrds_id = "example_user", format = "dbi")
```

`wrds_credentials_available`
Check whether WRDS credentials are available

Description

Check whether WRDS credentials are available

Usage

```
wrds_credentials_available(
  wrds_id = NULL,
  password = NULL,
  prompt = interactive(),
  passfile = NULL
)
```

Arguments

wrds_id	Optional WRDS username override.
password	Optional password. If omitted, a matching .pgpass entry is preferred. When .pgpass has no matching entry, an existing WRDS_PASSWORD environment variable is used before an interactive prompt. PostgreSQL may also use PGPASSWORD or another libpq-supported mechanism when no password is passed to the connection.
prompt	If TRUE, prompt interactively for a WRDS PostgreSQL password when no password was supplied and no matching .pgpass entry exists.
passfile	PostgreSQL password-file path. Defaults to PGPASSFILE when set, otherwise the platform default password-file path.

Value

TRUE if wrds_check_credentials() can open a WRDS PostgreSQL connection, otherwise FALSE.

Examples

```
wrds_credentials_available(prompt = FALSE)
```

---

wrds_get_tables *List WRDS relations in a schema*

Description

List WRDS relations in a schema

Usage

```
wrds_get_tables(schema, wrds_id = NULL, views = FALSE)
```

Arguments

schema	WRDS schema name.
wrds_id	Optional WRDS username override.
views	If TRUE, include views as well as base tables.

Value

A character vector of relation names.

Examples

```
if (wrds_credentials_available(prompt = FALSE)) {
  wrds_get_tables("crsp")
}
```

`wrds_get_username` *Resolve a WRDS username*

Description

Resolves a WRDS username from, in order, an explicit argument, `WRDS_ID`, `WRDS_USER`, and the keyring entry written by `wrds::wrds_set_credentials()`.

Usage

```
wrds_get_username(
  wrds_id = NULL,
  user_key = "wrds_user",
  prompt = interactive()
)
```

Arguments

`wrds_id`	Optional WRDS username override.
`user_key`	Keyring key used by the `wrds` package for the WRDS username.
`prompt`	If `TRUE`, prompt interactively for a WRDS username when no explicit, environment, or keyring username can be resolved.

Value

A non-empty character string.

Examples

```
wrds_get_username(wrds_id = "example_user")
```

`wrds_schema_to_pq` *Export all tables in a WRDS schema to Parquet*

Description

Iterates over tables in a WRDS PostgreSQL schema and calls `wrds_update_pq` for each table.

Usage

```
wrds_schema_to_pq(
  schema,
  data_dir = NULL,
  force = FALSE,
  tables = NULL,
  views = FALSE,
  chunk_size = 100000L,
  wrds_id = NULL,
  transfer_method = c("dbi", "adbc"),
  numeric_mode = c("decimal", "float64", "text", "raw"),
  ...
)
```

Arguments

`schema`	WRDS schema name.
`data_dir`	Root directory of the local Parquet data repository.
`force`	If `TRUE`, re-download tables even if local files appear current.
`tables`	Optional subset of table names to process.
`views`	If `TRUE`, include views as well as base tables.
`chunk_size`	Number of rows fetched and written per chunk.
`wrds_id`	Optional WRDS username override.
`transfer_method`	Transfer backend passed to `wrds_update_pq`.
`numeric_mode`	Numeric handling mode passed to `wrds_update_pq`.
`...`	Additional arguments passed to `wrds_update_pq`.

Value

Invisibly returns a named list of output paths or `NULL` values for skipped/failed tables.

Examples

```
if (wrds_credentials_available(prompt = FALSE)) {
  wrds_schema_to_pq("crsp", data_dir = "~/pq_data")

  # Process a specific subset of tables
  wrds_schema_to_pq("crsp", data_dir = "~/pq_data",
                    tables = c("dsi", "dsf"))
}
```

`wrds_sql_to_pq` *Export a WRDS SQL query to Parquet*

Description

Runs SQL against WRDS PostgreSQL and writes the result to a Parquet file using the same DBI or ADBC transfer paths as `wrds_update_pq`.

Usage

```
wrds_sql_to_pq(
  sql,
  table_name,
  schema,
  wrds_id = NULL,
  data_dir = NULL,
  out_file = NULL,
  modified = NULL,
  alt_table_name = NULL,
  chunk_size = NULL,
  transfer_method = c("dbi", "adbc"),
  archive = FALSE,
  archive_dir = "archive",
  col_types = NULL
)
```

Arguments

`sql`	SQL query to execute against WRDS PostgreSQL.
`table_name`	Logical table name used for the output basename.
`schema`	Logical schema name used for the output directory.
`wrds_id`	Optional WRDS username override.
`data_dir`	Root directory of the local Parquet data repository.
`out_file`	Optional full output path.
`modified`	Optional last-modified metadata string to embed.
`alt_table_name`	Optional output basename when `out_file` is omitted.

chunk_size	Number of rows fetched and written per chunk.
transfer_method	Transfer backend: "dbi" or "adbc".
archive	Whether an existing output file should be archived before replacement.
archive_dir	Archive directory name.
col_types	Optional named list of Arrow output type overrides.

Value

Invisibly returns the output file path.

Examples

```
if (wrds_credentials_available(prompt = FALSE)) {
  wrds_sql_to_pq(
    "SELECT * FROM crsp.dsi WHERE date >= '2020-01-01'",
    table_name = "dsi_recent", schema = "crsp",
    data_dir = "~/pq_data"
  )
}
```

---

wrds_update_pq *Export a WRDS table to Parquet, skipping if already up to date*

Description

Exports a table from the WRDS PostgreSQL database to a Parquet file. Before downloading, the WRDS table comment is compared against the `last_modified` metadata embedded in any existing local Parquet file. The download is skipped if the local file is already up to date, making this function safe to call repeatedly as part of a scheduled data refresh.

Usage

```
wrds_update_pq(
  table_name,
  schema,
  data_dir = NULL,
  out_file = NULL,
  force = FALSE,
  where = NULL,
  obs = NULL,
  keep = NULL,
  drop = NULL,
  rename = NULL,
  alt_table_name = NULL,
  chunk_size = NULL,
  col_types = NULL,
```

```
  tz = "UTC",
  archive = FALSE,
  archive_dir = "archive",
  use_sas = FALSE,
  sas_schema = NULL,
  encoding = "utf-8",
  wrds_id = NULL,
  transfer_method = c("dbi", "adbc"),
  numeric_mode = c("decimal", "float64", "text", "raw")
)
```

Arguments

table_name	Name of the table in the WRDS PostgreSQL database.
schema	Name of the database schema (e.g. "crsp", "comp").
data_dir	Root directory of the local Parquet data repository. Defaults to the DATA_DIR environment variable, with interactive setup when needed. The output file is written to <data_dir>/<schema>/<table_name>.parquet.
out_file	Optional. Full path for the output Parquet file. Overrides the path derived from data_dir, schema, and table_name.
force	If TRUE, download proceeds regardless of the date comparison result.
where	Optional SQL WHERE clause (without the WHERE keyword) to filter rows before export. For example, where = "date > '2020-01-01'".
obs	Optional integer. Limits the number of rows imported using SQL LIMIT. Useful for testing with large tables (e.g. obs = 1000).
keep	Optional character vector of regex patterns. Only columns whose names match at least one pattern are retained. Applied after drop.
drop	Optional character vector of regex patterns. Columns whose names match at least one pattern are removed. Applied before keep.
rename	Optional named character vector or list mapping source column names to output column names. col_types names should refer to output names after renaming.
alt_table_name	Optional. Alternative basename for the output Parquet file, used when the file should have a different name from table_name.
chunk_size	Number of rows fetched and written per chunk. If omitted, defaults to 100000 for transfer_method = "dbi" and 250000 for transfer_method = "adbc".
col_types	Optional named list specifying column type overrides. Values may be string type names (e.g. "int32", "float32", "date") or Arrow DataType objects. Only columns that need to differ from their inferred types need to be supplied. See col_types = list(permno = "int32", ret = "float32"). String names such as "int32", "float32", "date", "timestamp", and "timestamptz" are supported.
tz	Time zone used to interpret TIMESTAMP WITHOUT TIME ZONE columns. Such columns are cast to TIMESTAMPTZ in the SQL query using AT TIME ZONE, so they are written as UTC-normalised timestamps in the Parquet file. Defaults to "UTC". Set to NULL to leave naive timestamps as-is.

archive	If TRUE, the existing local Parquet file (if any) is moved to the archive subdirectory before being replaced. The archived filename is <table>_<YYYYMMDDTHHMMSSz>.parquet, where the timestamp suffix is derived from the last_modified metadata embedded in the existing file (i.e. the date it was last downloaded, not the incoming WRDS table comment).
archive_dir	Name of the archive subdirectory relative to the schema directory. Defaults to "archive".
use_sas	If TRUE, the last-modified date is obtained by running PROC CONTENTS on the SAS dataset via SSH, rather than reading the PostgreSQL table comment. Requires the **ssh** package and SSH key access to wrds-cloud-sshkey.wharton.upenn.edu.
sas_schema	SAS library name to use when use_sas = TRUE. Defaults to schema.
encoding	Character encoding passed to PROC CONTENTS when use_sas = TRUE. Default is "utf-8".
wrds_id	WRDS user ID used for the SSH connection when use_sas = TRUE. Defaults to the WRDS_ID environment variable.
transfer_method	Transfer backend used for the SQL-to-Parquet step. Use "dbi" for the existing DBI/data-frame path or "adbc" for the experimental ADBC/Arrow path.
numeric_mode	Numeric handling mode for PostgreSQL numeric columns. The default "decimal" transfers values as text and writes bounded numeric(p, s) columns as Arrow decimals. Use "float64" to cast values to DOUBLE PRECISION before fetching, "text" to retain text values, or "raw" to keep the transfer backend's default representation.

Value

Invisibly returns the path to the active Parquet file. If the local file is already current, no download is performed but the same path is returned.

See Also

pq_last_modified, pq_archive, pq_restore

Examples

```
if (wrds_credentials_available(prompt = FALSE)) {
  wrds_update_pq("dsi", "crsp")
  wrds_update_pq("feed21_bankruptcy_notification", "audit")

  # Force re-download even if local file is current
  wrds_update_pq("dsi", "crsp", force = TRUE)

  # Limit columns and rows (useful for testing)
  wrds_update_pq("dsf", "crsp", obs = 1000, keep = c("permno", "date", "ret"))
}
```

Index