

Package ‘connector.databricks’

September 5, 2025

Title Expand 'connector' Package for 'Databricks' Tables and Volumes

Version 0.1.0

Description Expands the 'connector' <<https://github.com/NovoNordisk-OpenSource/connector>> package and provides a convenient interface for accessing and interacting with 'Databricks' <<https://www.databricks.com>> volumes and tables directly from R.

License Apache License (>= 2)

URL <https://novonordisk-opensource.github.io/connector.databricks/>,
<https://github.com/NovoNordisk-OpenSource/connector.databricks>

BugReports <https://github.com/NovoNordisk-OpenSource/connector.databricks/issues>

Imports arrow, brickster (>= 0.2.7), checkmate, cli, connector (>= 1.0.0), DBI, dbplyr, dplyr, fs, hms, odbc (>= 1.4.0), purrr, R6 (>= 2.4.0), rlang, withr, zephyr

Suggests glue, knitr, mockery (>= 0.4.4), rmarkdown, testthat (>= 3.2.3), tibble, whirl (>= 0.3.0)

VignetteBuilder knitr

Config/Needs/website rmarkdown

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.2

NeedsCompilation no

Author Vladimir Obucina [aut, cre],
Steffen Falgreen Larsen [aut],
Aksel Thomsen [aut],
Cervan Girard [aut],
Oliver Lundsgaard [ctb],
Skander Mulder [ctb],
Novo Nordisk A/S [cph]

Maintainer Vladimir Obucina <vlob@novonordisk.com>

Depends R (>= 4.1.0)

Repository CRAN

Date/Publication 2025-09-05 12:00:02 UTC

Contents

connector-options-databricks	2
ConnectorDatabricksTable	3
ConnectorDatabricksVolume	5
connector_databricks_table	6
connector_databricks_volume	8
create_directory_cnt	9
disconnect_cnt	10
download_cnt	10
download_directory_cnt	11
list_content_cnt	12
log_read_connector.ConnectorDatabricksTable	13
log_remove_connector.ConnectorDatabricksTable	14
log_write_connector.ConnectorDatabricksTable	15
read_cnt	16
remove_cnt	17
remove_directory_cnt	18
tbl_cnt	18
upload_cnt	19
upload_directory_cnt	20
write_cnt	21
Index	24

connector-options-databricks
Options for connector.databricks

Description

Configuration options for the connector.databricks

overwrite:

Overwrite existing content if it exists in the connector?

- Default: FALSE
- Option: `connector.databricks.overwrite`
- Environment: `R_CONNECTOR.DATABRICKS_OVERWRITE`

verbosity_level:

Verbosity level for functions in connector. See [zephyr::verbosity_level](#) for details.

- Default: "verbose"

- Option: `connector.databricks.verbosity_level`
- Environment: `R_CONNECTOR.DATABRICKS_VERBOSITY_LEVEL`

ConnectorDatabricksTable

Connector for connecting to Databricks using DBI

Description

Extension of the `connector::connector_dbi` making it easier to connect to, and work with tables in Databricks.

Details

All methods for `ConnectorDatabricksTable` object are working from the catalog and schema provided when initializing the connection. This means you only need to provide the table name when using the built in methods. If you want to access tables outside of the chosen schema, you can either retrieve the connection with `ConnectorDatabricksTable$conn` or create a new connector.

When creating the connections to Databricks you either need to provide the `sqlpath` to Databricks cluster or the SQL warehouse you want to connect to. Authentication to databricks is handed by the `odbc::databricks()` driver and supports general use of personal access tokens and credentials through Posit Workbench. See also `odbc::databricks()` On more information on how the connection to Databricks is established.

Super classes

`connector::Connector` -> `connector::ConnectorDBI` -> `ConnectorDatabricksTable`

Active bindings

`conn` The DBI connection object of the connector

`catalog` The catalog used in the connector

`schema` The schema used in the connector

Methods

Public methods:

- `ConnectorDatabricksTable$new()`
- `ConnectorDatabricksTable$clone()`

Method `new()`: Initialize the connection to Databricks

Usage:

```
ConnectorDatabricksTable$new(http_path, catalog, schema, extra_class = NULL)
```

Arguments:

http_path **character** The path to the Databricks cluster or SQL warehouse you want to connect to

catalog **character** The catalog to use

schema **character** The schema to use

extra_class **character** Extra class to assign to the new connector

Returns: A [ConnectorDatabricksTable](#) object

Method clone(): The objects of this class are cloneable with this method.

Usage:

```
ConnectorDatabricksTable$clone(deep = FALSE)
```

Arguments:

deep Whether to make a deep clone.

Examples

```
## Not run:
# Establish connection to your cluster

con_databricks <- ConnectorDatabricksTable$new(
  http_path = "path-to-cluster",
  catalog = "my_catalog",
  schema = "my_schema"
)

# List tables in my_schema

con_databricks$list_content()

# Read and write tables

con_databricks$write(mtcars, "my_mtcars_table")

con_databricks$read("my_mtcars_table")

# Use dplyr::tbl

con_databricks$tbl("my_mtcars_table")

# Remove table

con_databricks$remove("my_mtcars_table")

# Disconnect

con_databricks$disconnect()

## End(Not run)
```

ConnectorDatabricksVolume

Connector for databricks volume storage

Description

The ConnectorDatabricksVolume class, built on top of [connector::connector](#) class. It is a file storage connector for accessing and manipulating files inside Databricks volumes.

Super classes

[connector::Connector](#) -> [connector::ConnectorFS](#) -> ConnectorDatabricksVolume

Active bindings

path [character](#) Path to the file storage on Volume

catalog [character](#) Databricks catalog

schema [character](#) Databricks schema

full_path [character](#) Full path to the file storage on Volume

Methods

Public methods:

- [ConnectorDatabricksVolume\\$new\(\)](#)
- [ConnectorDatabricksVolume\\$clone\(\)](#)

Method [new\(\)](#): Initializes the connector for Databricks volume storage.

Usage:

```
ConnectorDatabricksVolume$new(
  full_path = NULL,
  catalog = NULL,
  schema = NULL,
  path = NULL,
  extra_class = NULL,
  force = FALSE,
  ...
)
```

Arguments:

full_path [character](#) Full path to the file storage in format catalog/schema/path. If NULL, catalog, schema, and path must be provided.

catalog [character](#) Databricks catalog

schema [character](#) Databricks schema

path [character](#) Path to the file storage

extra_class [character](#) Extra class to assign to the new connector.

force [logical](#) If TRUE, the volume will be created without asking if it does not exist.

... Additional arguments passed to the initialize method of superclass

Returns: A new [ConnectorDatabricksVolume](#) object

Method `clone()`: The objects of this class are cloneable with this method.

Usage:

```
ConnectorDatabricksVolume$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

Examples

```
## Not run:
# Create Volume file storage connector
cnt <- ConnectorDatabricksVolume$new(full_path = "catalog/schema/path")

cnt

# List content
cnt$list_content_cnt()

# Write to the connector
cnt$write_cnt(iris, "iris.rds")

# Check it is there
cnt$list_content_cnt()

# Read the result back
cnt$read_cnt("iris.rds") |>
  head()

## End(Not run)
```

connector_databricks_table

Create ConnectorDatabricksTable connector

Description

Initializes the connector for table type of storage. See [ConnectorDatabricksTable](#) for details.

Usage

```
connector_databricks_table(http_path, catalog, schema, extra_class = NULL)
```

Arguments

http_path	character	The path to the Databricks cluster or SQL warehouse you want to connect to
catalog	character	The catalog to use
schema	character	The schema to use
extra_class	character	Extra class to assign to the new connector

Details

The `extra_class` parameter allows you to create a subclass of the `ConnectorDatabricksTable` object. This can be useful if you want to create a custom connection object for easier dispatch of new `s3` methods, while still inheriting the methods from the `ConnectorDatabricksTable` object.

Value

A new `ConnectorDatabricksTable` object

Examples

```
## Not run:
# Establish connection to your cluster

con_databricks <- connector_databricks_table(
  http_path = "path-to-cluster",
  catalog = "my_catalog",
  schema = "my_schema"
)

# List tables in my_schema

con_databricks$list_content()

# Read and write tables

con_databricks$write(mtcars, "my_mtcars_table")

con_databricks$read("my_mtcars_table")

# Use dplyr::tbl

con_databricks$tbl("my_mtcars_table")

# Remove table

con_databricks$remove("my_mtcars_table")

# Disconnect

con_databricks$disconnect()

## End(Not run)
```

`connector_databricks_volume`*Create databricks volume connector*

Description

Create a new databricks volume connector object. See [ConnectorDatabricksVolume](#) for details.

Initializes the connector for Databricks volume storage.

Usage

```
connector_databricks_volume(  
  full_path = NULL,  
  catalog = NULL,  
  schema = NULL,  
  path = NULL,  
  extra_class = NULL,  
  force = FALSE,  
  ...  
)
```

Arguments

<code>full_path</code>	Full path to the file storage in format catalog/schema/path. If NULL, catalog, schema, and path must be provided.
<code>catalog</code>	Databricks catalog
<code>schema</code>	Databricks schema
<code>path</code>	Path to the file storage
<code>extra_class</code>	Extra class to assign to the new connector.
<code>force</code>	If TRUE, the volume will be created without asking if it does not exist.
<code>...</code>	Additional arguments passed to the connector::connector

Details

The `extra_class` parameter allows you to create a subclass of the `ConnectorDatabricksVolume` object. This can be useful if you want to create a custom connection object for easier dispatch of new s3 methods, while still inheriting the methods from the `ConnectorDatabricksVolume` object.

Value

A new [ConnectorDatabricksVolume](#) object

Examples

```
## Not run:
# Connect to a file system
databricks_volume <- "catalog/schema/path"
db <- connector_databricks_volume(databricks_volume)

db

# Create subclass connection
db_subclass <- connector_databricks_volume(databricks_volume,
  extra_class = "subclass"
)

db_subclass
class(db_subclass)

## End(Not run)
```

create_directory_cnt *Create a directory*

Description

Additional list content methods for Databricks connectors implemented for `connector::create_directory_cnt()`:

- **ConnectorDatabricksVolume**: Reuses the `connector::create_directory_cnt()` method for **ConnectorDatabricksVolume**, but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```
create_directory_cnt(connector_object, name, open = TRUE, ...)
```

```
## S3 method for class 'ConnectorDatabricksVolume'
create_directory_cnt(connector_object, name, open = TRUE, ...)
```

Arguments

```
connector_object      Connector The connector object to use.
name                  character The name of the directory to create
open                  create a new connector object
...                   ConnectorDatabricksVolume: Additional parameters to pass to the brickster::db_volume_dir_create
                      method
```

Value

invisible connector_object.

disconnect_cnt	<i>Disconnect (close) the connection of the connector</i>
----------------	---

Description

Generic implementing of how to disconnect from the relevant connections. Mostly relevant for DBI connectors.

- [ConnectorDBI](#): Uses `DBI::dbDisconnect()` to create a table reference to close a DBI connection.

Usage

```
disconnect_cnt(connector_object, ...)
```

Arguments

connector_object

[Connector](#) The connector object to use.

...

Additional arguments passed to the method for the individual connector.

Value

[invisible](#) connector_object.

download_cnt	<i>Download content from the connector</i>
--------------	--

Description

Additional list content methods for Databricks connectors implemented for `connector::download_cnt()`:

- [ConnectorDatabricksVolume](#): Reuses the `connector::download_cnt()` method for [ConnectorDatabricksVolume](#), but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```
download_cnt(connector_object, src, dest = basename(src), ...)
```

```
## S3 method for class 'ConnectorDatabricksVolume'
download_cnt(connector_object, src, dest = basename(src), ...)
```

Arguments

connector_object	Connector The connector object to use.
src	character Name of the content to read, write, or remove. Typically the table name.
dest	character Path to the file to download to or upload from
...	ConnectorDatabricksVolume : Additional parameters to pass to the brickster::db_volume_read() method

Value

[invisible](#) connector_object.

download_directory_cnt

Download a directory

Description

Additional list content methods for Databricks connectors implemented for [connector::download_directory_cnt\(\)](#):

- [ConnectorDatabricksVolume](#): Reuses the [connector::download_directory_cnt\(\)](#) method for [ConnectorDatabricksVolume](#), but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```
download_directory_cnt(connector_object, src, dest = basename(src), ...)
```

```
## S3 method for class 'ConnectorDatabricksVolume'
download_directory_cnt(connector_object, src, dest = basename(src), ...)
```

Arguments

connector_object	Connector The connector object to use.
src	character The name of the directory to download from the connector
dest	character Path to the directory to download to
...	ConnectorDatabricksVolume : Additional parameters to pass to the brickster::db_volume_dir_create method

Value

[invisible](#) connector_object.

list_content_cnt	<i>List available content from the connector</i>
------------------	--

Description

Additional list content methods for Databricks connectors implemented for `connector::list_content_cnt()`:

- **ConnectorDatabricksTable**: Reuses the `connector::list_content_cnt()` method for **ConnectorDatabricksTable**, but always sets the catalog and schema as defined in when initializing the connector.
- **ConnectorDatabricksVolume**: Reuses the `connector::list_content_cnt()` method for **ConnectorDatabricksVolume**, but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```
list_content_cnt(connector_object, ...)

## S3 method for class 'ConnectorDatabricksTable'
list_content_cnt(connector_object, ..., tags = NULL)

## S3 method for class 'ConnectorDatabricksVolume'
list_content_cnt(connector_object, ...)
```

Arguments

connector_object	Connector The connector object to use.
...	ConnectorDatabricksVolume : Additional parameters to pass to the <code>brickster::db_volume_list()</code> method
tags	Expression to be translated to SQL using <code>dbplyr::translate_sql()</code> e.g. <code>((tag_name == "name1" && tag_value == "value1") (tag_name == "name2"))</code> . It should contain <code>tag_name</code> and <code>tag_value</code> values to filter by.

Value

A **character** vector of content names

log_read_connector.ConnectorDatabricksTable
Connector Logging Functions

Description

- [ConnectorDatabricksTable](#): Implementation of the log_read_connector function for the ConnectorDatabricksTable class.
- [ConnectorDatabricksVolume](#): Implementation of the log_read_connector function for the ConnectorDatabricksVolume class.

Additional log read methods for Databricks connectors implemented for `connector::log_read_connector()`:

Usage

```
## S3 method for class 'ConnectorDatabricksTable'  
log_read_connector(connector_object, name, ...)
```

```
## S3 method for class 'ConnectorDatabricksVolume'  
log_read_connector(connector_object, name, ...)
```

```
log_read_connector(connector_object, name, ...)
```

Arguments

connector_object	The connector object to log operations for. Can be any connector class (ConnectorFS, ConnectorDBI, ConnectorLogger, etc.)
name	Character string specifying the name or identifier of the resource being operated on (e.g., file name, table name)
...	Additional parameters passed to specific method implementations. May include connector-specific options or metadata.

Details

Connector Logging Functions

The logging system is built around S3 generic functions that dispatch to specific implementations based on the connector class. Each operation is logged with contextual information including connector details, operation type, and resource names.

Value

These are primarily side-effect functions that perform logging. The actual return value depends on the specific method implementation, typically:

- `log_read_connector`: Result of the read operation

- `log_write_connector`: Invisible result of write operation
- `log_remove_connector`: Invisible result of remove operation
- `log_list_content_connector`: List of connector contents

`log_remove_connector.ConnectorDatabricksTable`
Connector Logging Functions

Description

- [ConnectorDatabricksTable](#): Implementation of the `log_remove_connector` function for the `ConnectorDatabricksTable` class.
- [ConnectorDatabricksVolume](#): Implementation of the `log_remove_connector` function for the `ConnectorDatabricksVolume` class.

Additional log remove methods for Databricks connectors implemented for `connector::log_remove_connector()`:

Usage

```
## S3 method for class 'ConnectorDatabricksTable'
log_remove_connector(connector_object, name, ...)

## S3 method for class 'ConnectorDatabricksVolume'
log_remove_connector(connector_object, name, ...)

log_remove_connector(connector_object, name, ...)
```

Arguments

<code>connector_object</code>	The connector object to log operations for. Can be any connector class (<code>ConnectorFS</code> , <code>ConnectorDBI</code> , <code>ConnectorLogger</code> , etc.)
<code>name</code>	Character string specifying the name or identifier of the resource being operated on (e.g., file name, table name)
<code>...</code>	Additional parameters passed to specific method implementations. May include connector-specific options or metadata.

Details

Connector Logging Functions

The logging system is built around S3 generic functions that dispatch to specific implementations based on the connector class. Each operation is logged with contextual information including connector details, operation type, and resource names.

Value

These are primarily side-effect functions that perform logging. The actual return value depends on the specific method implementation, typically:

- `log_read_connector`: Result of the read operation
- `log_write_connector`: Invisible result of write operation
- `log_remove_connector`: Invisible result of remove operation
- `log_list_content_connector`: List of connector contents

log_write_connector.ConnectorDatabricksTable
Connector Logging Functions

Description

- [ConnectorDatabricksTable](#): Implementation of the `log_write_connector` function for the `ConnectorDatabricksTable` class.
- [ConnectorDatabricksVolume](#): Implementation of the `log_write_connector` function for the `ConnectorDatabricksVolume` class.

Additional log write methods for Databricks connectors implemented for `connector::log_write_connector()`:

Usage

```
## S3 method for class 'ConnectorDatabricksTable'
log_write_connector(connector_object, name, ...)

## S3 method for class 'ConnectorDatabricksVolume'
log_write_connector(connector_object, name, ...)

log_write_connector(connector_object, name, ...)
```

Arguments

<code>connector_object</code>	The connector object to log operations for. Can be any connector class (<code>ConnectorFS</code> , <code>ConnectorDBI</code> , <code>ConnectorLogger</code> , etc.)
<code>name</code>	Character string specifying the name or identifier of the resource being operated on (e.g., file name, table name)
<code>...</code>	Additional parameters passed to specific method implementations. May include connector-specific options or metadata.

Details

Connector Logging Functions

The logging system is built around S3 generic functions that dispatch to specific implementations based on the connector class. Each operation is logged with contextual information including connector details, operation type, and resource names.

Value

These are primarily side-effect functions that perform logging. The actual return value depends on the specific method implementation, typically:

- `log_read_connector`: Result of the read operation
- `log_write_connector`: Invisible result of write operation
- `log_remove_connector`: Invisible result of remove operation
- `log_list_content_connector`: List of connector contents

read_cnt

Read content from the connector

Description

Additional read methods for Databricks connectors implemented for `connector::read_cnt()`:

- **ConnectorDatabricksTable**: Reuses the `connector::read_cnt()` method for **ConnectorDatabricksTable**, but always sets the catalog and schema as defined in when initializing the connector.
- **ConnectorDatabricksVolume**: Reuses the `connector::read_cnt()` method for **ConnectorDatabricksVolume**, but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```
read_cnt(connector_object, name, ...)
```

```
## S3 method for class 'ConnectorDatabricksTable'
read_cnt(connector_object, name, ..., timepoint = NULL, version = NULL)
```

```
## S3 method for class 'ConnectorDatabricksVolume'
read_cnt(connector_object, name, ...)
```

Arguments

connector_object	Connector The connector object to use.
name	character Name of the content to read, write, or remove. Typically the table name.
...	ConnectorDatabricksVolume : Additional parameters to pass to the <code>brickster::db_volume_read()</code> method
timepoint	Timepoint in Delta time travel syntax format.
version	Table version generated by the operation.

Value

R object with the content. For rectangular data a [data.frame](#).

remove_cnt	<i>Remove content from the connector</i>
------------	--

Description

Additional remove methods for Databricks connectors implemented for `connector::remove_cnt()`:

- [ConnectorDatabricksTable](#): Reuses the `connector::list_content_cnt()` method for [ConnectorDatabricksTable](#), but always sets the catalog and schema as defined in when initializing the connector.
- [ConnectorDatabricksVolume](#): Reuses the `connector::remove_cnt()` method for [ConnectorDatabricksVolume](#), but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```
remove_cnt(connector_object, name, ...)

## S3 method for class 'ConnectorDatabricksTable'
remove_cnt(connector_object, name, ...)

## S3 method for class 'ConnectorDatabricksVolume'
remove_cnt(connector_object, name, ...)
```

Arguments

connector_object	Connector The connector object to use.
name	character Name of the content to read, write, or remove. Typically the table name.
...	ConnectorDatabricksTable : Additional parameters to pass to the <code>brickster::db_uc_tables_delete()</code> method

Value

[invisible](#) connector_object.

remove_directory_cnt *Remove a directory*

Description

Additional list content methods for Databricks connectors implemented for `connector::remove_directory_cnt()`:

- `ConnectorDatabricksVolume`: Reuses the `connector::remove_directory_cnt()` method for `ConnectorDatabricksVolume`, but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```
remove_directory_cnt(connector_object, name, ...)
```

```
## S3 method for class 'ConnectorDatabricksVolume'
remove_directory_cnt(connector_object, name, ...)
```

Arguments

connector_object

[Connector](#) The connector object to use.

name [character](#) The name of the directory to remove

... [ConnectorDatabricksVolume](#): Additional parameters to pass to the `brickster::db_volume_dir_delete` method

Value

[invisible](#) connector_object.

tbl_cnt *Use dplyr verbs to interact with the remote database table*

Description

Additional tbl methods for Databricks connectors implemented for `connector::tbl_cnt()`:

- `ConnectorDatabricksTable`: Reuses the `connector::list_content_cnt()` method for `ConnectorDatabricksTable`, but always sets the catalog and schema as defined in when initializing the connector.
- `ConnectorDatabricksVolume`: Reuses the `connector::remove_directory_cnt()` method for `ConnectorDatabricksVolume`, but always sets the catalog, schema and path as defined in when initializing the connector. Uses `read_cnt()` to allow redundancy between Volumes and Tables.

Usage

```
tbl_cnt(connector_object, name, ...)

## S3 method for class 'ConnectorDatabricksTable'
tbl_cnt(connector_object, name, ...)

## S3 method for class 'ConnectorDatabricksVolume'
tbl_cnt(connector_object, name, ...)
```

Arguments

connector_object [Connector](#) The connector object to use.

name [character](#) Name of the content to read, write, or remove. Typically the table name.

... Additional arguments passed to the method for the individual connector.

Value

A [dplyr::tbl](#) object.

upload_cnt	<i>Upload content to the connector</i>
------------	--

Description

Additional list content methods for Databricks connectors implemented for [connector::upload_cnt\(\)](#):

- [ConnectorDatabricksVolume](#): Reuses the [connector::upload_cnt\(\)](#) method for [ConnectorDatabricksVolume](#), but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```
upload_cnt(
  connector_object,
  src,
  dest = basename(src),
  overwrite = zephyr::get_option("overwrite", "connector"),
  ...
)

## S3 method for class 'ConnectorDatabricksVolume'
upload_cnt(
  connector_object,
  src,
```

```

    dest = basename(src),
    overwrite = zephyr::get_option("overwrite", "connector.databricks"),
    ...
)

```

Arguments

connector_object	Connector The connector object to use.
src	character Path to the file to download to or upload from
dest	character Name of the content to read, write, or remove. Typically the table name.
overwrite	Overwrites existing content if it exists in the connector.
...	ConnectorDatabricksVolume : Additional parameters to pass to the <code>brickster::db_volume_write()</code> method

Value

invisible connector_object.

upload_directory_cnt *Upload a directory*

Description

Additional list content methods for Databricks connectors implemented for `connector::upload_directory_cnt()`:

- **ConnectorDatabricksVolume**: Reuses the `connector::upload_directory_cnt()` method for **ConnectorDatabricksVolume**, but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```

upload_directory_cnt(
  connector_object,
  src,
  dest,
  overwrite = zephyr::get_option("overwrite", "connector"),
  open = FALSE,
  ...
)

## S3 method for class 'ConnectorDatabricksVolume'
upload_directory_cnt(
  connector_object,
  src,

```

```

    dest = basename(src),
    overwrite = zephyr::get_option("overwrite", "connector"),
    open = FALSE,
    ...
)

```

Arguments

connector_object	Connector The connector object to use.
src	character Path to the directory to upload
dest	character The name of the new directory to place the content in
overwrite	Overwrite existing content if it exists in the connector? See connector-options for details. Default can be set globally with <code>options(connector.overwrite = TRUE/FALSE)</code> or environment variable <code>R_CONNECTOR_OVERWRITE..</code> Default: <code>FALSE</code> .
open	logical Open the directory as a new connector object.
...	ConnectorDatabricksVolume : Additional parameters to pass to the <code>brickster::db_volume_dir_create</code> method

Value

invisible connector_object.

write_cnt	<i>Write content to the connector</i>
-----------	---------------------------------------

Description

Additional write methods for Databricks connectors implemented for `connector::write_cnt()`:

- **ConnectorDatabricksTable**: Reuses the `connector::write_cnt()` method for **ConnectorDatabricksTable**, but always sets the catalog and schema as defined in when initializing the connector. Creates temporary volume to write object as a parquet file and then convert it to a table.
- **ConnectorDatabricksVolume**: Reuses the `connector::write_cnt()` method for **ConnectorDatabricksVolume**, but always sets the catalog, schema and path as defined in when initializing the connector.

Usage

```

write_cnt(
  connector_object,
  x,
  name,
  overwrite = zephyr::get_option("overwrite", "connector"),
  ...
)

## S3 method for class 'ConnectorDatabricksTable'
write_cnt(
  connector_object,
  x,
  name,
  overwrite = zephyr::get_option("overwrite", "connector.databricks"),
  ...,
  method = "volume",
  tags = NULL
)

## S3 method for class 'ConnectorDatabricksVolume'
write_cnt(
  connector_object,
  x,
  name,
  overwrite = zephyr::get_option("overwrite", "connector.databricks"),
  ...
)

```

Arguments

connector_object	Connector The connector object to use.
x	The object to write to the connection
name	character Name of the content to read, write, or remove. Typically the table name.
overwrite	Overwrite existing content if it exists in the connector.
...	ConnectorDatabricksVolume : Additional parameters to pass to the brickster::db_volume_write() method
method	<ul style="list-style-type: none"> ConnectorDatabricksTable: Which method to use for writing the table. Options: <ul style="list-style-type: none"> – volume - using temporary volume to write data and then convert it to a table.
tags	<ul style="list-style-type: none"> ConnectorDatabricksTable: Named list containing tag names and tag values, e.g. <code>list("tag_name1" = "tag_value1", "tag_name2" = "tag_value2")</code>. More info here

write_cnt

23

Value

[invisible](#) connector_object.

Index

brickster::db_uc_tables_delete(), 17
brickster::db_volume_dir_create, 9
brickster::db_volume_dir_create(), 11, 21
brickster::db_volume_dir_delete(), 18
brickster::db_volume_list(), 12
brickster::db_volume_read(), 11, 17
brickster::db_volume_write(), 20, 22

character, 4, 5, 7, 9, 11, 12, 17–22
Connector, 9–12, 17–22
connector-options, 21
connector-options-databricks, 2
connector::Connector, 3, 5
connector::connector, 5, 8
connector::connector_dbi, 3
connector::ConnectorDBI, 3
connector::ConnectorFS, 5
connector::create_directory_cnt(), 9
connector::download_cnt(), 10
connector::download_directory_cnt(), 11
connector::list_content_cnt(), 12, 17, 18
connector::log_read_connector(), 13
connector::log_remove_connector(), 14
connector::log_write_connector(), 15
connector::read_cnt(), 16
connector::remove_cnt(), 17
connector::remove_directory_cnt(), 18
connector::tbl_cnt(), 18
connector::upload_cnt(), 19
connector::upload_directory_cnt(), 20
connector::write_cnt(), 21
connector_databricks_table, 6
connector_databricks_volume, 8
ConnectorDatabricksTable, 3, 3, 4, 6, 7, 12–18, 21, 22
ConnectorDatabricksVolume, 5, 6, 8–22
ConnectorDBI, 10

create_directory_cnt, 9

data.frame, 17
DBI::dbDisconnect(), 10
dbplyr::translate_sql(), 12
disconnect_cnt, 10
download_cnt, 10
download_directory_cnt, 11
dplyr::tbl, 19

invisible, 9–11, 18, 20, 21, 23

list_content_cnt, 12
log_read_connector
 (log_read_connector.ConnectorDatabricksTable), 13
log_read_connector.ConnectorDatabricksTable, 13
log_remove_connector
 (log_remove_connector.ConnectorDatabricksTable), 14
log_remove_connector.ConnectorDatabricksTable, 14
log_write_connector
 (log_write_connector.ConnectorDatabricksTable), 15
log_write_connector.ConnectorDatabricksTable, 15

logical, 6, 21

odbc::databricks(), 3

read_cnt, 16
read_cnt(), 18
remove_cnt, 17
remove_directory_cnt, 18

tbl_cnt, 18

upload_cnt, 19
upload_directory_cnt, 20

`write_cnt`, [21](#)

`zephyr::verbosity_level`, [2](#)