

# Package ‘implyr’

June 27, 2017

**Type** Package

**Title** R Interface for Apache Impala

**Version** 0.2.1

**Maintainer** Ian Cook <ian@cloudera.com>

**Description** 'SQL' back-end to 'dplyr' for Apache Impala (incubating), the massively parallel processing query engine for Apache 'Hadoop'. Impala enables low-latency 'SQL' queries on data stored in the 'Hadoop' Distributed File System '(HDFS)', Apache 'HBase', Apache 'Kudu', and Amazon Simple Storage Service '(S3)'. See <<https://impala.apache.org>> for more information about Impala.

**URL** <https://github.com/ianmcook/implyr>

**BugReports** <https://github.com/ianmcook/implyr/issues>

**Depends** R (>= 3.1.2), dplyr (>= 0.5.0.9005)

**Imports** assertthat, DBI (>= 0.6-1), dbplyr (>= 0.0.0.9001), methods, rlang (>= 0.1.0), utils

**Suggests** Lahman (>= 3.0-1), odbc, RJDBC, nycflights13, testthat

**SystemRequirements** Impala driver to support a 'DBI'-compatible R interface

**NeedsCompilation** no

**License** Apache License 2.0 | file LICENSE

**RoxygenNote** 6.0.1

**Author** Ian Cook [aut, cre],  
Cloudera [cph]

**Repository** CRAN

**Date/Publication** 2017-06-27 21:16:44 UTC

## R topics documented:

compute	2
copy_to	3

dbDisconnect,src_impala-method . . . . .	4
dbExecute,src_impala,character-method . . . . .	5
dbGetQuery,src_impala,character-method . . . . .	6
db_desc . . . . .	7
src_impala . . . . .	7
tbl . . . . .	9

<b>Index</b>	<b>10</b>
--------------	-----------

---

compute	<i>Force execution of an Impala query</i>
---------	---

---

## Description

compute() Executes the query and stores the result in a new Impala table  
 collect() Executes the query and returns the result to R as a data frame tbl  
 collapse() Generates the query for later execution

## Usage

```
## S3 method for class 'tbl_impala'
compute(x, name, temporary = TRUE,
        unique_indexes = NULL, indexes = NULL, analyze = FALSE,
        external = FALSE, overwrite = FALSE, force = FALSE,
        field_terminator = NULL, line_terminator = NULL, file_format = NULL,
        ...)
```

```
## S3 method for class 'tbl_impala'
collect(x, ..., n = Inf, warn_incomplete = TRUE)
```

```
## S3 method for class 'tbl_impala'
collapse(x, vars = NULL, ...)
```

## Arguments

x	an object with class tbl_impala
name	the name for the new Impala table
temporary	must be set to FALSE
unique_indexes	not used
indexes	not used
analyze	whether to run COMPUTE STATS after adding data to the new table
external	whether the new table will be externally managed
overwrite	whether to overwrite existing table data (currently ignored)
force	whether to silently fail if the table already exists

field_terminator	the delimiter to use between fields in text file data. Defaults to the ASCII control-A (hex 01) character
line_terminator	the line terminator. Defaults to "\n"
file_format	the storage format to use. Options are "TEXTFILE" (default) and "PARQUET"
...	other arguments passed on to methods
n	the number of rows to return
warn_incomplete	whether to issue a warning if not all rows retrieved
vars	not used

**Note**

Impala does not support temporary tables. When using `compute()` to store results in an Impala table, you must set `temporary = FALSE`.

---

copy_to	<i>Copy a (very small) local data frame to Impala</i>
---------	---

---

**Description**

`copy_to` inserts the contents of a local data frame into a new Impala table. `copy_to` currently only supports very small data frames (1000 or fewer row/column positions). It uses the SQL `INSERT ... VALUES()` technique, which is not suitable for loading large amounts of data.

This package does not provide tools for loading larger amounts of local data into Impala tables. This is because Impala can query data stored in several different filesystems and storage systems (HDFS, Apache Kudu, Apache HBase, and Amazon S3) and Impala does not include built-in capability for loading local data into these systems.

**Usage**

```
## S3 method for class 'src_impala'
copy_to(dest, df, name = deparse(substitute(df)),
        overwrite = FALSE, types = NULL, temporary = TRUE,
        unique_indexes = NULL, indexes = NULL, analyze = TRUE,
        external = FALSE, force = FALSE, field_terminator = NULL,
        line_terminator = NULL, file_format = NULL, ...)
```

**Arguments**

<code>dest</code>	an object with class with class <code>src_impala</code>
<code>df</code>	a (very small) local data frame
<code>name</code>	name for the new Impala table
<code>overwrite</code>	whether to overwrite existing table data (currently ignored)

types	a character vector giving variable types to use for the columns
temporary	must be set to FALSE
unique_indexes	not used
indexes	not used
analyze	whether to run COMPUTE STATS after adding data to the new table
external	whether the new table will be externally managed
force	whether to silently continue if the table already exists
field_terminator	the delimiter to use between fields in text file data. Defaults to the ASCII control-A (hex 01) character
line_terminator	the line terminator. Defaults to "\n"
file_format	the storage format to use. Options are "TEXTFILE" (default) and "PARQUET"
...	other arguments passed on to methods

**Value**

An object with class tbl\_impala, tbl\_sql, tbl\_lazy, tbl

**Note**

Impala does not support temporary tables. When using copy\_to() to insert local data into an Impala table, you must set temporary = FALSE.

**Examples**

```
library(nycflights13)
dim(airlines) # airlines data frame is very small
# [1] 16 2

## Not run:
copy_to(impala, airlines, temporary = FALSE)
## End(Not run)
```

---

```
dbDisconnect,src_impala-method
      Close the connection to Impala
```

---

**Description**

Closes (disconnects) the connection to Impala.

**Usage**

```
## S4 method for signature 'src_impala'
dbDisconnect(conn, ...)
```

**Arguments**

conn            object with class class src\_impala  
...            other arguments passed on to methods

**Value**

Returns TRUE, invisibly

**Examples**

```
## Not run:  
dbDisconnect(impala)  
## End(Not run)
```

---

dbExecute,src\_impala,character-method

*Execute an Impala statement that returns no result*

---

**Description**

Executes an Impala statement that returns no result.

**Usage**

```
## S4 method for signature 'src_impala,character'  
dbExecute(conn, statement, ...)
```

**Arguments**

conn            object with class class src\_impala  
statement       a character string containing SQL  
...            other arguments passed on to methods

**Value**

Depending on the package used to connect to Impala, either a scalar numeric that specifies the number of rows affected by the statement, or NULL

**Note**

This method is for statements that return no result, such as data definition or data manipulation statements. Use [dbGetQuery\(\)](#) for SELECT queries.

**Examples**

```
## Not run:  
dbExecute(impala, "INVALIDATE METADATA")  
## End(Not run)
```

---

`dbGetQuery,src_impala,character-method`*Send SQL query to Impala and retrieve results*

---

## Description

Returns the result of an Impala SQL query as a data frame.

## Usage

```
## S4 method for signature 'src_impala,character'  
dbGetQuery(conn, statement, ...)
```

## Arguments

<code>conn</code>	object with class <code>class src_impala</code>
<code>statement</code>	a character string containing SQL
<code>...</code>	other arguments passed on to methods

## Value

A `data.frame` with as many rows as records were fetched and as many columns as fields in the result set, even if the result is a single value or has one or zero rows

## Note

This method is for SELECT queries only. Use `dbExecute()` for data definition or data manipulation statements.

## Examples

```
## Not run:  
flights_by_carrier_df <- dbGetQuery(  
  impala,  
  "SELECT carrier, COUNT(*) FROM flights GROUP BY carrier"  
)  
## End(Not run)
```

---

db_desc	<i>Describe the Impala data source</i>
---------	--

---

**Description**

Describe the Impala data source

**Usage**

```
## S3 method for class 'impala_connection'
db_desc(x)
```

**Arguments**

x                    an object with class class impala\_connection

**Value**

A string containing information about the connection to Impala

---

src_impala	<i>Connect to Impala and create a remote dplyr data source</i>
------------	--

---

**Description**

src\_impala creates a SQL backend to dplyr for [Apache Impala \(incubating\)](#), the massively parallel processing query engine for Apache Hadoop.

src\_impala can work with any DBI-compatible interface that provides connectivity to Impala. Currently, two packages that can provide this connectivity are `odbc` and `RJDBC`.

**Usage**

```
src_impala(drv, ..., auto_disconnect = TRUE)
```

**Arguments**

drv                    an object that inherits from [DBIDriver-class](#). For example, an object returned by `odbc` or `JDBC`

...                    arguments passed to the underlying Impala database connection method [dbConnect](#). See [dbConnect](#), [OdbcDriver-method](#) or [dbConnect](#), [JDBCdriver-method](#)

auto\_disconnect        Should the connection to Impala be automatically closed when the object returned by this function is deleted? Pass NA to auto-disconnect but print a message when this happens.

**Value**

An object with class `src_impala`, `src_sql`, `src`

**See Also**

[Impala ODBC driver](#), [Impala JDBC driver](#)

**Examples**

```
# Using ODBC connectivity:

## Not run:
library(odbc)
drv <- odbc::odbc()
impala <- src_impala(
  drv = drv,
  driver = "Cloudera ODBC Driver for Impala",
  host = "host",
  port = 21050,
  database = "default",
  uid = "username",
  pwd = "password"
)
## End(Not run)

# Using JDBC connectivity:

## Not run:
library(RJDBC)
Sys.setenv(JAVA_HOME = "/path/to/java/home/")
impala_classpath <- list.files(
  path = "/path/to/jdbc/driver",
  pattern = "\\\\.jar$",
  full.names = TRUE
)
.jinit(classpath = impala_classpath)
drv <- JDBC(
  driverClass = "com.cloudera.impala.jdbc41.Driver",
  classPath = impala_classpath,
  identifier.quote = ""
)
impala <- src_impala(
  drv,
  "jdbc:impala://host:21050",
  "username",
  "password"
)
## End(Not run)
```



---

tbl	<i>Create a lazy tbl from an Impala table</i>
-----	---

---

**Description**

Create a lazy tbl from an Impala table

**Usage**

```
## S3 method for class 'src_impala'  
tbl(src, from, ...)
```

**Arguments**

src	an object with class with class src_impala
from	a table name or identifier
...	not used

**Value**

An object with class tbl\_impala, tbl\_sql, tbl\_lazy, tbl

**See Also**

[in\\_schema](#)

**Examples**

```
## Not run:  
flights_tbl <- tbl(impala, "flights")  
  
flights_tbl <- tbl(impala, in_schema("nycflights13", "flights"))  
## End(Not run)
```

# Index

`collapse (compute)`, 2  
`collect (compute)`, 2  
`compute`, 2  
`copy_to`, 3

`db_desc`, 7  
`dbConnect`, 7  
`dbDisconnect`, `src_impala`-method, 4  
`dbExecute()`, 6  
`dbExecute`, `src_impala`, `character`-method,  
5  
`dbGetQuery()`, 5  
`dbGetQuery`, `src_impala`, `character`-method,  
6

`in_schema`, 9

JDBC, 7

odbc, 7

`src_impala`, 7

`tbl`, 9