Package 'tdR'

March 23, 2016

Title Teradata interaction and query package

Version 0.0.1.0000

Author Linh Tran

Maintainer Linh Tran <`linh_m_tran@apple.com`>

Description Helper functions for Teradata queries and other miscellaneous things.

Depends R (>= 3.2.2), RJDBC

Imports RJDBC, DBI

License MIT + file LICENSE

LazyData true

RoxygenNote 5.0.1

R topics documented:

td *td*

Description

Sends queries to Teradata. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (`conn`) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (`conn`) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (`conn`) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
td(query = "", ...)
```

Arguments

`query`	Query string to send to Teradata.
`...`	Optional connection settings.

Details

Uses the v15.10.00.33 release (12 Jan 2016) tdgssconfig.jar and terajdbc4.jar drivers.

Value

If no data is returned from query, then an `invisible` object is returned. Otherwise, a `data.frame` object with all data queried will be returned.

See Also

`tdConn` for connection, `tdDisk` for disk usage, `tdSpool` for spool usage, `tdCpu` for CPU usage, and `tdJoin` for joining tables.

Examples

```
## NOT RUN ##
## Runs a quick query based on connection profile
# td("select count(*) from ICDB_PERSON", username=<username>, password=<password>, db="GCA")

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>, db="GCA")
# td("select count(*) from ICDB_PERSON", conn=conn)

## Uses same connection, but allows code to find globally
# td("select count(*) from ICDB_PERSON")
```

`tdClose` *tdClose*

Description

Closes connection to the Teradata server. If none exists, then does nothing. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources.

Takes either a connection object provided, or looks for one globally.

Usage

```
tdClose(conn = NULL)
```

Arguments

`conn`	`JDBCConnection` Connection object.

Value

An `invisible` object is returned, indicating success or failure.

See Also

\codetdConn for Teradata connection, `td` for Teradata queries.

Examples

```
## NOT RUN ##
## Connect to default data warehouse and data base
# conn = tdConn(<username>, <password>)

## Close connection
# tdClose()
```

`tdConn` *tdConn*

Description

Checks for a connection to the Teradata server. If none exists, tries to establish one. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. If no JDBC connection is provided (`tdConn`), then a connection is attempted using the `user`, and `password` provided.

Usage

```
tdConn(username = getOption("tdPassword"), password = NULL,
  addr = "jdbc:teradata://megadew.corp.apple.com/charset=utf8",
  db = ifelse(addr == "jdbc:teradata://megadew.corp.apple.com/charset=utf8",
  "CDM_Special", ""), classPath = NULL, conn = NULL)
```

Arguments

`username`	Connection user name.
`password`	Connection password.
`addr`	String containing address of database to connect to. By default, is *jdbc:teradata://megadew.corp.appl*
`db`	Name of database to connect to. By default, is *CDM_Special*.
`classPath`	The location of the JDBC drivers. By default, will use the drivers included in the package.
`conn`	`DBIConnection` object with established connection to the RDMBS. Only used internally to check connection and establish a connection if none exists.

Details

If desired, you can define your username and password in the .Rprofile file using the command `options(tdPassword = c(<username>="<password>"))`, which will then automatically assign the password in the background each time R is started. This then allows you to connect without having to enter your username and password manually each time you connect.

The JDBC driver included uses the v15.10.00.33 release (12 Jan 2016) tdgssconfig.jar and terajdbc4.jar drivers.

Value

A `RJDBC` connection object is returned.

See Also

`td` for Teradata queries, `tdDisk` for disk usage, `tdSpool` for spool usage, `tdCpu` for CPU usage, and `tdJoin` for joining tables.

Examples

```
## NOT RUN ##
## Connect to default data warehouse and data base
# conn = tdConn(<username>, <password>)

## Connect to data warehouse using different data base
# conn = tdConn(<username>, <password>, db='ADM_AMR')

## Connect to different data warehouse than default
# conn = tdConn(<username>, <password>, addr="jdbc:teradata://redwood.corp.apple.com")
```

`tdCpu` *tdCpu*

Description

Queries Teradata for CPU use. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (`conn`) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (`conn`) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (`conn`) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdCpu(user = "user", date = as.Date(Sys.time()), ...)
```

Arguments

`user`	Username to grab CPU use from. Defaults to the `username` used in the Teradata connection.
`date`	Date desired for query information. Defaults to today's date. If overwritten, can be in either the date format for R or the Teradata format YYMMDD.
`...`	Optional connection settings.

Value

A `data.frame` object is returned with the Teradata query information of the specified date.

See Also

`tdConn` for connection, `tdDisk` for disk usage, `tdSpool` for spool usage, and `td` for general queries

Examples

```
## NOT RUN ##
## Connect to default data warehouse and data base
# tdCpu(<username>, <password>)

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>)
# tdCpu(conn=conn)

## Uses same connection, but allows code to find globally
# tdCpu()
```

`tdDim` *tdDim*

Description

Gets the dimensions in Teradata tables. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (`conn`) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (`conn`) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (`conn`) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdDim(table = NULL, where = "", ...)
```

Arguments

`table`	String vector of name of table to get table dimensions from.
`where`	String statement to subset table with.
`...`	Optional connection settings.

Value

Returns a `data.frame` of the the Teradata table dimensions.

See Also

`tdConn` for connection, `tdNames` for table names, `td` for general queries, `tdCpu` for CPU usage, and `tdJoin` for joining tables.

Examples

```
## NOT RUN (will also result in errors due to user restrictions) ##
## Runs a quick query based on connection profile
# tdDim("ICDB_PERSON", username=<username>, password=<password>, db="GCA")

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>, db="GCA")
# tdDim("ICDB_PERSON", conn=conn)

## Uses same connection, but allows code to find globally.
# tdDim("ICDB_PERSON")
```

tdDisk *tdDisk*

Description

Queries Teradata for disk space used. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (`conn`) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (`conn`) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (`conn`) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdDisk(user = "user", ...)
```

Arguments

`user`	Username to grab CPU use from. Defaults to the `username` used in the Teradata connection.
`...`	Optional connection settings.

Value

A `data.frame` object is returned with all of the Teradata query information of the specified date.

See Also

`tdConn` for connection, `tdCpu` for CPU usage, `tdSpool` for spool usage, and `td` for general queries

Examples

```
## NOT RUN ##
## Connect to default data warehouse and data base
# tdDisk(<username>, <password>)

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>)
# tdDisk(conn=conn)

## Uses same connection, but allows code to find globally
# tdDisk()
```

tdDrop *tdDrop*

Description

Drops tables from Teradata. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (`conn`) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (`conn`) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (`conn`) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdDrop(tables = "", ...)
```

Arguments

`tables`	String vector of Teradata tables to drop
`...`	Optional connection settings.

Value

An `invisible` object containing the tables dropped is returned.

See Also

`tdConn` for connection, `tdDisk` for disk usage, `tdSpool` for spool usage, `tdCpu` for CPU usage, and `tdJoin` for joining tables.

Examples

```
## NOT RUN ##
## Runs a quick drop query based on connection profile
# tdDrop(<tableName>, username=<username>, password=<password>, db="GCA")

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>, db="GCA")
# tdDrop(<tableName>, conn=conn)

## Uses same connection, but allows code to find globally.
# Can also drop multiple tables.
# tdDrop(c(<table1Name>, <table2Name>))
```

tdExists *tdExists*

Description

Determines if Teradata table exists. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (`conn`) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (`conn`) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (`conn`) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdExists(table = NULL, ...)
```

Arguments

`table`	String vector of name of table to get column names from.
`...`	Optional connection settings.

Details

This function looks in the `DBC.COLUMNS` table to determine whether the Teradata table exists.

Value

Returns a boolean indicator of either `TRUE` if table exists or `FALSE` if table does not.

See Also

`tdConn` for connection, `tdDisk` for disk usage, `tdSpool` for spool usage, `tdCpu` for CPU usage, and `tdJoin` for joining tables.

Examples

```
## NOT RUN (will also result in errors due to user restrictions) ##
## Runs a quick query based on connection profile
# tdExists("ICDB_PERSON", username=<username>, password=<password>, db="GCA")

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>, db="GCA")
# tdExists("ICDB_PERSON", conn=conn)

## Uses same connection, but allows code to find globally.
# tdExists("ICDB_PERSON")
```

`tdFile` *tdFile*

Description

Submits a SQL file to Teradata to run. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (`conn`) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (`conn`) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (`conn`) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdFile(file = NULL, ...)
```

Arguments

`file`	File to submit to Teradata.
`...`	Optional connection settings.

Details

Warning: This function rads in all lines and parses commands using `";"`. Thus, commands should be separated using that character. If a literal ";" is desired within the code, an escape character of "\" should precede it, e.g. `where column="\;"`.

Value

An `invisible` object is returned indicating whether the file ran successfully.

See Also

`tdConn` for connection, `tdHead` for top observations, `tdSpool` for spool usage, and `td` for general queries

Examples

```
## NOT RUN ##
## Connect to default data warehouse and data base
# tdFile("file.sql", <username>, <password>)

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>)
# tdFile("file.sql", conn=conn)

## Uses same connection, but allows code to find globally
# tdFile("file.sql")
```

tdHead *tdHead*

Description

Gets the top observations in a Teradata table. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (conn) if provided. If no JDBC connection is provided, then a connection is attempted using the user, and password provided. If none is provided, then tries to locate a connection object (conn) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (conn) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdHead(table = NULL, n = 10, cols = NULL, where = "", ...)
```

Arguments

table	A string stating the Teradata table name.
n	A single integer, representing the number of rows desired. Defaults to 10.
cols	Columns desired. Defaults to all columns.
where	String statement to subset table with.
...	Optional connection settings.

Value

Returns a data.frame of the the Teradata table with the first n observations.

See Also

tdConn for connection, tdNames for table names, td for general queries, tdCpu for CPU usage, and tdJoin for joining tables.

Examples

```
## NOT RUN (will also result in errors due to user restrictions) ##
## Runs a quick query based on connection profile
# tdHead("ICDB_PERSON", username=<username>, password=<password>, db="GCA")

## Runs query using a separately established connection. Selects 20 observations
# from only two columns, subset by even Person_Id.
# conn = tdConn(<username>, <password>, db="GCA")
# tdHead("ICDB_PERSON", 20, c("PERSON_ID", "INDIV_ID"), "PERSON_ID mod 2 = 0", conn=conn)

## Uses same connection, but allows code to find globally. Also subsets on PERSON_ID.
# tdHead("ICDB_PERSON")
```

`tdJoin` *tdJoin*

Description

Takes two (or more) Teradata EDW tables using a JDBC connection object via the `RJDBC` package and merges them together. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources.

Usage

```
tdJoin(tdf0, tdf1, tdf2, index1, index2, col1 = NULL, col2 = NULL,
  joinType = "inner", ...)
```

Arguments

`tdf0`	Name of resulting Teradata to output.
`tdf1`	Name of first Teradata table to merge.
`tdf2`	Name of second Teradata table to merge.
`index1`	Name of index from first table to merge by.
`index2`	Name of index from second table to merge by.
`col1`	Name of columns from first table to merge.
`col2`	Name of columns from second table to merge.
`joinType`	Type of merge to perform. Needs to be one of following: `inner,left outer, right outer, full`
`...`	Additional `tdfX` and `indexX` to merge, where `X` is the count. Also can take optional connection settings.

Details

By default, the code tries to do joins starting from Table 1 going up. So if, for example, three tables are provided for inner joins, then Table 1 and Table 2 will first be inner joined, and the resulting output will then be inner joined with Table 3. If a left join is desired for the three tables, then Table 2 will be left joined to Table 1 and Table 3 will then be left joined with the resulting table.

If desired, column names for each table can be provided to merge together. By default, the code will try to use all columns of the tables provided. All tables will be searched for duplicate column names. If any exists, then copies will be renamed with a suffix of `_copyX` where `X` represents the number of copies. If an index name merging by has copies across the tables, then only one index name is kept.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (`conn`) is provided to the function (or one is found globally), then the connection remains open.

Value

The code creates the data table on the Teradata server via the `JDBCConnection` object. Names of each table created are returned as a string vector.

See Also

`tdConn` for connection, `tdDisk` for disk usage, `tdSpool` for spool usage, `tdCpu` for CPU usage, and `td` for general queries.

Examples

```
## NOT RUN ##
## With connection pre-established, inner join on table ##
# conn = tdConn(<username>, <password>)
# tdJoin(<outputTable>, <inputTable1>, <inputTable2>, <index1>, <index2>)

## inner join on table with select columns ##
# tdJoin(<outputTable>, <inputTable1>, <inputTable2>, <index1>, <index2>, joinType="left")

## left join on table ##
# tdJoin(<outputTable>, <inputTable1>, <inputTable2>, <index1>, <index2>, joinType="left")
```

`tdNames` *tdNames*

Description

Gets column names from Teradata tables. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (`conn`) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (`conn`) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (`conn`) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdNames(table = NULL, ...)
```

Arguments

`table`	String vector of name of table to get column names from.
`...`	Optional connection settings.

Details

Many of the core production tables in Teradata are locked, such that trying to query for indicies will result in `Error 3853`. This is a part of the user restrictions and can be circumvented by creating a new subset of the table and querying that subset. If the index is unable to be determined, a value of `NA` will be returned.

Value

Returns a `data.frame` object with the following items:

- "DatabaseName"Database name
- "TableName"Table name
- "ColumnName"Column name
- "ColumnFormat"Column format
- "ColumnType"Column type.
- "ColumnLength"Column length.
- "Index"Indicator of whether column is a primary index (1) or secondary index (2)

See Also

`tdConn` for connection, `tdDisk` for disk usage, `tdSpool` for spool usage, `tdCpu` for CPU usage, and `tdJoin` for joining tables.

Examples

```
## NOT RUN (will also result in errors due to user restrictions) ##
## Runs a quick query based on connection profile
# tdNames("ICDB_PERSON", username=<username>, password=<password>, db="GCA")

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>, db="GCA")
# tdNames("ICDB_PERSON", conn=conn)

## Uses same connection, but allows code to find globally.
# tdNames("ICDB_PERSON")
```

`tdQuantile` *tdQuantile*

Description

Gets column quantiles from a Teradata table. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (conn) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (conn) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (conn) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdQuantile(table = NULL, probs = 0.5, cols = NULL, where = "", ...)
```

Arguments

`table`	A string stating the Teradata table name.
`probs`	Numeric vector of quantiles with values in [0,1]. Defaults to median (i.e. 0.5)
`cols`	Columns desired. Defaults to all columns.
`where`	Statement to subset table with.
`...`	Optional connection settings.

Details

This code is CPU intensive, especially for large data tables, as it requires that the column values be ordered. It is advised to take care when implementing, as user limits may prevent the code from sucessfully running. If CPU or spool limits are reached, a workaround could be implemented by first breaking the data table into smaller subsets and subsequently taking the percentiles over them.

The code is really meant for numeric valued columns. If string columns are provided, the code will still run. However, the results will be less interpretable.

Value

Returns a `data.frame` of the the Teradata table with the quantiles specified.

See Also

`tdConn` for connection, `tdNames` for table names, `td` for general queries, `tdCpu` for CPU usage, and `tdHead` for top rows in table.

Examples

```
## NOT RUN (will also result in errors due to user restrictions) ##
## Runs a quick query based on connection profile
# tdQuantiles("ICDB_PERSON", username=<username>, password=<password>, db="GCA")

## Runs query using a separately established connection. Selects only two columns.
# conn = tdConn(<username>, <password>, db="GCA")
# tdQuantilesy("ICDB_PERSON", c("PERSON_ID", "INDIV_ID"), conn=conn)

## Uses same connection, but allows code to find globally. Also subsets on PERSON_ID.
# tdQuantiles("ICDB_PERSON", where="PERSON_ID mod 2 = 0")
```

`tdSample` *tdSample*

Description

Gets random observations in a Teradata table. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (conn) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (conn) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (conn) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdSample(table = NULL, n = 10, cols = NULL, where = "", ...)
```

Arguments

`table`	A string stating the Teradata table name.
`n`	A single integer, representing the number of rows desired. Defaults to 10.
`cols`	Columns desired. Defaults to all columns.
`where`	String statement to subset table with.
`...`	Optional connection settings.

Details

Whereas this fucntion grabs a random sample, `tdHead` will grab the top observations in the table. Thus, this function will be more CPU intensive than `tdHead`.

Value

Returns a `data.frame` of the the Teradata table with the first `n` observations.

See Also

`tdConn` for connection, `tdNames` for table names, `td` for general queries, `tdHead` for top observations usage, and `tdJoin` for joining tables.

Examples

```
## NOT RUN (will also result in errors due to user restrictions) ##
## Runs a quick query based on connection profile
# tdSample("ICDB_PERSON", username=<username>, password=<password>, db="GCA")

## Runs query using a separately established connection. Selects 20 observations
# from only two columns, subset by even Person_Id.
# conn = tdConn(<username>, <password>, db="GCA")
# tdSample("ICDB_PERSON", 20, c("PERSON_ID", "INDIV_ID"), "PERSON_ID mod 2 = 0", conn=conn)

## Uses same connection, but allows code to find globally. Also subsets on PERSON_ID.
# tdSample("ICDB_PERSON")
```

`tdShow` *tdShow*

Description

Gets the Teradata code used to create the table. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (conn) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (conn) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (conn) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdShow(table = NULL, ...)
```

Arguments

`table`	String of name of table to get Teradata code from.
`...`	Optional connection settings.

Details

Many of the core production tables in Teradata are locked, such that trying to query to show table code will result in `Error 3853`. This is a part of the user restrictions and can be circumvented by creating a new subset of the table and querying that subset. If the Teradata code is unable to be determined, a value of `NA` will be returned.

Value

Returns a string `vector` of the the Teradata code.

See Also

`tdConn` for connection, `tdDisk` for disk usage, `tdSpool` for spool usage, `tdCpu` for CPU usage, and `tdJoin` for joining tables.

Examples

```
## NOT RUN (will also result in errors due to user restrictions) ##
## Runs a quick query based on connection profile
# tdShow("ICDB_PERSON", username=<username>, password=<password>, db="GCA")

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>, db="GCA")
# tdShow("ICDB_PERSON", conn=conn)

## Uses same connection, but allows code to find globally. Also used for multiple tables.
# tdShow(c("ICDB_PERSON", "ICDB_PERSON_X"))
```

`tdSpool` *tdSpool*

Description

Queries Teradata for spool use. This code is specifically designed for connectivity to Teradata servers using OSX at Apple using JDBC drivers and should be updated if connected to other sources. Can take a JDBC connection object (conn) if provided. If no JDBC connection is provided, then a connection is attempted using the `user`, and `password` provided. If none is provided, then tries to locate a connection object (conn) in the global environment.

If a connection profile (e.g. username, password, etc.) is provided, then an attempt is made to connect to Teradata. Once the query is run, the connection is then closed. If a connection object (conn) is provided to the function (or one is found globally), then the connection remains open.

Usage

```
tdSpool(user = "user", ...)
```

Arguments

`user`	Username to grab CPU use from. Defaults to the `username` used in the Teradata connection.
`...`	Optional connection settings.

Value

A `data.frame` object is returned with all of the Teradata spool information.

Examples

```
## NOT RUN ##
## Connect to default data warehouse and data base
# tdSpool(<username>, <password>)

## Runs query using a separately established connection
# conn = tdConn(<username>, <password>)
# tdSpool(conn=conn)

## Uses same connection, but allows code to find globally
# tdSpool()
```

Index