Package 'RHive'

December 22, 2014

Type Package

Title R and Hive

Version 2.0-0.2

Description RHive is an R extension facilitating
distributed computing via HIVE query.
It provides an easy to use HQL like SQL
and R objects and functions in HQL.

Author NexR

Maintainer Bruce Shin <bruce.shin@nexr.com>

License Apache License (== 2.0)

Depends R (>= 2.13.0), rJava (>= 0.9-0)

SystemRequirements Hadoop core >= 0.20.3
(http://hadoop.apache.org/core/), Hive >= 0.8
(http://hive.apache.org/)

OS_type unix

NeedsCompilation no

Repository CRAN

Date/Publication 2014-09-30 07:48:49

R topics documented:

`rhive` *rhive: R and Hive*

Description

RHive is an R extension facilitating distributed computing via HIVE query. It allows easy usage of HQL in R, and allows easy usage of R objects and R functions in Hive.

Details

The RHive package supplies functions to interact with Hive from within R. There are functions for exporting and connecting as well as querying hive.

Author(s)

<rhive@nexr.com>

See Also

rhive.init rhive.connect rhive.set rhive.unset rhive.query rhive.execute rhive.big.query rhive.assign rhive.rm rhive.export rhive.exportAll rhive.list.udfs rhive.rm.udf rhive.script.export rhive.script.unexport rhive.close rhive.list.databases rhive.show.databases rhive.use.database rhive.list.tables rhive.show.tables rhive.desc.table rhive.load.table rhive.exist.table rhive.size.table rhive.drop.table rhive.napply rhive.sapply rhive.aggregate rhive.load rhive.save rhive.sample rhive.mrapply rhive.mapapply rhive.reduceapply rhive.hdfs.connect rhive.hdfs.ls rhive.hdfs.get rhive.hdfs.put rhive.hdfs.rm rhive.hdfs.rename rhive.hdfs.exists rhive.hdfs.mkdirs rhive.hdfs.cat rhive.hdfs.tail rhive.hdfs.du rhive.hdfs.close rhive.hdfs.info rhive.hdfs.chmod rhive.hdfs.chown rhive.hdfs.chgrp rhive.basic.mode rhive.basic.range rhive.basic.merge rhive.basic.xtabs rhive.basic.cut rhive.basic.cut2 rhive.basic.by rhive.basic.scale rhive.basic.t.test rhive.block.sample rhive.write.table hiveConnect hiveSet hiveQuery hiveAssign hiveRm hiveExport hiveExportAll hiveClose hiveListDatabases hiveListTables hiveDescTable hiveLoadTable

---

`rhive-api` *R functions to get informations of table from HIVE*

Description

R functions to get informations of table from HIVE

Usage

```
rhive.list.databases(pattern)
rhive.show.databases(pattern)
rhive.use.database(databaseName)
rhive.list.tables(pattern)
rhive.show.tables(pattern)
rhive.desc.table(tableName, detail=FALSE)
rhive.load.table(tableName, fetchSize=50, limit=-1)
rhive.load.table2(tableName, limit=-1, remote=TRUE)
rhive.exist.table(tableName)
rhive.size.table(tableName)
rhive.drop.table(tableName, list)
rhive.set(key, value)
rhive.unset(key)
```

Arguments

databaseName	hive database name.
tableName	hive table name.
remote	hiveserver mode.
detail	a flag on whether to show detail of table info.
limit	total fetch size. -1 means full fetch
fetchSize	the count of record to load at one time
pattern	an optional regular expression. Only names matching 'pattern' are returned. 'glob2rx' can be used to convert wildcard patterns to regular expressions.
list	a character vector naming tables to be removed. or rhive.list.tables's result.
key	hive configuration key
value	hive configuration value

Author(s)

<rhive@nexr.com>

Examples

```
## try to connect hive server
## Not run: rhive.connect("hive-server-ip")

## get list of databases in the Hive
## Not run: rhive.list.databases()

## set current database
## Not run: rhive.use.database('default')

## get list of tables in the Hive
## Not run: rhive.list.tables()
```

```
## get table info in the Hive
## Not run: rhive.desc.table('emp')

## get detail information of a table in the Hive
## Not run: rhive.desc.table('emp', TRUE)

## retrieve data from hive
## Not run: emp <- rhive.load.table('emp')

## display column names
## Not run: colnames(emp)

## display row count
## Not run: length(rownames(emp))

## close connection
## Not run: rhive.close()
```

---

rhive-apply *R Distributed apply function using HQL*

Description

R Distributed apply function using HQL

Usage

```
rhive.napply(tableName, FUN, ...,forcedRef=TRUE)
rhive.sapply(tableName, FUN, ..., forcedRef=TRUE)
rhive.mrapply(tableName, mapperFUN, reducerFUN, mapInput=NULL,
  mapOutput=NULL, by=NULL, reduceInput=NULL,reduceOutput=NULL,
  mapperArgs=NULL, reducerArgs=NULL, bufferSize=-1L, verbose=FALSE,
  forcedRef=TRUE)
rhive.mapapply(tableName, mapperFUN, mapInput=NULL, mapOutput=NULL,
  by=NULL, args=NULL, bufferSize=-1L, verbose=FALSE, forcedRef=TRUE)
rhive.reduceapply(tableName, reducerFUN, reduceInput=NULL,
  reduceOutput=NULL, args=NULL, bufferSize=-1L, verbose=FALSE,
  forcedRef=TRUE)
```

Arguments

tableName	hive table name.
FUN	the function to be applied.
...	optional arguments to 'FUN'.
mapperFUN	a function which is executed on each worker node. The so-called mapper typically maps input key/value pairs to a set of intermediate key/value pairs.

reducerFUN	a function which is executed on each worker node. The so-called reducer reduces a set of intermediate values which share a key to a smaller set of values. If no reducer is used leave NULL.
mapInput	map-input column list.
mapOutput	map-output column list.
by	cluster key column
reduceInput	reduce-input column list.
reduceOutput	reduce-output column list.
bufferSize	streaming buffer size.
verbose	print generated HQL.
args	custom environment.
mapperArgs	mapper custom environment.
reducerArgs	reducer custom environment.
forcedRef	the option which forces to create temp-table for result.

Author(s)

<rhive@nexr.com>

Examples

```
## try to connect hive server
## Not run: rhive.connect("hive-server-ip")

## invoke napply for numeric return type
## Not run: rhive.napply('emp', function(item) {
item * 10
},'sal')
## End(Not run)

## invoke sapply for string return type
## Not run: rhive.napply('emp', function(item) {
paste('NAME : ', item, sep='')
}, 'ename')
## End(Not run)

## custom map/reduce script
## Not run: map <- function(k, v) {
    if(is.null(v)) {
        put(NA, 1)
    }
    lapply(v, function(vv) {
        lapply(strsplit(x = vv, split = "\t")[[1]],
            function(w) put(paste(args, w, sep = ""), 1))
    })
}

reduce <- function(k, vv) {
```

```
    put(k, sum(as.numeric(vv)))
}

rhive.mrapply("emp", map, reduce, c("ename", "position"), c("position", "one"),
    by="position", c("position", "one"), c("position", "count"))
## End(Not run)


## close connection
## Not run: rhive.close()
```

---

rhive-connect *Manage connection to Hive using functions in Package 'RHive'*

Description

Manage connection to Hive using functions in Package 'RHive'

Usage

```
rhive.init(hiveHome=NULL, hiveLib=NULL, hadoopHome=NULL, hadoopConf=NULL,
  hadoopLib=NULL, verbose=FALSE)
rhive.env(ALL=FALSE)
rhive.connect(host="127.0.0.1",port=10000, hiveServer2=NA, defaultFS=NULL,
  updateJar=FALSE, user=NULL, password=NULL)
rhive.close()
```

Arguments

hiveHome	path of hive's installation or if HIVE_HOME is set, it is possible to use as NULL.
hadoopHome	path of hadoop's installation or if HADOOP_HOME is set, it is possible to use as NULL.
hadoopConf	path of hadoop's configuation or if HADOOP_CONF_DIR is set, it is possible to use as NULL.
hiveLib	library path to be added to classpath.
hadoopLib	hadoop library path to be added to classpath.
host	hive-server address for connecting to hive.
port	hive-server listen port.
hiveServer2	TRUE if you are using HiveServer2 and FALSE otherwise.
defaultFS	the url of hdfs namenode.
updateJar	update rhive_udf.jar
user	the username for the query to run as.
password	the user's password
verbose	an option on whether to print detail message.
ALL	show all rhive enviroment, such as classpath.

Author(s)

<rhive@nexr.com>

Examples

```
## initialize rhive
## Not run: rhive.init()

## try to connect hive server
## Not run: rhive.connect("127.0.0.1")

## close connection
## Not run: rhive.close()
```

---

rhive-execute *Execute HQL(Hive Query) in R, using functions in Package 'RHive'*

Description

Execute HQL(Hive Query) in R using functions in Package 'RHive'

Usage

```
rhive.execute(query)
```

Arguments

query	hive query.

Details

The query argument is Hive Query Language.

Value

rhive.execute returns TRUE on success.

Author(s)

<rhive@nexr.com>

References

Apache Hive Query Language Manual (https://cwiki.apache.org/confluence/display/Hive/LanguageManual).

Examples

```
## try to connect hive server
## Not run: rhive.connect("127.0.0.1")

## execute hive query
## Not run: rhive.execute("create table emp (empno int,
                                             ename string,
                                             position string,
                                             col int,
                                             fday string,
                                             sal double,
                                             status string,
                                             deptno int)")
## End(Not run)

## close connection
## Not run: rhive.close()
```

---

rhive-export *Export R function to Hive using functions in Package 'RHive'*

Description

Export R function to Hive using functions in Package 'RHive'

Usage

```
rhive.export(exportName, pos=-1, limit=100*1024*1024, ALL=FALSE)
rhive.exportAll(exportName, pos=1, limit=100*1024*1024)
rhive.assign(name, value)
rhive.assign.export(name, value)
rhive.rm(name)
rhive.rm.export(name)
rhive.script.export(exportName, mapper=NULL, reducer=NULL, mapArgs=NULL,
  reduceArgs=NULL, bufferSize=-1L)
rhive.script.unexport(exportName)
rhive.export.script(exportName, mapper=NULL, reducer=NULL, mapArgs=NULL,
  reduceArgs=NULL, bufferSize=-1L)
rhive.unexport.script(exportName)
rhive.list.udfs()
rhive.rm.udf(exportName)
```

Arguments

exportName	function name to be exported.
limit	total exported object size. default is 100MB
ALL	export all objects

name	a variable name, given as a character string.
value	a value to be assigned to 'name'
pos	where to do the assignment.
mapper	R object as map function or Hive query.
reducer	R object as reducer function.
bufferSize	streaming buffer size.
mapArgs	mapper custom environment.
reduceArgs	reducer custom environment.

Details

RHive supports the following additional Hive functions. One is RUDF and its syntax is `R(export-R-function-name, argum`

Another is RUDAF and its syntax is `RA(export-R-function-name, arguments, ...)`. R function which runs via RUDAF should be made with the following rule. This rule is a function naming rule. An R aggregation function is composed of 4 sub-functions and each sub-function has a naming rule. First sub-function uses user-defined name, which is export-R-function-name. Second is made from combining first sub-function name and '.partial'. Third is made from combining first function name and '.merge'. Final function is made from combining first name and '.terminate'.

UDTF is a built-in table-generating function in Hive. RHive supports two kinds of UDTF, unfold and expand. 'unfold' syntax is `unfold(value,col1-v,col2-v,...,delim) as (col1,col2,...)`. this 'unfold' function allows user to change one column into many columns. 'expand' syntax is `expand(value,col-v,delim) as(col)`. this 'expand' function allows user to change one column into many rows.

Author(s)

<rhive@nexr.com>

Examples

```
## try to connect hive server
## Not run: rhive.connect("127.0.0.1")

## execute HQL(hive query)
## Not run: rhive.query("select * from emp")


## define R function
## Not run: coff <- 5.2
## Not run: scoring <- function(sal) {
    coff * sal
}
## End(Not run)

## assign R object to Hive
## Not run: rhive.assign('scoring', scoring)
## Not run: rhive.assign('coff', coff)
```

```
## export R objects (scoring and coff) to Hive
## Not run: rhive.exportAll('scoring')

## execute HQL using exported R objects
## name of UDF is 'R'
## Not run: rhive.query("select R('scoring',sal,0.0) from emp")

## delete R object in .rhiveExportEnv
## Not run: rhive.rm('scoring')
## Not run: rhive.rm('coff')

## define R aggregation function
## define iterate operator
## Not run: hsum <- function(prev, sal) {
    if(is.null(prev))
        sal
    else
        prev + sal
}
## End(Not run)
## define partial aggregation operator
## Not run: hsum.partial <- function(agg_sal) {
agg_sal
}
## End(Not run)
## define merge operator
## Not run: hsum.merge <- function(prev, agg_sal) {
    if(is.null(prev))
        agg_sal
    else
        prev + agg_sal
}
## End(Not run)
## define final aggregation operator
## Not run: hsum.terminate <- function(agg_sal) {
    agg_sal
}
## End(Not run)

## Not run: rhive.assign('hsum', hsum)
## Not run: rhive.assign('hsum.partial', hsum.partial)
## Not run: rhive.assign('hsum.merge', hsum.merge)
## Not run: rhive.assign('hsum.terminate', hsum.terminate)
## Not run: rhive.exportAll('hsum')

## name of UDAF is 'RA'
## Not run: rhive.query("select RA('hsum',sal) from emp group by empno")


## export/unexport user define map/reduce script
## Not run:
map <- function(k, v) {
    if(is.null(v)) {
```

```
        put(NA,1)
    }
    lapply(v, function(vv) {
        lapply(strsplit(x=vv, split = "\t")[[1]],
            function(w) put(paste(args, w, sep = ""), 1))
    })
}

reduce <- function(k,vv) {
    put(k, sum(as.numeric(vv)))
}

mrscript <- rhive.script.export("scripttest", map, reduce)


rhive.query(paste("from (from emp MAP ename,position USING '", mrscript[1],
    "' as position, one cluster by position) map_output REDUCE map_output.aa,
map_output.bb USING '",
    mrscript[2], "' as position, count", sep = ""))


## End(Not run)

## close connection
## Not run: rhive.close()
```

---

rhive-fn *Methods for the class*

Description

These functions are wrappers to provide function style api for rhive functions.

Author(s)

<rhive@nexr.com>

---

rhive-hdfs *R functions to communicate with HDFS*

Description

R functions to communicate with HDFS

Usage

```
rhive.save(..., file, envir=parent.frame())
rhive.load(file, envir=parent.frame())
rhive.hdfs.ls(path="/")
rhive.hdfs.get(src, dst, srcDel=FALSE)
rhive.hdfs.put(src, dst, srcDel=FALSE, overwrite=FALSE)
rhive.hdfs.rm(...)
rhive.hdfs.rename(src, dst)
rhive.hdfs.exists(path)
rhive.hdfs.mkdirs(path)
rhive.hdfs.cat(path)
rhive.hdfs.tail(path)
rhive.hdfs.du(path="/", summary=FALSE)
rhive.hdfs.dus(path="/")
rhive.write.table(data, tableName, sep=",", naString=NULL, rowName=FALSE,
  rowNameColumn="rowname")
rhive.hdfs.info(path)
rhive.hdfs.chmod(option, path, recursive=FALSE)
rhive.hdfs.chown(option, path, recursive=FALSE)
rhive.hdfs.chgrp(option, path, recursive=FALSE)
```

Arguments

src	full path of source data.
dst	full path of target data.
file	the full-name of the file where the data will be saved or loaded
path	hdfs's full path.
envir	environment to search for objects to be saved or loaded.
srcDel	indicates if the source should be removed.
overwrite	if path exists,this option indicates whether to overwrite.
...	target path list.
data	the object to be written, preferably a data frame.
tableName	a character string naming a table
sep	the field separator string. Values within each row of 'data' are separated by this string
naString	default value for NA.
rowName	a logical value indicating whether the row names of 'data' are to be written along with data
rowNameColumn	a character string specifying the column which contains the row names of 'data'
summary	summarize result of 'du'.
option	specific option. chmod's option is 775 or chown's option is user-id.
recursive	apply command recursively

Details

rhive.hdfs.connect : Connect to HDFS

rhive.hdfs.ls : Lists the contents of the directory specified by path, showing the names, permissions, owner, size and modification date for each entry.

rhive.hdfs.put : Copy the file or directory from the local file system identified by source to target within the HDFS.

rhive.hdfs.get : Copy the file or directory in HDFS identified by source to the local file system path identified by target.

rhive.hdfs.rm : Removes the file or empty directory identified by path.

rhive.hdfs.rename : Rename the file or directory identified by source to target within the HDFS.

rhive.hdfs.exists : Check whether the file or directory specified by path is or not.

rhive.hdfs.mkdirs : Creates a directory named path in HDFS.

rhive.hdfs.close : Close hdfs connection

rhive.save : save R Objects to HDFS as R data format

rhive.load : load R data format file stored in HDFS

rhive.write.table : create Hive table using R data.frame

rhive.hdfs.info : report block information of path

rhive.hdfs.chmod : change mode for specified path.

rhive.hdfs.chown : change ownership for specified path.

rhive.hdfs.chgrp : change group for specified path.

Author(s)

<rhive@nexr.com>

Examples

```
## try to connect hdfs namenode
## Not run: rhive.hdfs.connect()

## get list of specified path
## Not run: rhive.hdfs.ls()

## load local-file to hdfs
## Not run: rhive.hdfs.put('/data/rhive.txt','/rhive/data/load.txt)

## download data from hdfs to local-file
## Not run: rhive.hdfs.get('/rhive/data/load.txt','/data/rhive.txt')

## delete data in hdfs
## Not run: rhive.hdfs.rm('/rhive/data/load.txt')

## close connection
## Not run: rhive.hdfs.close()
```

rhive-query *Execute HQL(Hive Query) in R, using functions in Package 'RHive'*

Description

Execute HQL(Hive Query) in R using functions in Package 'RHive'

Usage

```
rhive.query(query, fetchSize=50, limit=-1)
rhive.big.query(query ,fetchSize=50, limit=-1, memLimit=64*1024*1024)
```

Arguments

query	hive query.
fetchSize	fetch size for result.
limit	total result size.
memLimit	the size of data can be handled in R's memory.

Details

The query argument is Hive Query Language.

Value

rhive.query returns data.frame.

Author(s)

<rhive@nexr.com>

References

Apache Hive Query Language Manual (https://cwiki.apache.org/confluence/display/Hive/LanguageManual).

Examples

```
## try to connect hive server
## Not run: rhive.connect("127.0.0.1")

## execute hive query
## Not run: rhive.query("select ename from emp")

## close connection
## Not run: rhive.close()
```

rhive.aggregate *R Distributed aggregate function using HQL*

Description

R Distributed aggregate function using HQL

Usage

```
rhive.aggregate(tableName, hiveFUN, ..., groups=NULL,
  forcedRef=TRUE)
```

Arguments

tableName	hive table name.
hiveFUN	the hive buit-in function name to be applied.
...	optional arguments to 'hiveFUN'.
groups	aggregated key list. it is vector type
forcedRef	the option which forces to create temp-table for result.

Author(s)

<rhive@nexr.com>

Examples

```
## try to connect hive server
## Not run: rhive.connect("hive-server-ip")

## invoke napply for numeric return type
## Not run: rhive.aggregate('emp', 'sum', 'sal', c('ename'))

## close connection
## Not run: rhive.close()
```

rhive.basic *R Distributed basic statistic function using Hive*

Description

R Distributed basic statistic function using Hive

Usage

```
rhive.basic.mode(tableName, col, forcedRef=TRUE)
rhive.basic.range(tableName, col)
rhive.basic.merge(x, y, by.x, by.y, forcedRef=TRUE)
rhive.basic.xtabs(formula, tableName)
rhive.basic.cut(tableName, col, breaks, right=TRUE, summary=FALSE,
  forcedRef=TRUE)
rhive.basic.cut2(tableName, col1, col2, breaks1, breaks2, right=TRUE,
  keepCol=FALSE, forcedRef=TRUE)
rhive.basic.by(tableName, INDICES, fun, arguments, forcedRef=TRUE)
rhive.basic.scale(tableName, col)
rhive.basic.t.test(x,col1,y,col2)
rhive.block.sample(tableName, percent=0.01, seed=0, subset)
```

Arguments

tableName	hive table name.
x, y	table-names to be coerced to one or an object which can be coerced.
by.x, by.y	specifications of the common columns.
col	column name
col1	column name
col2	column name
formula	a formula object with the cross-classifying variables (separated by '+') on the right hand side (or an object which can be coerced to a formula).
breaks	a numeric vector of two or more cut points. a format is 'min:max:step' and 'step' is optional. or either a numeric vector of two or more cut points or a single number (greater than or equal to 2) giving the number of intervals into which 'x' is to be cut.
breaks1	a breaks of col1
breaks2	a breaks of col2
summary	a option whether summarize the result of cut or not.
INDICES	a list of column to be grouped.
fun	a hive function name to be applied.
arguments	input data for a function. for examples, arguments = c("sal", "deptno", 3.2, "'NexR'")
right	logical, indicating if the intervals should be closed on the right (and open on the left) or vice versa.
keepCol	an option which keeps original columns
forcedRef	the option which forces to create temp-table for result.
percent	percent of data size which is picked up.
seed	first selected block index.
subset	an optional record-set specifying a subset of observations to be used.

Author(s)

<rhive@nexr.com>

Examples

```
## try to connect hive server
## Not run: rhive.connect("hive-server-ip")

## find the most frequency data of specified column
## Not run: rhive.basic.mode('emp','deptno')

## calculate min,max of specified column
## Not run: rhive.basic.range('emp','sal')

## merge two tables using shared column
## Not run: rhive.basic.merge('emp','dept', by.x = 'deptno', by.y = 'id')

DF <- as.data.frame(UCBAdmissions)

## Not run: rhive.write.table(DF)

## Nice for taking margins ...
## Not run: rhive.basic.xtabs('freq', c('gender', 'admit'), 'df')

## divides the range of a column into intervals
## Not run: rhive.basic.cut('emp', 'sal', breaks='0:5000:100')

## divides the range of a column into intervals
## Not run: rhive.basic.cut2('emp', 'dept', 'sal', 'loc', breaks1='0:5000:100',
  breaks2='0:100:10')
## End(Not run)

## extract the summation of salary by group
## Not run: rhive.basic.by('emp', 'deptno', 'sum', c("sal"))

## centers and/or scales the columns of table
## Not run: rhive.basic.scale('emp', 'sal')

## analyze two dataset
## Not run: rhive.basic.t.test(emp$sal, emp$age)

## sampling
## Not run: rhive.basic.sample("emp", subset="id < 100")

## close connection
## Not run: rhive.close()
```

Index