IRIS DB connect

```
library(RJDBC)
```

```
## Loading required package: DBI
```

```
## Loading required package: rJava
```

```
.jinit()
.jaddClassPath("/usr/share/R/library/jettison-1.3.2.jar")
.jaddClassPath("/usr/share/R/library/log4j-1.2.17.jar")
.jaddClassPath("/docker/tools/Spark-on-IRIS/lib/java/mobigen-iris-jdbc-2.1.0.1.jar")

print(.jclassPath())
```

```
## [1] "/usr/lib64/R/library/rJava/java"
## [2] "/docker/tools/Spark-on-IRIS/lib/java/mobigen-iris-jdbc-2.1.0.1.jar"
```

```
drv <- RJDBC::JDBC("com.mobigen.iris.jdbc.IRISDriver",
                   "/docker/tools/Spark-on-IRIS/lib/java/mobigen-iris-jdbc-2.1.0.1.ja
r",
                   identifier.quote="`")
conn <- RJDBC::dbConnect(drv, "jdbc:iris://192.168.100.180:5050/myiris", "myiris", "m
yiris")
```

GLOBAL TABLE :

CREATE(dbSendUpdate) & INSERT(dbSendUpdate) & SELECT(dbGetQuery)

```
# 테이블이 있다면 DROP 하고 생성(주의할 것 !!!)
dbSendUpdate(conn, 'DROP TABLE IF EXISTS MYIRIS.IRIS_GLOBAL_TEST_1 ;')

sql_create <-  "CREATE TABLE IRIS_GLOBAL_TEST_1 (
                  irisid INTEGER,
                  sepal_length REAL,
                  sepal_width REAL,
                  petal_length REAL,
                  petal_width REAL,
                  species TEXT)
                  datascope       GLOBAL
                  ramexpire       0
                  diskexpire      0
                  partitionkey    None
                  partitiondate   None
                  partitionrange  0 ; "

# CREATE GLOBAL TABLE : dbSendUpdate 를 이용하며, SQL 문 끝에 ; 를 넣어야 한다. return값은 없음
dbSendUpdate(conn, sql_create)
```

```
# table 이 생성되었는지 확인하기
table_list <- dbGetQuery(conn, "table list")
#print(table_list)
```

INSERT into GLOBAL TABLE

```
ins_sql <- sprintf( "INSERT INTO IRIS_GLOBAL_TEST_1 (irisid, sepal_length,sepal_widt
h, petal_length,petal_width,  species) VALUES (1, 1.0, 2.0, 3.0, 4.0, 'test') ; ")

dbSendUpdate(conn, ins_sql)

# SELECT from GLOBAL TABLE
result2 <- dbGetQuery(conn, 'select * from IRIS_GLOBAL_TEST_1')
print(result2)
```

```
##   IRISID SEPAL_LENGTH SEPAL_WIDTH PETAL_LENGTH PETAL_WIDTH SPECIES
## 1      1          1.0         2.0          3.0         4.0    test
```

```
# DB DISCONNECT

dbDisconnect(conn)
```

```
## [1] TRUE
```

LOCAL TABLE CASE

대용량 테이블이므로 가져오는 데이터양을 적절하게 조정해야 한다.

```
.jinit()
```

```
## [1] 0
```

```
.jaddClassPath("/usr/share/R/library/jettison-1.3.2.jar")
.jaddClassPath("/usr/share/R/library/log4j-1.2.17.jar")
.jaddClassPath("/docker/tools/Spark-on-IRIS/lib/java/mobigen-iris-jdbc-2.1.0.1.jar")

print(.jclassPath())
```

```
## [1] "/usr/lib64/R/library/rJava/java"
## [2] "/docker/tools/Spark-on-IRIS/lib/java/mobigen-iris-jdbc-2.1.0.1.jar"
## [3] "/usr/lib64/R/library/RJDBC/java/RJDBC.jar"
```

```
drv <- RJDBC::JDBC("com.mobigen.iris.jdbc.IRISDriver",
                   "/docker/tools/Spark-on-IRIS/lib/java/mobigen-iris-jdbc-2.1.0.1.ja
r",
                   identifier.quote="`")
conn <- RJDBC::dbConnect(drv, "jdbc:iris://192.168.100.180:5050/myiris", "myiris", "m
yiris")
```

LOCAL TABLE :

- partiton range = 60min (60분으로 파티션 범위를 정함)
- partition gubun = DATETIME (partition 구분기준 컬럼이름. 반드시 해당 시간필드를 YYYYMMDDHHMMSS 14 자리 text 형식으로 변환해 놓아야 한다)
- partition키= HOST (partition key)

```
# SELECT from LOCAL TABLE EVA.SYSLOG
# 2019-11-12 15:00:00 ~ 16:59:59 ( 20191112150000, 20191112160000 2개의 파티션에서 데이터를 가지고 온다)
# count = 89887

select_sql <- "/*+ LOCATION ( PARTITION >= '20191112150000' AND PARTITION <= '20191112160000' ) */
SELECT
    DATETIME, HOST, FACILITY, PRIORITY, LEVEL, LEVEL_INT, TAG, PROGRAM
FROM
    EVA.SYSLOG
; "

my_dataframe <- data.frame()
rs <- dbSendQuery(conn, select_sql)  # dbSendQuery !!!! ( dbGetQuery 아님 )

nn = 1000   #  1000 개 단위로 fetch
tmp_df <- data.frame()

tmp_df <- dbFetch(rs, n=nn)
my_dataframe <- tmp_df

while ( nrow(tmp_df) ==  nn ) {
  tmp_df <- dbFetch(rs, n=nn)
  my_dataframe <- rbind(my_dataframe, tmp_df)
}
dbClearResult(rs)
```

```
## [1] TRUE
```

```
print(nrow(tmp_df))   # 마지막 fetch 레코드 수 < nn
```

```
## [1] 887
```

```
# select 한 전체 레코드 수
print(nrow(my_dataframe))
```

```
## [1] 89887
```

IRIS DB 에 LOCAL TABLE 생성하기

```
# CREATE TABLE : LOCAL IRIS Table

# 테이블이 있다면 DROP 하고 생성(주의할 것 !!!)
dbSendUpdate(conn, 'DROP TABLE IF EXISTS MYIRIS.IRIS_LOCAL_TEST_2 ;')



cr_table_sql <- 'CREATE TABLE IRIS_LOCAL_TEST_2 (
   DATETIME     TEXT,
   HOST         TEXT,
   FACILITY     TEXT,
   PRIORITY     TEXT,
   LEVEL        TEXT,
   LEVEL_INT    TEXT,
   TAG          TEXT,
   PROGRAM      TEXT )
datascope       LOCAL
ramexpire       60
diskexpire      2102400
partitionkey    HOST
partitiondate   DATETIME
partitionrange  60
; '

dbSendUpdate(conn, cr_table_sql)
```

```
# table 이 생성되었는지 확인하기
table_list <- dbGetQuery(conn, "table list")
#print(table_list)
```

R dataframe 을 IRIS DB 에 입력하기

- my_dataframe(총 89887 건) : 1000 건을 한번에 insert 하는 예제

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
table_name <- 'MYIRIS.IRIS_LOCAL_TEST_2'

insert_batch_sql_f <- function(conn, table, df) {
  batch <- apply(df, 1, FUN = function(x) paste0("'",trimws(x),"'",collapse = ",")) %
>% paste0("(",.,")",collapse = ", ")

  colums <-  paste(unlist(colnames(df)), collapse=',')
  query <- paste("INSERT INTO ", table, "(", colums, ") VALUES ", batch, ';')

  dbSendUpdate(conn, query)
}

insert_batch_sql_f(conn, table_name, my_dataframe[1:1000, ])
# 1건씩 인서트는 인서트 sql 을 만들어서 dbSendUpdate(conn, query)

my_count <- dbGetQuery(conn,"select count(*) from IRIS_LOCAL_TEST_2")
print(my_count)
```

```
##   COUNT(*)
## 1     1000
```

```
dbDisconnect(conn)
```

```
## [1] TRUE
```