week_13_assignment_join_performance

Prashant B. Bhuyan

December 10, 2014

Problem 1.

```
# Add relevant libraries
library(DBI)
library(RPostgreSQL)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
##
## The following object is masked from 'package:stats':
##
##     filter
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(reshape2)
library(microbenchmark)
library(nycflights13)

# Connect to postgreSQL database
drv <- dbDriver("PostgreSQL")
con <- dbConnect(drv, dbname = "postgres", host = "localhost", port = 5432, user = "postgres", password

# Set working directory
setwd("/Users/pbmrt/Downloads/")

# Read in airport and flight data into respective data frames
airport_data <- read.csv("neo4j-airport-csv-raw(1).csv")
flight_data <- read.csv("neo4j-flight-lab-data.csv")



# Write data as tables into postgresql db
dbWriteTable(con, "airport_data", airport_data)
```

```
## Warning: table airport_data exists in database: aborting assignTable
```

```
## [1] FALSE
```

```
dbWriteTable(con, "flight_data", flight_data)
```

```
## Warning: table flight_data exists in database: aborting assignTable
```

```
## [1] FALSE
```

```
# Create arrive and depart tables to do more complex join later on
airport_labels <- dbGetQuery(con, "select label from airport_data")
departures <- dbGetQuery(con, "select flight, airline, depart, capacity, takeoff from flight_data")
arrivals <- dbGetQuery(con, "select flight, airline, arrive, capacity, landing from flight_data")

# Write new tables to db
dbWriteTable(con, "airport_labels", airport_labels)
```

```
## Warning: table airport_labels exists in database: aborting assignTable
```

```
## [1] FALSE
```

```
dbWriteTable(con, "departures", departures)
```

```
## Warning: table departures exists in database: aborting assignTable
```

```
## [1] FALSE
```

```
dbWriteTable(con, "arrivals", arrivals)
```

```
## Warning: table arrivals exists in database: aborting assignTable
```

```
## [1] FALSE
```

```
# join departures and arrivals with labels
departures_with_labels <- dbGetQuery(con, "select * from airport_labels join departures on airport_label
arrivals_with_labels <- dbGetQuery(con, "select * from airport_labels join arrivals on airport_labels.la

# write final tables to db
new_departs <- data.frame(departures_with_labels$label,departures_with_labels$flight,departures_with_lab

colnames(new_departs) <- c('code','flight','airline','depart','capacity','takeoff')

new_arrivals <- data.frame(arrivals_with_labels$label,arrivals_with_labels$flight,arrivals_with_labels$a

colnames(new_arrivals) <- c('code','flight','airline','arrive','capacity','landing')

dbWriteTable(con, "new_departs", new_departs)
```

```
## Warning: table new_departs exists in database: aborting assignTable
```

```
## [1] FALSE
```

```
dbWriteTable(con, "new_arrivals", new_arrivals)
```

```
## Warning: table new_arrivals exists in database: aborting assignTable
```

```
## [1] FALSE
```

```
# Check data was loaded into db correctly
dbGetQuery(con, "select * from new_departs limit 5")
```

```
##   row.names code flight   airline depart capacity takeoff
## 1         1  DTW   1257  American    DTW      128    1310
## 2         2  DTW   1232  American    DTW      160    1305
## 3         3  DTW   1231  American    DTW      160    1300
## 4         4  DTW    103 Southwest    DTW      136    1615
## 5         5  DTW    101 Southwest    DTW      136    1600
```

```
dbGetQuery(con, "select * from new_arrivals limit 5")
```

```
##   row.names code flight   airline arrive capacity landing
## 1         1  DTW   1291  American    DTW      128    1530
## 2         2  DTW   1278  American    DTW      160    1505
## 3         3  DTW   1277  American    DTW      160    1530
## 4         4  DTW    104 Southwest    DTW      136    1735
## 5         5  DTW    102 Southwest    DTW      136    1815
```

```
# Moderately complex join to find all the flights that left detroit in the morning (before 12pm) with a
oldjoin <- dbGetQuery(con, "select * from new_departs join new_arrivals on new_departs.flight = new_arri

# check oldjoin
head(oldjoin)
```

```
##   row.names code flight airline depart capacity takeoff code flight
## 1        24  BOS     28   Delta    BOS      160    1009  DTW     28
## 2        13  ATL     23   Delta    ATL      160     926  DTW     23
## 3        17  PIT     44   Delta    PIT      160    1100  ATL     44
## 4        23  BOS     38   Delta    BOS      160     816  ATL     38
## 5         8  DTW     24   Delta    DTW      160     914  ATL     24
## 6        22  BOS     45   Delta    BOS      160     744  PIT     45
##   airline arrive capacity landing
## 1   Delta    DTW      160    1228
## 2   Delta    DTW      160    1109
## 3   Delta    ATL      160    1203
## 4   Delta    ATL      160    1040
## 5   Delta    ATL      160    1123
## 6   Delta    PIT      160     855
```

```
# Benchmark old join trial 1
microbenchmark(oldjoin)
```

```
## Unit: nanoseconds
##     expr min lq  mean median uq  max neval
##  oldjoin  50 52 71.26     54 56 1777   100
```

```
# Benchmark old join trial 2
microbenchmark(oldjoin)
```

```
## Unit: nanoseconds
##     expr min   lq  mean median uq  max neval
##  oldjoin  38 39.5 65.36   41.5 51 2073   100
```

```
# Benchmark old join trial 3
microbenchmark(oldjoin)
```

```
## Unit: nanoseconds
##     expr min lq  mean median uq  max neval
##  oldjoin  48 50 72.09   51.5 53 2095   100
```

```
dbRemoveTable(con, "new_departs")
```

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

```
dbRemoveTable(con, "new_arrivals")
```

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

```
# re-rewrite tables
dbWriteTable(con, "new_departs", new_departs)
```

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

```
dbWriteTable(con, "new_arrivals", new_arrivals)
```

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

```
# create indexes on capacity, takeoff and landing
dbGetQuery(con, "create index newid on new_departs (capacity,takeoff)")
```

```
## NULL
```

```
dbGetQuery(con, "create index newid2 on new_arrivals (landing)")
```

```
## NULL
```

```
newjoin <- dbGetQuery(con, "select * from new_departs join new_arrivals on new_departs.flight = new_arri

# check newjoin
head(newjoin)
```

```
##   row.names code flight airline depart capacity takeoff code flight
## 1        24  BOS     28   Delta    BOS      160    1009  DTW     28
## 2        13  ATL     23   Delta    ATL      160     926  DTW     23
## 3        17  PIT     44   Delta    PIT      160    1100  ATL     44
## 4        23  BOS     38   Delta    BOS      160     816  ATL     38
## 5         8  DTW     24   Delta    DTW      160     914  ATL     24
```

```
## 6        22  BOS     45   Delta    BOS      160     744  PIT     45
##   airline arrive capacity landing
## 1   Delta    DTW      160    1228
## 2   Delta    DTW      160    1109
## 3   Delta    ATL      160    1203
## 4   Delta    ATL      160    1040
## 5   Delta    ATL      160    1123
## 6   Delta    PIT      160     855
```

```
# benchmark newjoin trial 1
microbenchmark(newjoin)
```

```
## Unit: nanoseconds
##     expr min lq  mean median uq  max neval
##  newjoin  39 52 73.78     53 54 2367   100
```

```
# benchmark newjoin trial 2
microbenchmark(newjoin)
```

```
## Unit: nanoseconds
##     expr min lq  mean median uq  max neval
##  newjoin  39 42 73.82     53 54 2438   100
```

```
# benchmark newjoin trial 3
microbenchmark(newjoin)
```

```
## Unit: nanoseconds
##     expr min lq  mean median uq  max neval
##  newjoin  41 42 66.63   43.5 46 2316   100
```

The newjoin with the indexes on capacity, takeoff and landing improves performace over three trials. My data set is very small having only joined two files of 6 variables and 24 observations but these savings would be compounded signficantly over millions of observations and dozens of variables.

The reason that I did not add an index for every possible query is that that would decrease performance significantly. The way that these indexes work is that it creates a tree structure over each two table sequence- the variables that are queried and returned in the first table share an id and then that id maps to row instances of where those original variables match in the second table which is represented by another id. The final result is a more direct path the the value you are querying- as opposed to scanning the entire first table row by row, column by column and then the second table row by row and column by column to identify the desired values by using indexes we cut the number of rows in the query non-linearly.

Problem 2.

```
# create a new table from the new_departs and new_arrivals tables

dbRemoveTable(con, "new_departs")
```

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

```
dbRemoveTable(con, "new_arrivals")
```

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

```
dbWriteTable(con, "new_departs", new_departs)
```

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

```
dbWriteTable(con, "new_arrivals", new_arrivals)
```

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

```
nj2 <- dbGetQuery(con, "select * from new_arrivals join new_departs on new_arrivals.flight = new_departs

# prune
nj2_pruned <- data.frame(nj2$code, nj2$flight, nj2$airline, nj2$arrive, nj2$capacity, nj2$landing, nj2$

# rename cols
colnames(nj2_pruned) <- c('code','flight','airline','arrive','capacity','landing','depart','takeoff')

# write new joined nj2 to the db
dbWriteTable(con, "nj2_pruned", nj2_pruned)
```

```
## Warning: table nj2_pruned exists in database: aborting assignTable
```

```
## [1] FALSE
```

```
# query and rank flights by length of travel by longest flight first
nj2_lot <- dbGetQuery(con, "select flight, ((abs(takeoff-landing))/100.0) as traveltime_hrs, airline, d

# write nj2_lot travel time per flight ranked to db
dbWriteTable(con, "nj2_lot", nj2_lot)
```

```
## Warning: table nj2_lot exists in database: aborting assignTable
```

```
## [1] FALSE
```

```
head(dbGetQuery(con, "select * from nj2_lot"))
```

```
##   row.names flight traveltime_hrs  airline depart arrive takeoff landing
## 1         1     27           2.65    Delta    DTW    BOS     945    1210
## 2         2     37           2.43    Delta    ATL    BOS    1201    1444
## 3         3     38           2.24    Delta    BOS    ATL     816    1040
## 4         4   1231           2.20 American    DTW    BOS    1300    1520
## 5         5   1277           2.20 American    BOS    DTW    1310    1530
## 6         6     28           2.19    Delta    BOS    DTW    1009    1228
```

The difference between a view and a table is that in a view we can't modify anything in the db. By modifying the table you actually modify the database. I couldn't do the above without modifying and writing the tables to the database because my goal was to go get a new table back.

The advantages of the view is that you can hide unused columns and select data before writing the table. The advantages of the table is that you can update, delete, and modify the data.

Problem 3.

Benchmark the performance of two different functions- one using base R library and the other using the dplyr library.

The function is to get the the average departure delay of flights.

```
flt_data <- flights
# compare dplyr to base R apply()
# clearly dplyr works faster than the base R function.
measure_perf <- microbenchmark(
  apply(flt_data[4],2,mean,na.rm = TRUE),
  summarize(flt_data,avg_delay = mean(dep_delay, na.rm = TRUE))
  )

measure_perf
```

```
## Unit: milliseconds
##                                                          expr    min
##                       apply(flt_data[4], 2, mean, na.rm = TRUE) 15.214
##  summarize(flt_data, avg_delay = mean(dep_delay, na.rm = TRUE))  5.177
##      lq   mean median     uq   max neval
##  16.771 19.980 19.838 21.522 64.98   100
##   5.458  6.029  5.726  5.996 11.90   100
```