607-HW12

Jose Mawyin

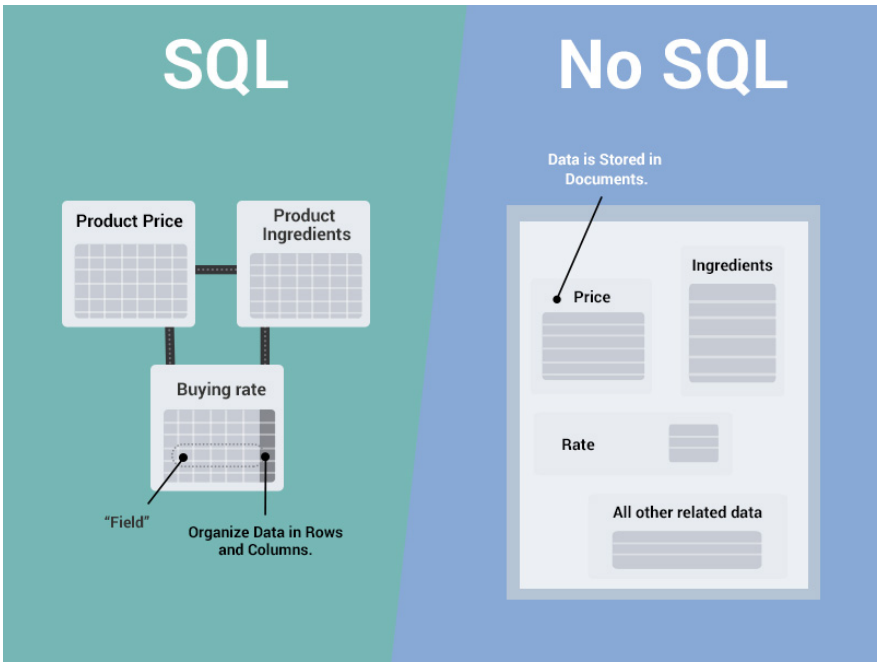
11/22/2019

Week 12 Assignment - NoSQL Migration

1. Introduction
2. Data
3. Loading Data into MySQL
4. Loading Data into MongoDB
5. Comparisson
6. Useful Links

1. Introduction

In this lab we will compare two type of database:



- MySQL: A relational Database
- MongoDB: A non-relational Database

2. Data

We will compare the two types of databases using a large set of data that consists of a list of every arrest in NYC going back to 2006 through the end of the previous calendar year. This dataset was found at:

NYPD Arrests Data (Historic) *https://catalog.data.gov/dataset/nypd-arrests-data-historic*

It is a relatively large datasets consisting of 4798339 rows by 18 columns for the following observations:

```
NYPD_Arrests=data.table::fread("/Users/josemawyin/Downloads/NYPD_Arrests.csv")
names(NYPD_Arrests)
```

```
##  [1] "ARREST_KEY"        "ARREST_DATE"       "PD_CD"
##  [4] "PD_DESC"           "KY_CD"             "OFNS_DESC"
##  [7] "LAW_CODE"          "LAW_CAT_CD"        "ARREST_BORO"
## [10] "ARREST_PRECINCT"   "JURISDICTION_CODE" "AGE_GROUP"
## [13] "PERP_SEX"          "PERP_RACE"         "X_COORD_CD"
## [16] "Y_COORD_CD"        "Latitude"          "Longitude"
```

```
dim(NYPD_Arrests)
```

```
## [1] 4798339      18
```

Since the dataset is so large, we tried to load it as-is into MySQL using the *.CSV import wizard. However, after 5 hours it did not finish to load the data into a table. Therefore, we just created a subset of the first 1000 rows that we then saved as a *.CSV file

```
NYPD_Arrests.1k <- NYPD_Arrests[1:1000, ]
write.csv(NYPD_Arrests.1k, "/Users/josemawyin/Library/Mobile Documents/com~apple~CloudDocs/Data Science
```

Once we had smaller *.CSV file, we just used the import wizard to create a schema and load a table with the information from the file.

CREATE SCHEMA `NYPD_Arrests` CREATE TABLE `NYPD_Arrests.nypd_arrests` (`ARREST_KEY` int, `ARREST_DATE` text, `PD_CD` int, `PD_DESC` text, `KY_CD` int, `OFNS_DESC` text, `LAW_CODE` text, `LAW_CAT_CD` text, `ARREST_BORO` text, `ARREST_PRECINCT` int, `JURISDICTION_CODE` int, `AGE_GROUP` text, `PERP_SEX` text, `PERP_RACE` text, `X_COORD_CD` double, `Y_COORD_CD` double, `Latitude` double, `Longitude` double)

3. Loading Data from MySQL

To get data from MySQL first we set the paraters to connect to the database as shown below:

```
## Entering MySQL local server connection info
drv <- dbDriver("MySQL")
# Loading MySQL data into MyData R dataframe
MyData <- dbConnect(drv, username = "JoseMawyin", password = "Beyfgv{6p",
    dbname = "NYPD_Arrests", host = "localhost")
dbListTables(MyData)
```

```
## [1] "nypd_arrests"   "nypd_arrests1k"
```

Once we had our login credentials and succefully connected to the database, we queried MySQL from R using the commands below:

```
# Function to make it easier to query
query <- function(...) dbGetQuery(MyData, ...)
# Get the ARREST_DATE, PD_DESC, OFNS_DESC, PERP_SEX,
# PERP_RACE, AGE_GROUP when the AGE_GROUP = '18-24' on the
# 'nypd_arrests1k' Table
NYPD_Arrests_MySQL <- query("SELECT ARREST_DATE, PD_DESC, OFNS_DESC, PERP_SEX, PERP_RACE, AGE_GROUP FRO
NYPD_Arrests_MySQL$PERP_RACE <- as.factor(NYPD_Arrests_MySQL$PERP_RACE)
NYPD_Arrests_MySQL$OFNS_DESC <- as.factor(NYPD_Arrests_MySQL$OFNS_DESC)
NYPD_Arrests_MySQL$PERP_SEX <- as.factor(NYPD_Arrests_MySQL$PERP_SEX)
head(NYPD_Arrests_MySQL)
```

```
##   ARREST_DATE                                  PD_DESC
## 1  12/31/2017      NY STATE LAWS,UNCLASSIFIED VIOLATION
## 2  12/31/2017           ROBBERY,UNCLASSIFIED,OPEN AREAS
## 3  12/31/2017                                ASSAULT 3
## 4  12/31/2017                        FORGERY,ETC.-MISD.
## 5  12/31/2017           CONTROLLED SUBSTANCE,POSSESS. 1
## 6  12/31/2017 PUBLIC ADMINISTRATION,UNCLASSIFIED FELONY
##                      OFNS_DESC PERP_SEX      PERP_RACE AGE_GROUP
## 1             OTHER STATE LAWS        M          BLACK     18-24
## 2                      ROBBERY        M          BLACK     18-24
## 3 ASSAULT 3 & RELATED OFFENSES        F WHITE HISPANIC     18-24
## 4     OFFENSES INVOLVING FRAUD        M          WHITE     18-24
## 5              DANGEROUS DRUGS        F          WHITE     18-24
## 6      MISCELLANEOUS PENAL LAW        M          BLACK     18-24
```

```
dim(NYPD_Arrests_MySQL)
```

```
## [1] 206   6
```

Our query results of 206(R)x6(C) was loaded into a dataframe for later use.

4. Loading Data from MongoDB

We preloaded a MongoDB database using the first 1000 rows of the NYPD Arrests Data using the code below.

```
my_collection = mongo(collection = "NYPD_Arrests.1k", db = "New_York")  # create connection, database a
my_collection$drop()  #To avoid entry duplications if we run the commands below multiple times.
my_collection$insert(NYPD_Arrests.1k)
```

```
## List of 5
##  $ nInserted  : num 1000
##  $ nMatched   : num 0
##  $ nRemoved   : num 0
##  $ nUpserted  : num 0
##  $ writeErrors: list()
```

```
my_collection$count()
```

```
## [1] 1000
```

Then we queried the MongoDB database to access the same subset of data we got from the MySQL database.

```
NYPD_Arrests_MongoDB = my_collection$find("{\"AGE_GROUP\" : \"18-24\"}",
    fields = "{\"_id\":0, \"ARREST_DATE\" : 1, \"PD_DESC\" : \"1\" ,\"OFNS_DESC\" : 1, \"PERP_SEX\" : \
NYPD_Arrests_MongoDB$PERP_RACE <- as.factor(NYPD_Arrests_MongoDB$PERP_RACE)
NYPD_Arrests_MongoDB$OFNS_DESC <- as.factor(NYPD_Arrests_MongoDB$OFNS_DESC)
NYPD_Arrests_MongoDB$PERP_SEX <- as.factor(NYPD_Arrests_MongoDB$PERP_SEX)
head(NYPD_Arrests_MongoDB)
```

```
##   ARREST_DATE                                  PD_DESC
## 1  12/31/2017      NY STATE LAWS,UNCLASSIFIED VIOLATION
## 2  12/31/2017           ROBBERY,UNCLASSIFIED,OPEN AREAS
## 3  12/31/2017                                ASSAULT 3
## 4  12/31/2017                    FORGERY,ETC.-MISD.
## 5  12/31/2017           CONTROLLED SUBSTANCE,POSSESS. 1
## 6  12/31/2017 PUBLIC ADMINISTRATION,UNCLASSIFIED FELONY
##                      OFNS_DESC AGE_GROUP PERP_SEX      PERP_RACE
## 1             OTHER STATE LAWS     18-24        M          BLACK
## 2                      ROBBERY     18-24        M          BLACK
## 3 ASSAULT 3 & RELATED OFFENSES     18-24        F WHITE HISPANIC
## 4     OFFENSES INVOLVING FRAUD     18-24        M          WHITE
## 5              DANGEROUS DRUGS     18-24        F          WHITE
## 6      MISCELLANEOUS PENAL LAW     18-24        M          BLACK
```

```
dim(NYPD_Arrests_MongoDB)
```

```
## [1] 206   6
```

Notice how different is the syntax used to produced the same results of 206(R)x6(C):

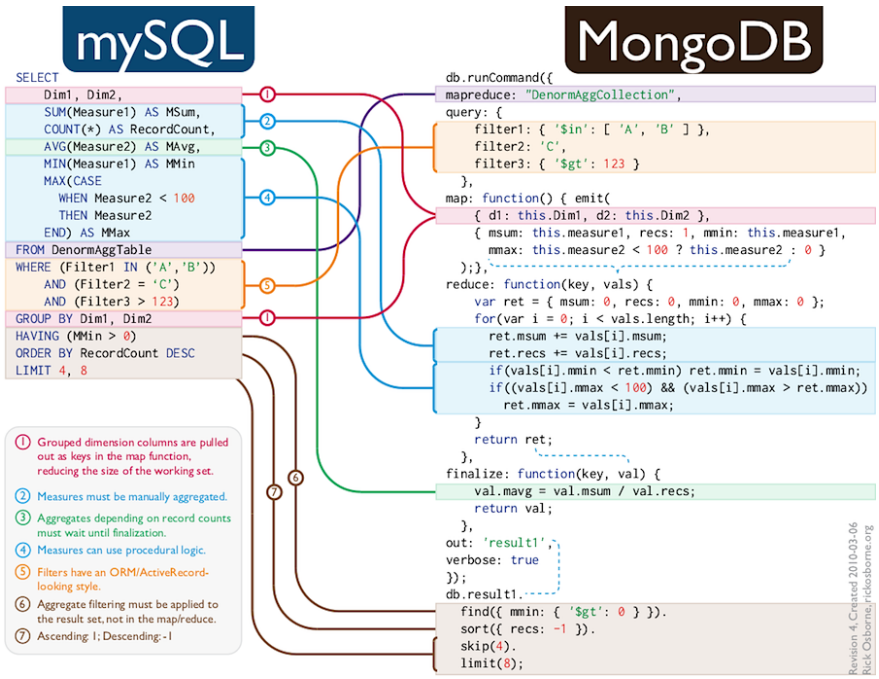
MySQL

> NYPD_Arrests_MySQL <- query("SELECT ARREST_DATE, PD_DESC, OFNS_DESC, PERP_SEX, PERP_RACE, AGE_GROUP FROM nypd_arrests1k WHERE AGE_GROUP='18-24';")

MongoDB

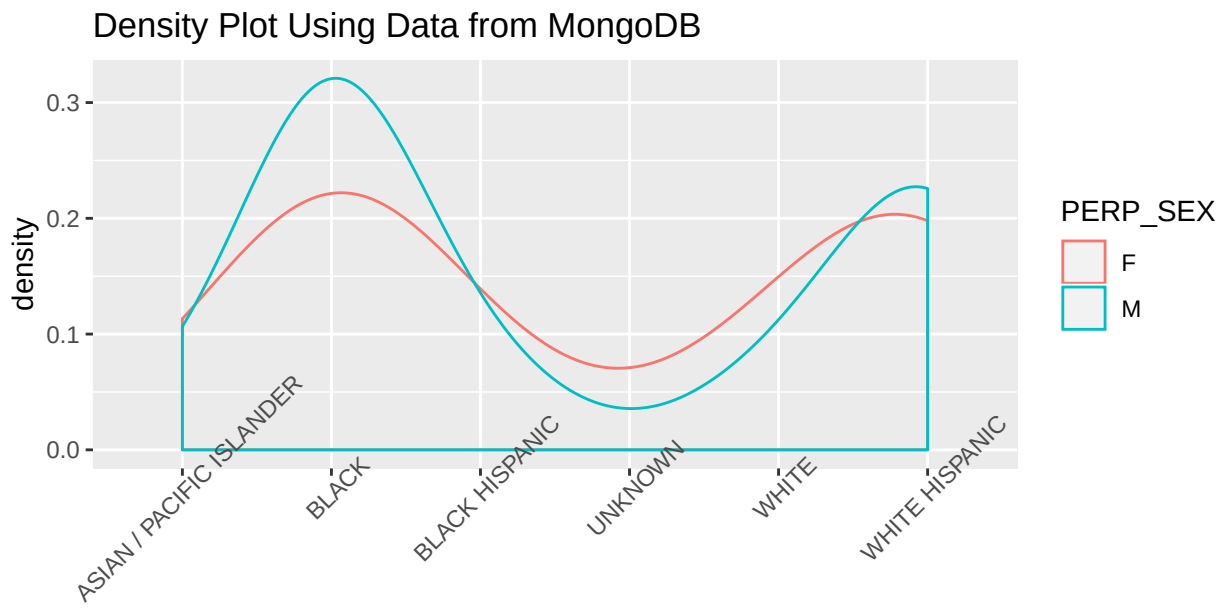
> NYPD_Arrests_MongoDB= my_collection$find('{"AGE_GROUP" : "18-24"}', fields = '{"_id":0,"ARREST_DATE" : 1, "PD_DESC" : "1" ,"OFNS_DESC" : 1, "PERP_SEX" : "1" ,"PERP_RACE" : 1, "AGE_GROUP" : 1 }')

5. Comparisson

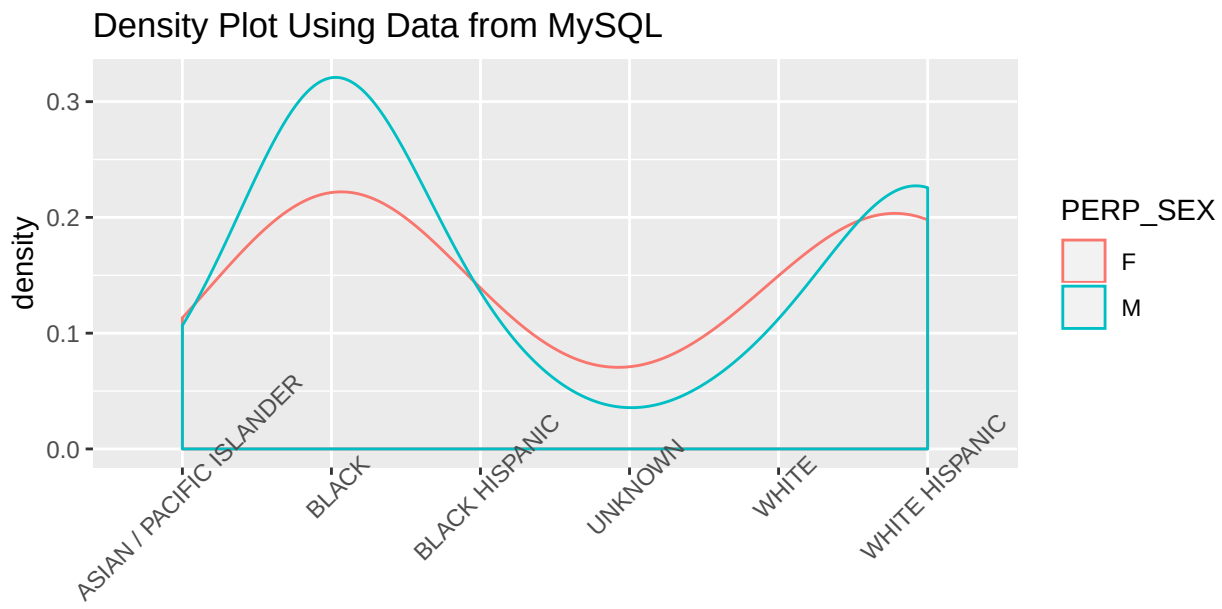


We can see in the diagram above how different are the queries in *MySQL* compared to *MongoDB*. MySQL queries follow a natural-language-like format while MongoDB queries are expressed like a programming language parsing data from a JSON file. Not suprising since MongoDB stores data in documents formed by JSON strings.

```
library(ggplot2)
par(mfrow = c(2, 1))
ggplot(NYPD_Arrests_MongoDB, aes(PERP_RACE, colour = PERP_SEX,
    group = PERP_SEX)) + geom_density() + ggtitle("Density Plot Using Data from MongoDB") +
    theme(axis.text.x = element_text(angle = 45))
```




```
ggplot(NYPD_Arrests_MySQL, aes(PERP_RACE, colour = PERP_SEX,
    group = PERP_SEX)) + geom_density() + ggtitle("Density Plot Using Data from MySQL") +
    theme(axis.text.x = element_text(angle = 45))
```



```
#ggplot(NYPD_Arrests_MongoDB, aes(x = "PERP_SEX", y = "OFNS_DESC")) + geom_tile(aes(fill = "PERP_SEX"))
```

Once we are able to succesfully query data from either MySQL or MongoDB and into R, any data analysis is the same. The two graphs above show the same density plots as the queries were able to get the same data from the two different databases.

6. Useful Links

The followings links were useful in trying to get the databses to work with R.

Installing MondoDB https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/

Read-only file system when attempting mkdir /data/db on Mac https://stackoverflow.com/questions/58034955/read-only-file-system-when-attempting-mkdir-data-db-on-mac

Using MongoDB with R https://datascienceplus.com/using-mongodb-with-r/