Project5_RPostgreSQL_RNeo4J

Prashant B. Bhuyan

December 6, 2014

In this project I will load data into a relational database from within R and then load that dataframe into neo4j as a graph database and then query that data from the neo4j browser.

I'm loading trading data and will create nodes for orders and exchanges. Later for my final project I'll build off of this data model to analyze how orders are distributed between exchanges and sub accounts (or strategies) and how risk concentrates. I may be able to build a predictive model that tells me which strategies are about to go into a drawdown in my portfolio.

Make Sure Neo4J Server Session is Running Before calling "http://localhost:7474/db/data/ with startGraph from RNeo4j.

```r
library(DBI)
library(RPostgreSQL)
library(RNeo4j)

setwd("~/Documents")

# Here I build the graph database to be populated later.

graph = startGraph("http://localhost:7474/db/data/")

# clear(graph)

# Prompt . . . Answer Y

# You are about to delete all nodes, relationships, indexes, and constraints from the  # # graph databa
# 1: Y

trades <- read.csv("account-summary-MRTTRADING1-20140918.csv")
drv <- dbDriver("PostgreSQL")
con <- dbConnect(drv, dbname = "postgres", host = "localhost", port = 5433, user = "postgres", password
dbWriteTable(con, "trades", trades)
```

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

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

```r
# save data from trades table to a data frame
data <- dbGetQuery(con, "select * from trades")

# check data
head(data)
```

```
##   row.names     Account Security        Order.ID Trade.Date Trade.Time
## 1         1 MRTTRADING1      GDX '2783467014823'  9/18/2014   15:58:37
## 2         2 MRTTRADING1      GDX '3049754987175'  9/18/2014   15:58:37
## 3         3 MRTTRADING1      GDX '3316042959527'  9/18/2014   15:58:37
## 4         4 MRTTRADING1      GDX '3582330931879'  9/18/2014   15:58:37
## 5         5 MRTTRADING1      GDX '3848618904231'  9/18/2014   15:58:37
## 6         6 MRTTRADING1      GDX '4114906876583'  9/18/2014   15:58:37
##   Side Liquidity Route Quantity Price Lime.Fee ECN.Fee ACT.Fee SEC.Fee
## 1    1   Removed EDGAA      100 23.15     0.08   -0.02       0       0
## 2    1   Removed EDGAA      100 23.15     0.08   -0.02       0       0
## 3    1   Removed EDGAA      100 23.15     0.08   -0.02       0       0
## 4    1   Removed EDGAA      100 23.15     0.08   -0.02       0       0
## 5    1   Removed EDGAA      100 23.15     0.08   -0.02       0       0
## 6    1   Removed EDGAA      100 23.15     0.08   -0.02       0       0
##   NASD.Fee Rounded.Order.Commission
## 1        0                     0.06
## 2        0                     0.06
## 3        0                     0.06
## 4        0                     0.06
## 5        0                     0.06
## 6        0                     0.06
```

```
# save variables related to an order object into a data frame
orders_data <- data[c('Account', 'Security', 'Order.ID', 'Side', 'Quantity', 'Route')]

# check order_data
head(orders_data)
```

```
##       Account Security        Order.ID Side Quantity Route
## 1 MRTTRADING1      GDX '2783467014823'    1      100 EDGAA
## 2 MRTTRADING1      GDX '3049754987175'    1      100 EDGAA
## 3 MRTTRADING1      GDX '3316042959527'    1      100 EDGAA
## 4 MRTTRADING1      GDX '3582330931879'    1      100 EDGAA
## 5 MRTTRADING1      GDX '3848618904231'    1      100 EDGAA
## 6 MRTTRADING1      GDX '4114906876583'    1      100 EDGAA
```

```
# save variables related to an exchange object into a data frame
exchange_data <- data[c('Route')]

# check exchange_data
head(exchange_data)
```

```
##   Route
## 1 EDGAA
## 2 EDGAA
## 3 EDGAA
## 4 EDGAA
## 5 EDGAA
## 6 EDGAA
```

```
# save variables related to an account object into a data frame
account_data <- data[c('Account')]

# check account_data
head(account_data)
```

```
##         Account
## 1 MRTTRADING1
## 2 MRTTRADING1
## 3 MRTTRADING1
## 4 MRTTRADING1
## 5 MRTTRADING1
## 6 MRTTRADING1
```

```
# save unique order id's into an orders object for the orders node.
order <- unique(orders_data$Order.ID)

# check order
head(order)
```

```
## [1] "'2783467014823'" "'3049754987175'" "'3316042959527'" "'3582330931879'"
## [5] "'3848618904231'" "'4114906876583'"
```

```
# save unique account names into account object for the accounts node.
accounts <- unique(account_data$Account)

# check accounts
head(accounts)
```

```
## [1] "MRTTRADING1"  "MRTTRADING10" "MRTTRADING11" "MRTTRADING13"
## [5] "MRTTRADING2"  "MRTTRADING3"
```

```
# save unique exchange names into the exchange object for the exchanges node.
exchange <- unique(exchange_data$Route)

# check exchange
head(exchange)
```

```
## [1] "EDGAA"    "BATSY"    "BSX"      "INET-FIX" "BATSZ"    "ARCA"
```

```
# load data into the neo4j graph and create the Orders node.
for(x in 1:length(order)){
  createNode(graph, "Orders", order = order[x])
  }

# load data into the neo4j graph and create the Exchanges node.
for(x in 1:length(exchange)){
  createNode(graph, "Exchanges", exchange = exchange[x])
  }

# query some orders from the data loaded to Neo4j into the order node
getNodes(graph, "match n where n:Orders return n limit 5")
```

```
## [[1]]
## $order
## [1] "'2783467014823'"
##
```

```
##
## [[2]]
## $order
## [1] "'3049754987175'"
##
##
## [[3]]
## $order
## [1] "'3316042959527'"
##
##
## [[4]]
## $order
## [1] "'3582330931879'"
##
##
## [[5]]
## $order
## [1] "'3848618904231'"
```

```
# query some exchanges from the data loaded to Neo4j into the exchange node
getNodes(graph, "match n where n:Exchanges return n limit 5")
```

```
## [[1]]
## $exchange
## [1] "EDGAA"
##
##
## [[2]]
## $exchange
## [1] "BATSY"
##
##
## [[3]]
## $exchange
## [1] "BSX"
##
##
## [[4]]
## $exchange
## [1] "INET-FIX"
##
##
## [[5]]
## $exchange
## [1] "BATSZ"
```

The advantage of modeling the trade data above as a graph is that querying paths and relationships of large datasets can be performed in constant time thus paving the way for real time analytics. Of course this particular transactional data are well structured and can be easily stored in a relational database. However, performing analytical operations on any large sized dataset will be extremely inefficient in if that data are stored in a relational database.

Further, using R to manipulate and then push data back and forth between PostgreSQL and Neo4j makes it easy to visualize the data, to create and populate nodes and properties and to define relationships between nodes. The drawback of modeling data as a graph is that neo4j is relatively new and so documentation on how to perform advanced tasks is somewhat scarce. Other than that I don't see much of a disadvantage to modeling transactional data as a graph.