Prashant B. Bhuyan
is607 Data Acquisition and Management
Assignment Week 12

Part 1)

Researched Use Cases for neo4j graph databases.

Part 2a)

<u>High Level Description of Use Case</u>

The infamous 'Flash Crash' of May 6, 2010 where the US Stock Market crashed and subsequently recovered 10% of its value over a chaotic 20 minute period was arguably caused by a gap in the liquidity structure of markets.

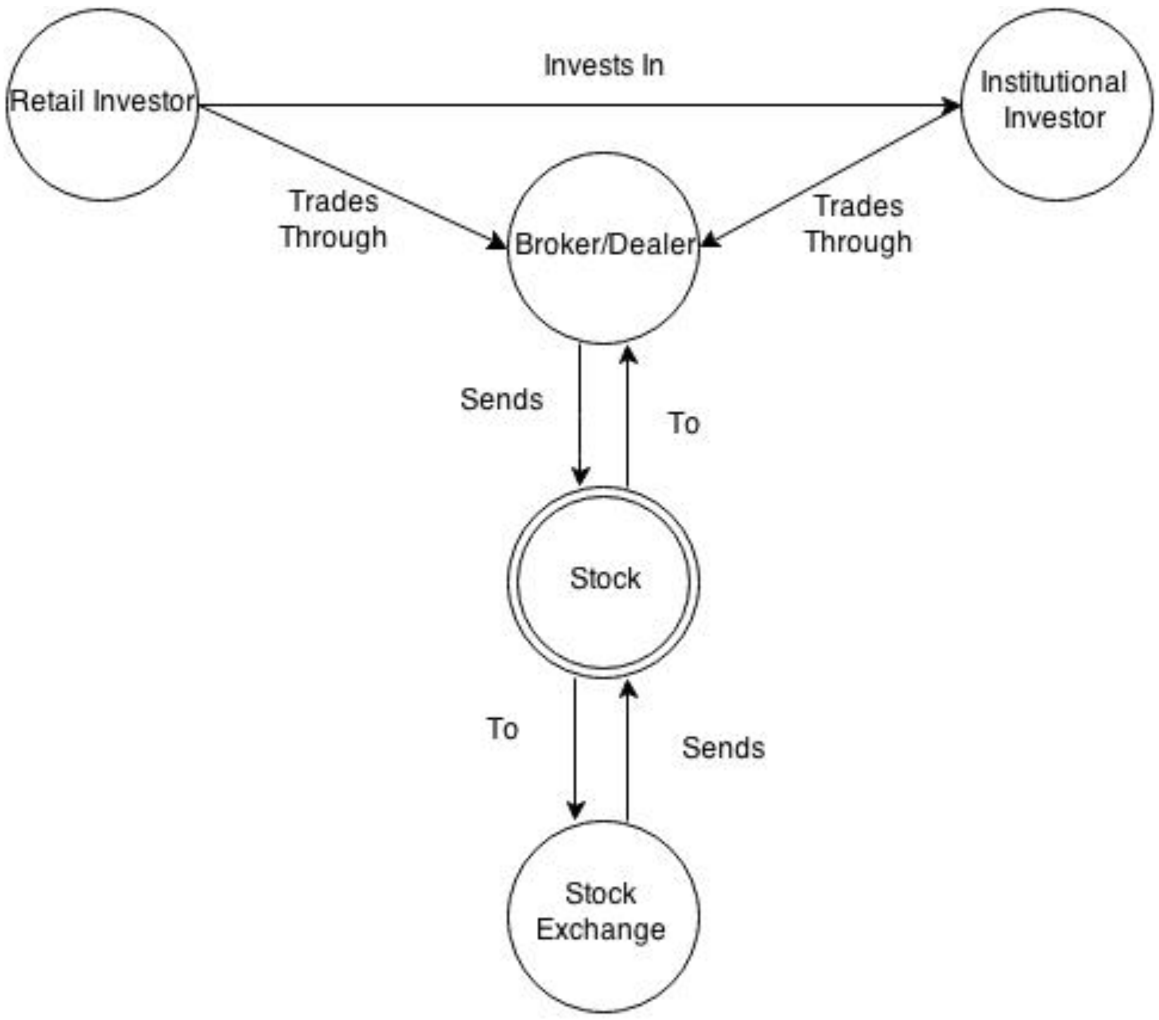
Neo4j is a scalable graph database. As such neo4j is perfectly suited to model the complex relationships between buyers, sellers, intermediaries and exchanges in the stock market since all players are essentially relatable nodes. Since all trades go through just a small set of intermediaries and all intermediaries go through a smaller set of exchanges I believe that all trades are relatable in a graph.

Graph databases such as neo4j are very well suited as recommendation engines that observe relationships and behaviors of nodes within a given system to infer the likelihood of some event (such as the odds that a user will like some new product or discover a new friend in common). Likewise, I hypothesize that if applied to the stock market the graph based system could infer the odds of situations where demand for liquidity would outstrip the availability of liquidity in real time (leveraging constant time queries) in a massively scalable but symbol specific way.

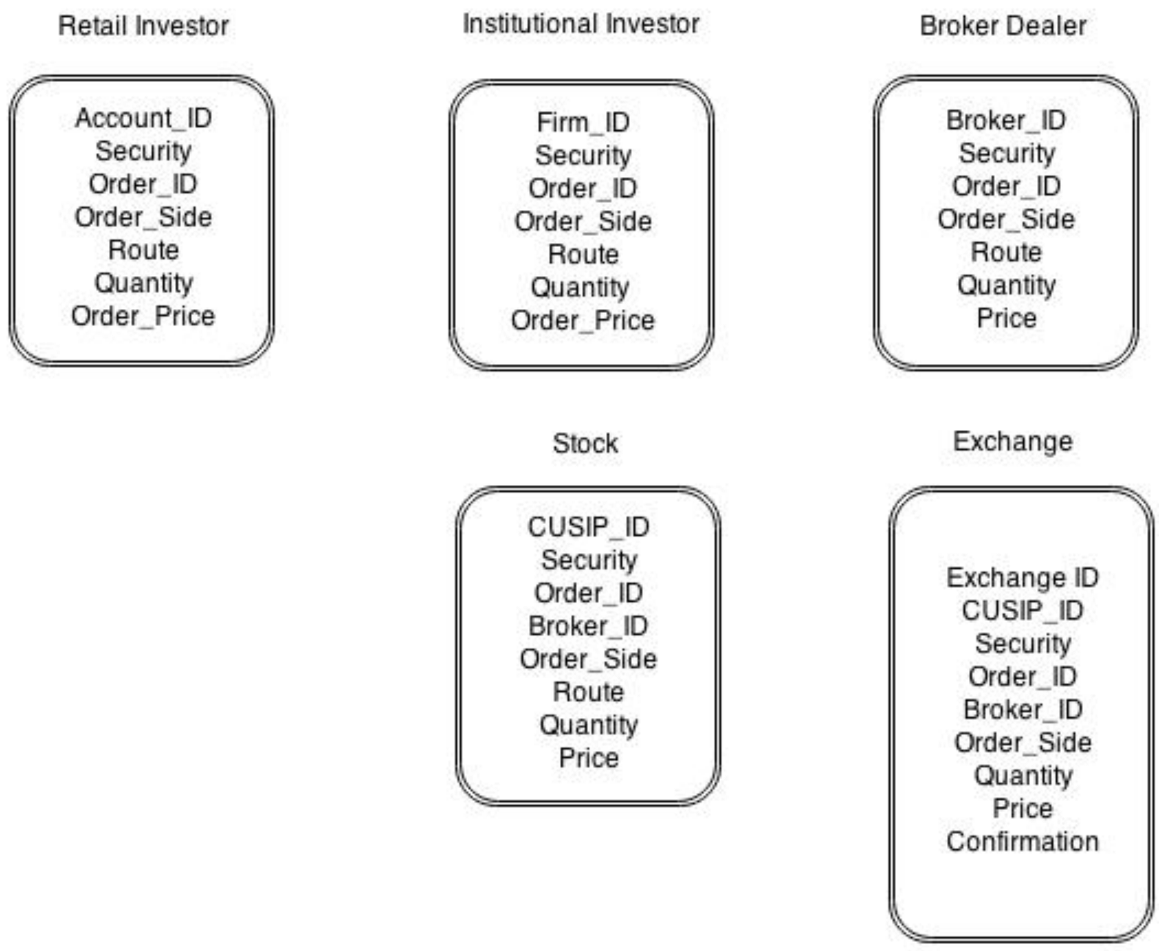
The end result would be a risk management system where if the odds of a liquidity hole in a particular stock breaches some optimized threshold a 'circuit breaker' would be invoked before a crash could take place. However, even if such as system actually worked then it would still have to pass through the bureaucracy of Washington before implementation in the real world vis a vis the SEC, Finra Regulations etc. Nevertheless, such a system would add real value as an internal risk check to many banks, broker dealers and other intermediaries that may want to manage event risk associated with the next 'Flash Crash'.

Part 2b)

Description of Data Model (Nodes, Relationships and Attributes)



Node Attributes



Part 2c)

Retail Investor Data:

Account_ID	Security	Order_ID	Order_Side	Route	Quantity	Order_Price
joe1	SPLV	'146361289383'	3	BATSY	100	37.25
joe1	SPLV	'163541158567'	3	EDGAA	100	37.25
joe1	LQD	'395469392551'	1	BATSY	100	118.59
joe1	VNQ	'2388334217895'	2	BATSY	100	79.2
joe1	EWW	'8384108563111'	1	EDGAA	100	67.195
joe1	EWW	'8401288432295'	1	BSX	100	67.2

Institutional Investor Data:

Firm_ID	Security	Order_ID	Order_Side	Route	Quantity	Price
0002XXX	SPLV	'146361289383'	3	BATSY	100	37.25
0002XXX	SPLV	'163541158567'	3	EDGAA	100	37.25
0002XXX	LQD	'395469392551'	1	BATSY	100	118.59
0002XXX	VNQ	'2388334217895'	2	BATSY	100	79.2
0002XXX	EWW	'8384108563111'	1	EDGAA	100	67.195
0002XXX	EWW	'8401288432295'	1	BSX	100	67.2

Broker/Dealer Data:

Broker_ID	Security	Order_ID	Order_Side	Route	Quantity	Price
brk1001	SPLV	'146361289383'	3	BATSY	100	37.25
brk1001	SPLV	'163541158567'	3	EDGAA	100	37.25
brk1001	LQD	'395469392551'	1	BATSY	100	118.59
brk1001	VNQ	'2388334217895'	2	BATSY	100	79.2
brk1001	EWW	'8384108563111'	1	EDGAA	100	67.195
brk1001	EWW	'8401288432295'	1	BSX	100	67.2

Stock Data:

CUSIP_ID	Security	Order_ID	Broker_ID	Order_Side	Route	Quantity	Price
123555	SPLV	'146361289383'	brk1001	3	BATSY	100	37.25
643223	SPLV	'163541158567'	brk1001	3	EDGAA	100	37.25
724333	LQD	'395469392551'	brk1001	1	BATSY	100	118.59
550055	VNQ	'2388334217895'	brk1001	2	BATSY	100	79.2
723444	EWW	'8384108563111'	brk1001	1	EDGAA	100	67.195
773356	EWW	'8401288432295'	brk1001	1	BSX	100	67.2

Part 2d)

Here is code for Cypher queries to acquire and manage the data in Neo4j.

Loading Stock Data

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_stock.csv' as stocks create (s1: Stocks {CUSIP_ID:
stocks.CUSIP_ID, Security: stocks.Security, Order_ID: stocks.Order_ID,
Broker_ID: stocks.Broker_ID, Order_Side: stocks.Order_Side, Route:
```

```
stocks.Route, Quantity: stocks.Quantity, Price: stocks.Price})
```

Added 6 labels, created 6 nodes, set 48 properties, returned 0 rows in 907 ms

Loading Retail Investor Data

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_retailinvestor.csv' as retailinvestors create (a: Retailinvestor
{Account_ID: retailinvestors.Account_ID, Security:
retailinvestors.Security, Order_ID: retailinvestors.Order_ID, Route:
retailinvestors.Route, Quantity: retailinvestors.Quantity, Order_Price:
retailinvestors.Order_Price})
```

Added 6 labels, created 6 nodes, set 36 properties, returned 0 rows in 899 ms

Loading Institutional Investor Data

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_instinvestor.csv' as institutionalinvestors create (b:
Institutionalinvestor {Firm_ID: institutionalinvestors.Firm_ID,
Security: institutionalinvestors.Security, Order_ID:
institutionalinvestors.Order_ID, Order_Side:
institutionalinvestors.Order_Side, Route: institutionalinvestors.Route,
Quantity: institutionalinvestors.Quantity, Price:
institutionalinvestors.Price})
```

Added 6 labels, created 6 nodes, set 42 properties, returned 0 rows in 214 ms

Loading Exchange Data

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_exchange.csv' as exchanges create (d: Exchanges {Exchange_ID:
exchanges.Exchange_ID, CUSIP_ID: exchanges.CUSIP_ID, Security:
exchanges.Security, Order_ID: exchanges.Order_ID, Broker_ID:
exchanges.Broker_ID, Order_Side: exchanges.Order_Side, Quantity:
exchanges.Quantity, Price: exchanges.Price, Confirmation:
exchanges.Confirmation})
```

Added 6 labels, created 6 nodes, set 54 properties, returned 0 rows in 274 ms

Loading Broker Data

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_brokerdealer.csv' as brokers create (c: Brokers {Broker_ID:
brokers.Broker_ID, Security: brokers.Security, Order_ID:
brokers.Order_ID, Order_Side: brokers.Order_Side, Route: brokers.Route,
Quantity: brokers.Quantity, Price: brokers.Price})
```

```
Added 6 labels, created 6 nodes, set 42 properties, returned 0 rows in 197 ms
```

Part 2e)

Create Relationship Between Retail Investors and Institutional Investors

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_retailinvestor.csv' as retailinvestors match(a: Retailinvestor
{Order_ID: retailinvestors.Order_ID}),(b: Institutionalinvestor
{Order_ID:retailinvestors.Order_ID}) create (a) - [r:Invests_IN] -> (b)
```

```
Created 6 relationships, returned 0 rows in 457 ms
```

Create Relationship Between Retail Investors and Broker/Dealer

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_retailinvestor.csv' as retailinvestors match(a: Retailinvestor
{Order_ID: retailinvestors.Order_ID}),(c: Brokers
{Order_ID:retailinvestors.Order_ID}) create (a) - [r:Trades_Through] ->
(c)
```

```
Created 6 relationships, returned 0 rows in 509 ms
```

Create Relationship Between Institutional Investor and Broker/Dealer

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_instinvestor.csv' as institutionalinvestors match(b:
Institutionalinvestor {Order_ID: institutionalinvestors.Order_ID}),(c:
Brokers {Order_ID:institutionalinvestors.Order_ID}) create (b) -
[r:Trades_Through] -> (c)
```

```
Created 6 relationships, returned 0 rows in 181 ms
```

Create Relationship Between Broker and Exchange

*(*Please Note: I decided that the relationship between the Broker and the Stock was Redundant- A Broker sends a stock to an exchange and receives a confirmation back. I left the diagram above to contain the stock node just because you can see my thought pattern- the model became clearer to me after I started to map the relationships of the nodes which I think was the point of this assignment).*

Broker Sends Order to Exchange . . .

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_brokerdealer.csv' as brokers match(c: Brokers {Order_ID:
brokers.Order_ID}),(d: Exchanges {Order_ID: brokers.Order_ID}) create
(c) - [r:Sends_To] -> (d)
```

```
Created 6 relationships, returned 0 rows in 236 ms
```

Broker Receives Order Confirmation From Exchange . . .

```
load csv with headers from
'file:/Users/MicrostrRes/Desktop/is607AssignmentWk12/stockmarket_graphm
odel_exchange.csv' as exchanges match(d: Exchanges {Order_ID:
exchanges.Order_ID}),(c: Brokers {Order_ID: exchanges.Order_ID}) create
(d) - [r:Sends_Confirm_Back_To] -> (c)
```

```
Created 6 relationships, returned 0 rows in 187 ms
```