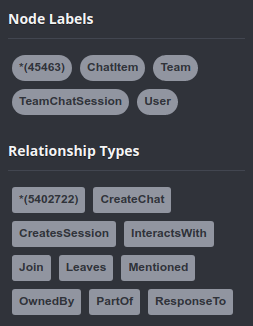
**Graph Analytics**

**Modeling Chat Data using a Graph Data Model**

(Describe the graph model for chats in a few sentences. Try to be clear and complete.)

* The graph model contains User node and Team node and are setup for chatting by relationship TeamChatSession and CreateSession.
* User node can use relationship Join to connect with a TeamChatSession node. Similarly they can use relationship Leave to end their session with the TeamChatSession node
* User node use the relationship CreateChat to create ChatItem node. Also, ChatItem is connected with a relationship PartOf to the TeamChatSession node.
* User can be Mentioned in a ChatItem and a ChatItem can be created in ResponseTo relationship a previous ChatItem node.

**Creation of the Graph Database for Chats**

Describe the steps you took for creating the graph database. As part of these steps

1. Write the schema of the 6 CSV files

|  |  |  |
| --- | --- | --- |
| **File Name** | **Description** | **Fields** |
| chat\_create\_team\_chat.csv | userid | the user id assigned to the user |
| teamid | the id of the team |
| teamChatSessionID | a unique id for the chat session |
| timestamp | a timestamp denoting when the chat session created |
| chat\_item\_team\_chat.csv | userid | the user id assigned to the user |
| teamchatsessionid | a unique id for the chat session |
| chatitemid | a unique id for the chat item |
| timestamp | a timestamp denoting when the  chat item created |
| chat\_join\_team\_chat.csv | userid | the user id assigned to the user |
| teamChatSessionID | a unique id for the chat session |
| timestamp | a timestamp denoting when the  user join in a chat session |
| chat\_leave\_team\_chat.csv | userid | the user id assigned to the user |
| teamchatsessionid | a unique id for the chat session |
| timestamp | a timestamp denoting when the user leave a chat session |
| chat\_mention\_team\_chat.csv | ChatItemId | the id of the ChatItem |
| userid | the user id assigned to the user |
| timeStamp | a timestamp denoting when the user mentioned by a chat item |
| chat\_respond\_team\_chat.csv | chatid1 | the id of the chat post 1 |
| chatid2 | the id of the chat post 2 |
| timestamp | a timestamp denoting when the chat post 1 responds to the chat post 2 |

1. Explain the loading process and include a sample LOAD command

Load process of a file in Neo4j starts with a LOAD command where you specify the file to load. Each row of the file is process by MERGE commands that creates nodes and relations, assigning values for attributes of these entities with the values of the columns of the row.

# chat\_item\_team\_chat.csv

**LOAD CSV FROM "**file:///home/rahul/workspace/coursera-sdsc-words/coursera/big-data-capstone/chat-data/chat\_item\_team\_chat.csv**" AS row**

**MERGE (u:User {id: toInt(row[0])})**

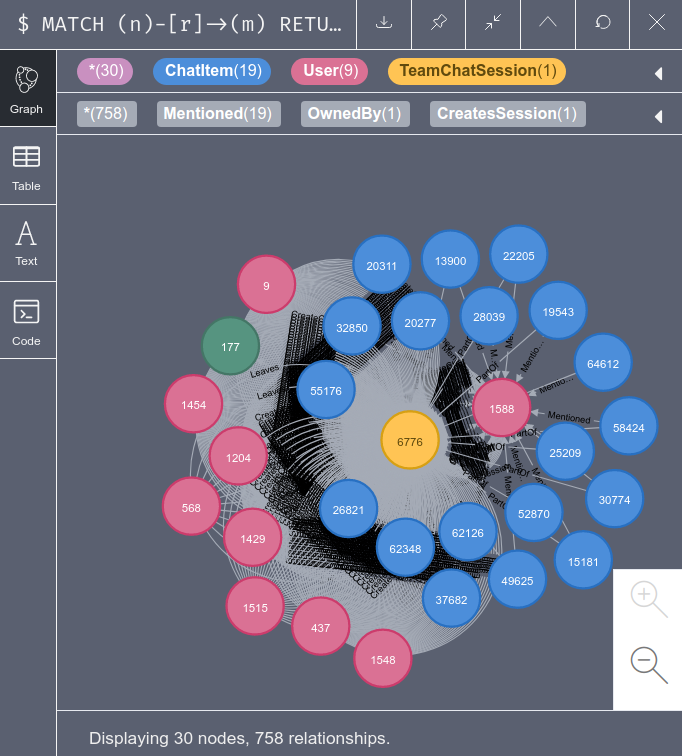
**MERGE (c:TeamChatSession {id: toInt(row[1])})**

**MERGE (i:ChatItem {id: toInt(row[2])})**

**MERGE (u)-[:CreateChat{timeStamp: row[3]}]->(c)**

**MERGE (i)-[:PartOf{timeStamp: row[3]}]->(c)**;

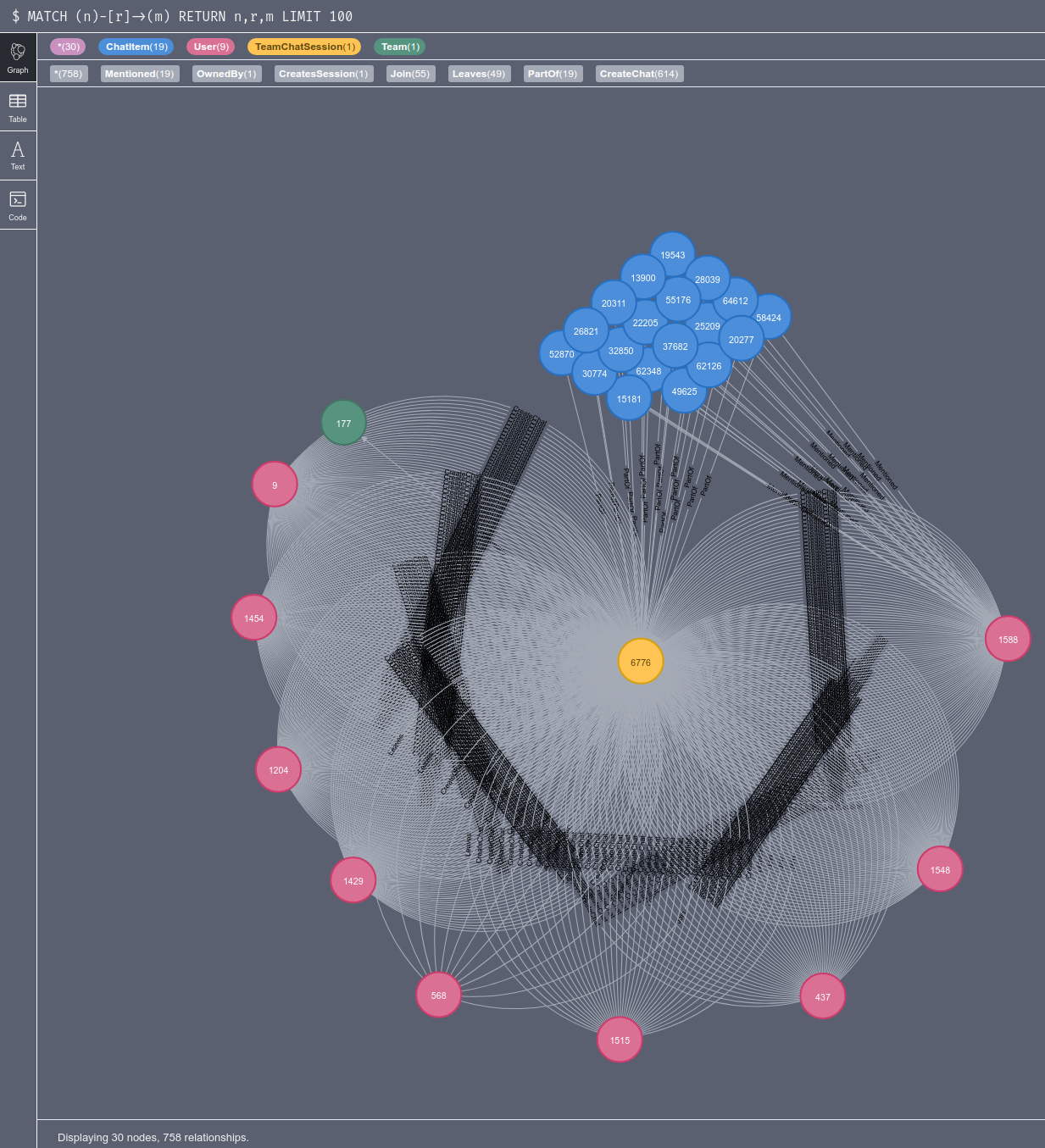
The above loading statement reads the CSV file row wise and maps the various columns to nodes and relationship properties. We see that User, TeamChatSession and ChatItem are nodes receiving data from column 0,1,2 respectively. In the last two lines we see relationships being created. A CreateChat relationship is connected from User node to TeamChatSession node. A PartOf relationship is created from ChatItem to TeamChatSession node. Both the relationship receive timestamp as properties.

1. Present a screenshot of some part of the graph you have generated. The graphs must include clearly visible examples of most node and edge types.

Below are two views that include examples of most node and edge types.

Query: MATCH (n)-[r]->(m) RETURN n,r,m LIMIT 100

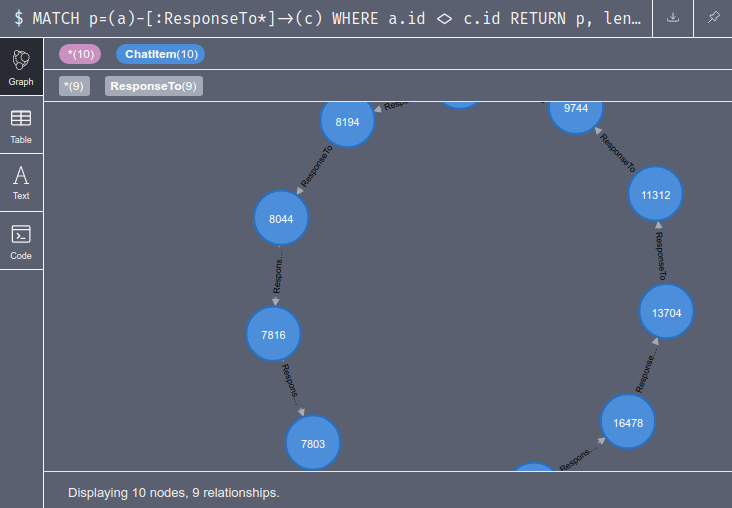




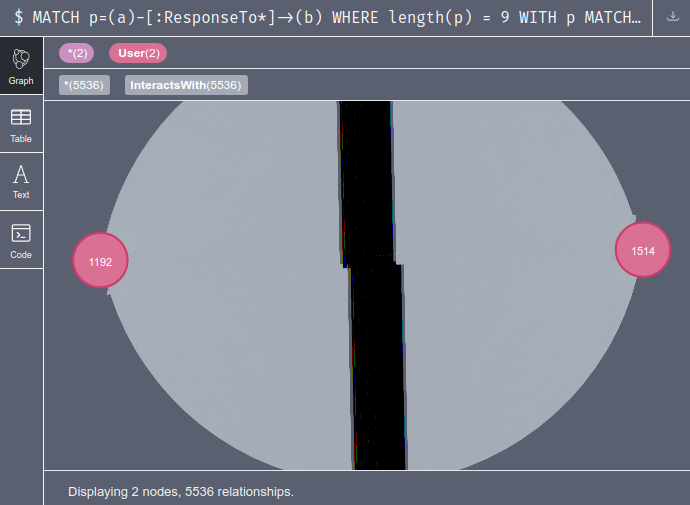
**Finding the longest conversation chain and its participants**

Report the results including the length of the conversation (path length) and how many unique users were part of the conversation chain. Describe your steps. Write the query that produces the correct answer.

|  |  |
| --- | --- |
| **Query** | MATCH p=(a)-[:ResponseTo\*]->(c) WHERE a.id <> c.id RETURN length(p) ORDER BY length(p) DESC LIMIT 1; |
| **Path Length** | 9 |

There are 10 ChatItem nodes and 9 Relationships in this path.

|  |  |
| --- | --- |
| **Query** | MATCH p=(a)-[:ResponseTo\*]->(b) WHERE length(p) = 9 WITH p MATCH (u:User)-[]-(d) WHERE d IN NODES(p) RETURN DISTINCT u; |
| **2 Distinct Users** | (:User {id: 1514}), (:User {id: 1192}) |



**Analyzing the relationship between top 10 chattiest users and top 10 chattiest teams**

Describe your steps from Question 2. In the process, create the following two tables. You only need to include the top 3 for each table. Identify and report whether any of the chattiest users were part of any of the chattiest teams.

* We need to first Match and count distinct users in the Create chat relationship.
* Next we Match the ChatItem that are part of a session and owned by a team for the counts per team.
* Finally, we check the entire chain from user to ChatSession which is OwnedBy Team for user and team for the chattiest session

**Chattiest Users**

**Query:** MATCH (u:User)-[:CreateChat]->() RETURN u.id AS userID, COUNT(u) ORDER BY COUNT(u) DESC LIMIT 10;

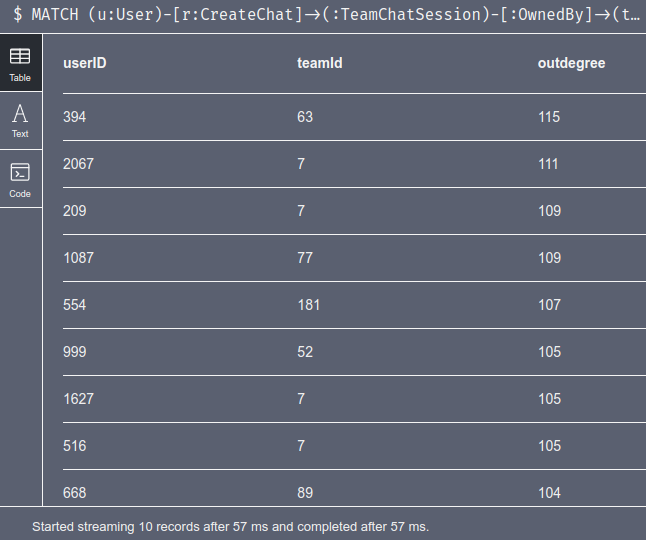
|  |  |
| --- | --- |
| **Users** | **Number of Chats** |
| 394 | 115 |
| 2067 | 111 |
| 1087 | 109 |

**Chattiest Teams**

**Query:** MATCH (i:ChatItem)-[:PartOf]->(c:TeamChatSession)-[:OwnedBy]->(t:Team) RETURN t.id AS teamId, COUNT(i) ORDER BY COUNT(i) DESC LIMIT 10;

|  |  |
| --- | --- |
| **Teams** | **Number of Chats** |
| 82 | 1324 |
| 185 | 1036 |
| 112 | 957 |

Finally, present your answer, i.e. whether or not any of the chattiest users are part of any of the chattiest teams.

**Query:** MATCH (u:User)-[r:CreateChat]->(:TeamChatSession)-[:OwnedBy]->(t:Team) RETURN u.id AS userID, t.id as teamId, COUNT(r) AS outdegree ORDER BY outdegree DESC LIMIT 10;

When we compare this userId to the chattiest user list we see that **all except User:Id:999** shown here is the chattiest.

We see that **only Team:Id:52** in the chattiest team show up here.

**How Active Are Groups of Users?**

Describe your steps for performing this analysis. Be as clear, concise, and as brief as possible. Finally, report the top 3 most active users in the table below.

We **create relationship InteractWith** to make a query later. The relationship is on User node which are creating chat, mentioned, or responding to a Chat item.

* MATCH (u1:User)-[:CreateChat]->(c:TeamChatSession)<-[:PartOf]-(i:ChatItem)-[:Mentioned]->(u2:User) CREATE (u1)-[:InteractsWith]->(u2);
* MATCH (u1:User)-[:CreateChat]->(c1:TeamChatSession)<-[:PartOf]-(i1:ChatItem)<-[:ResponseTo]-(i2:ChatItem)-[:PartOf]->(c2:TeamChatSession)<-[:CreateChat]-(u2:User) CREATE (u1)-[:InteractsWith]->(u2);
* MATCH (u1)-[r:InteractsWith]->(u1) DELETE r;

Now we query the User node with their neighbors (counted as *K*) so as to compute the cluster coefficient based on the formula below

**Cluster Coefficient = COUNT(InteractsWith edge) / (k\*(k-1))**

**Query**:

MATCH (u1:User{id:1192})-[iw1:InteractsWith]->(u2:User)

WITH COLLECT(u2.id) as neighbours, COUNT(u2) AS k

MATCH (u3:User)-[iw2:InteractsWith]->(u4:User)

WHERE (u3.id in (neighbours)) AND (u4.id in (neighbours)) AND (u3.id <> u4.id)

WITH COUNT(iw2) AS numerator, (k \* (k - 1) \* 1.0) AS denominator

RETURN numerator/denominator AS clusterCoefficient;

**Most Active Users (based on top users)**

|  |  |
| --- | --- |
| **User ID** | **Coefficient** |
| User{id:1192} | 0.4722 |
| User{id:394} | 0.2908 |
| User{id:1087} | 0.2618 |

We have used non-distinct count k for User{id:1192} which is 11118. It includes a large amount of InteractWith edges when compared to all Users having 5284220 such edges. Whereas distinct count for User{id:1192} is 6 and all users is 714.

Actual Answers (Fix the query for Cluster coefficient query)

|  |  |
| --- | --- |
| User | Coefficient |
| 209 | 0.9523809523809523 |
| 554 | 0.9047619047619048 |
| 1087 | 0.8 |