**Graph Analytics**

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

There are 4 types of nodes in total: user, team session, team, chat item. And different edge types represent different actions between the nodes:

“CreatesSession” edge represents that a user creates a team chat session.

“OwnedBy” edge represents that a team chat session is owned by a team.

"Joins" edge represents that a user joins a team chat session.

“Leaves” edge represents that a user leaves a team chat session.

“CreateChat” edge represents that a user creates chat item.

“PartOf” edge represents that a chat item is part of a chat session.

“Mentioned” edge represents that a user is mentioned in a chat item.

“ResponseTo” edge represents that a chat item is a response to another chat item.

**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

1. File: ​chat\_create\_team\_chat.csv

Schema: userid, teamid, TeamChatSessionID, timestamp

2. File: chat\_join\_team\_chat.csv

Schema: userid, TeamChatSessionID, teamstamp

3. File: chat\_leave\_team\_chat.csv

Schema: userid, teamchatsessionid, timestamp

4. File: chat\_item\_team\_chat.csv

Schema: userid, teamchatsessionid, chatitemid, timestamp

5. File: chat\_mention\_team\_chat.csv

Schema: userid, teamchatsessionid, chatitemid, timestamp

6. File: chat\_respond\_team\_chat.csv

Schema: chatitemid1, chatitemid2, timestamp

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

Step 1:

Create the constraints for uniqueness of IDs:

CREATE CONSTRAINT ON (u:User) ASSERT u.id IS UNIQUE;

CREATE CONSTRAINT ON (t:Team) ASSERT t.id IS UNIQUE;

CREATE CONSTRAINT ON (c:TeamChatSession) ASSERT c.id IS UNIQUE;

CREATE CONSTRAINT ON (i:ChatItem) ASSERT i.id IS UNIQUE;

Step 2:

Load the csv files into the database according to the schema. For example, when loading chat\_join\_team\_chat.csv, the following command was used:

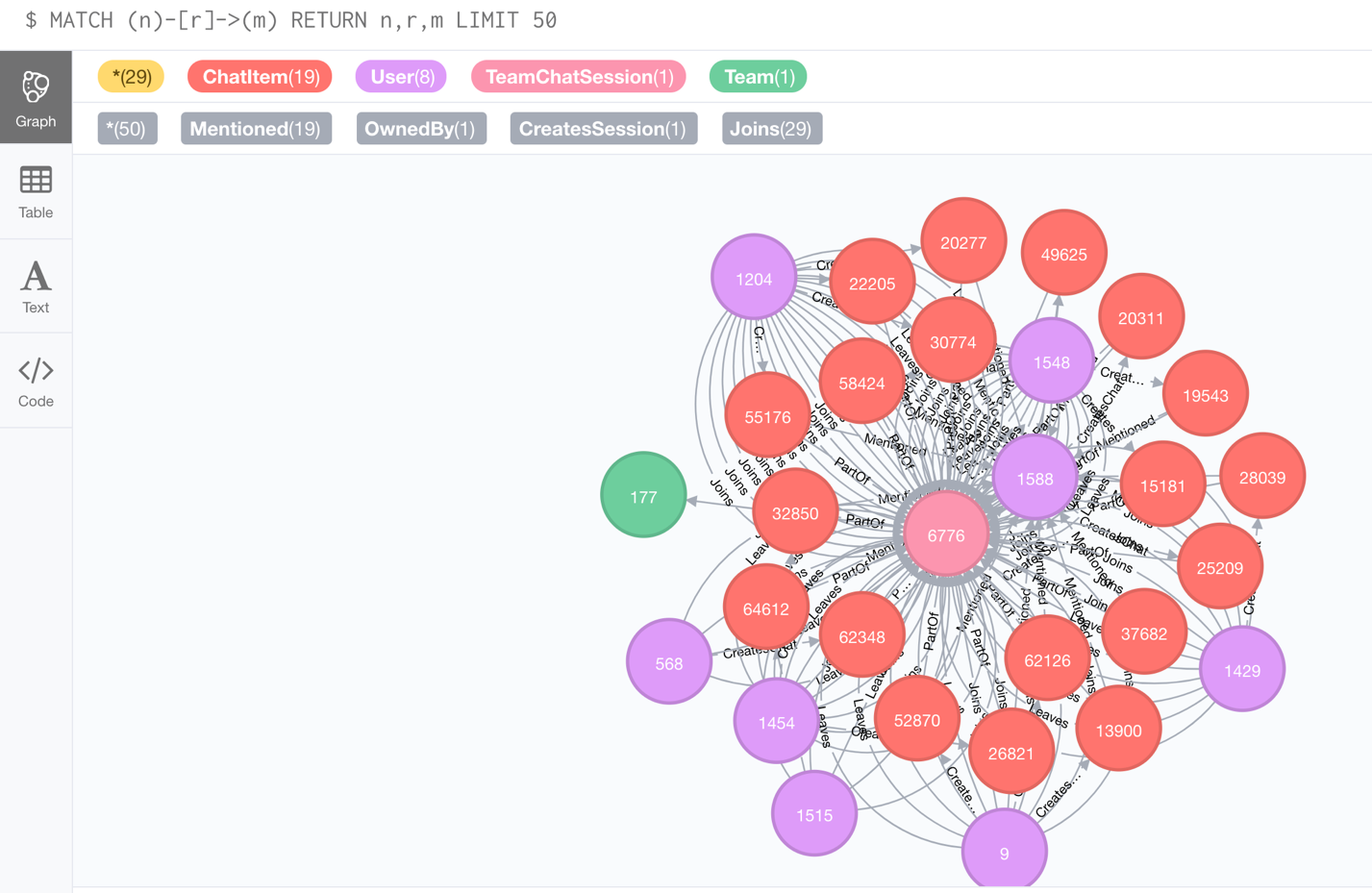
LOAD CSV FROM "file:///chat\_join\_team\_chat.csv" AS row

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

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

MERGE (u)-[:Joins{timeStamp: row[2]}]->(c)

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.

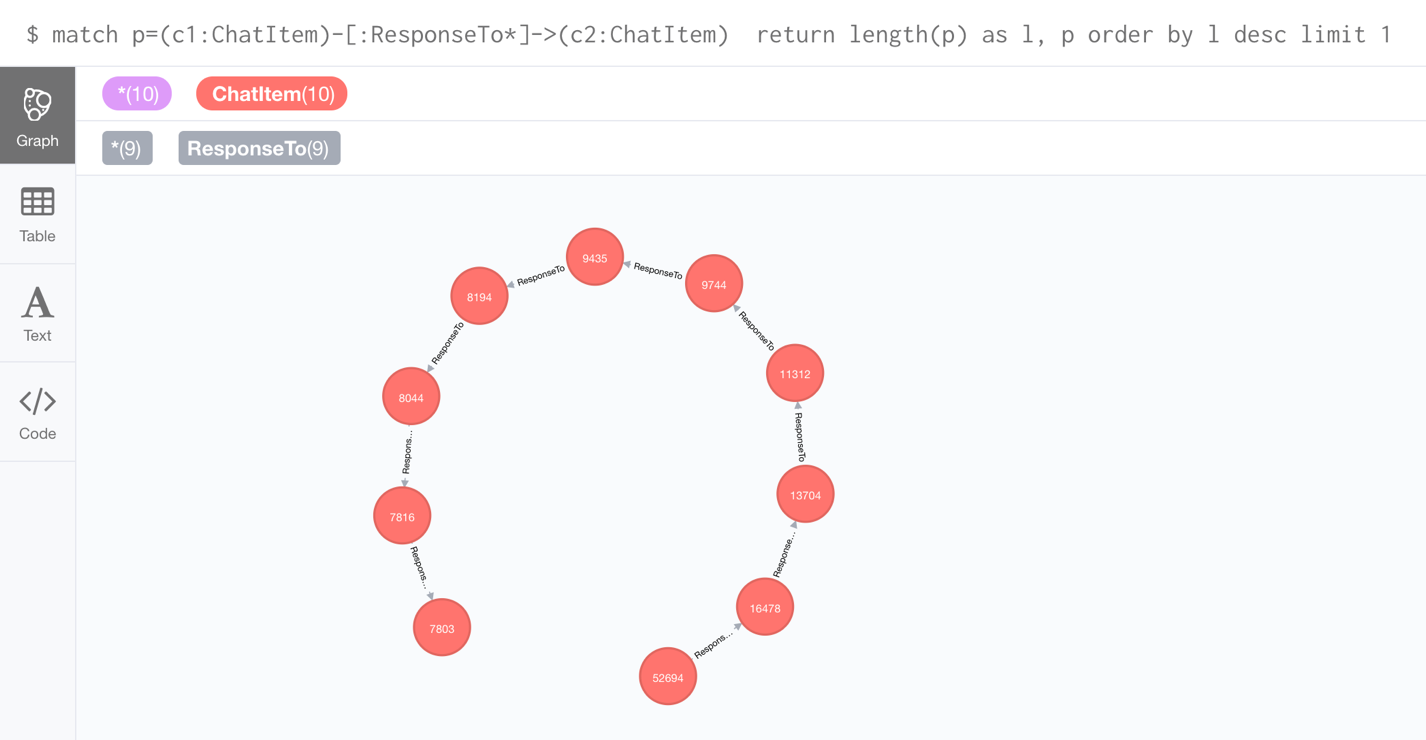


**Finding the longest conversation chain and its participants**

To find the longest conversation chain, I used the command:

match p=(c1:ChatItem)-[:ResponseTo\*]->(c2:ChatItem) return length(p) as l, p order by l desc limit 1

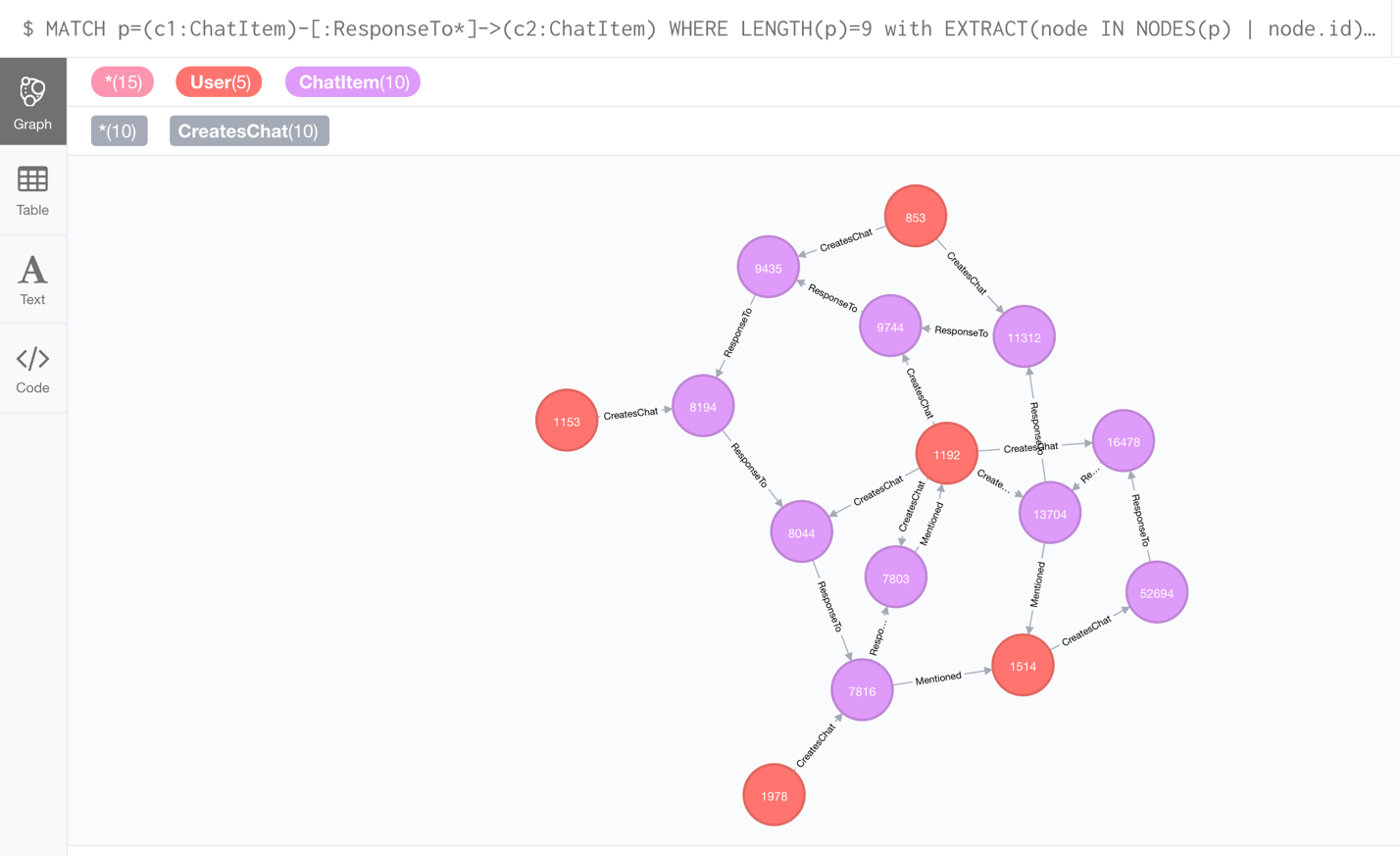
The result is shown below. We can see that the length of the longest path is 9.



To get the number of distinct users involved, the following command is used:

MATCH p=(c1:ChatItem)-[:ResponseTo\*]->(c2:ChatItem) WHERE LENGTH(p)=9 WITH EXTRACT(node IN NODES(p) | node.id) AS ids

MATCH (u:User)-[r:CreatesChat]-(c:ChatItem) WHERE c.id in ids RETURN u,r,c;



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

To find the chattiest users, the following command is used:

MATCH (u:User)-[r:CreatesChat]->() RETURN u.id, COUNT(u) ORDER BY COUNT(u) DESC LIMIT 10;

**Chattiest Users**

|  |  |
| --- | --- |
| **Users** | **Number of Chats** |
| 394 | 115 |
| 2067 | 111 |
| 209 | 109 |

To find the chattiest teams, the following command is used:

MATCH (ci:ChatItem)-[p:PartOf]->(tc:TeamChatSession)-[o:OwnedBy]->(t:Team) RETURN t.id, COUNT(t) ORDER BY COUNT(t) DESC LIMIT 10;

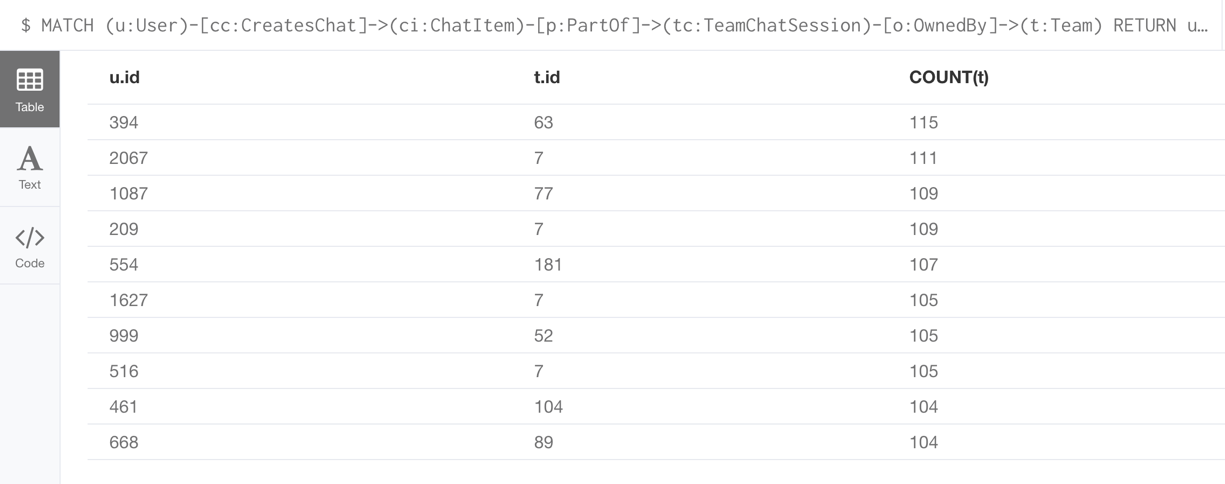
**Chattiest Teams**

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

To determine whether the chattiest users belong to the chattiest team:

MATCH (u:User)-[cc:CreatesChat]->(ci:ChatItem)-[p:PartOf]->(tc:TeamChatSession)-[o:OwnedBy]->(t:Team) RETURN u.id, t.id, COUNT(t) ORDER BY COUNT(t) DESC LIMIT 10;

We can see that user 999 is one of the chattiest users, and it belongs to one of the chattiest team, team 52.



**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.

Step 1:

Create the “InteractsWith” relation:

MATCH (u1:User)-[r:CreatesChat]->(ci1:ChatItem)-[m:Mentioned]->(u2:User) CREATE (u1)-[:InteractsWith]->(u2)

MATCH (u1:User)-[:CreatesChat]->(ci1:ChatItem)-[r:ResponseTo]->(ci2:ChatItem)<-[:CreatesChat]-(u2:User) CREATE (u1)-[:InteractsWith]->(u2)

Match (u1)-[r:InteractsWith]->(u1) delete r

Step 2:

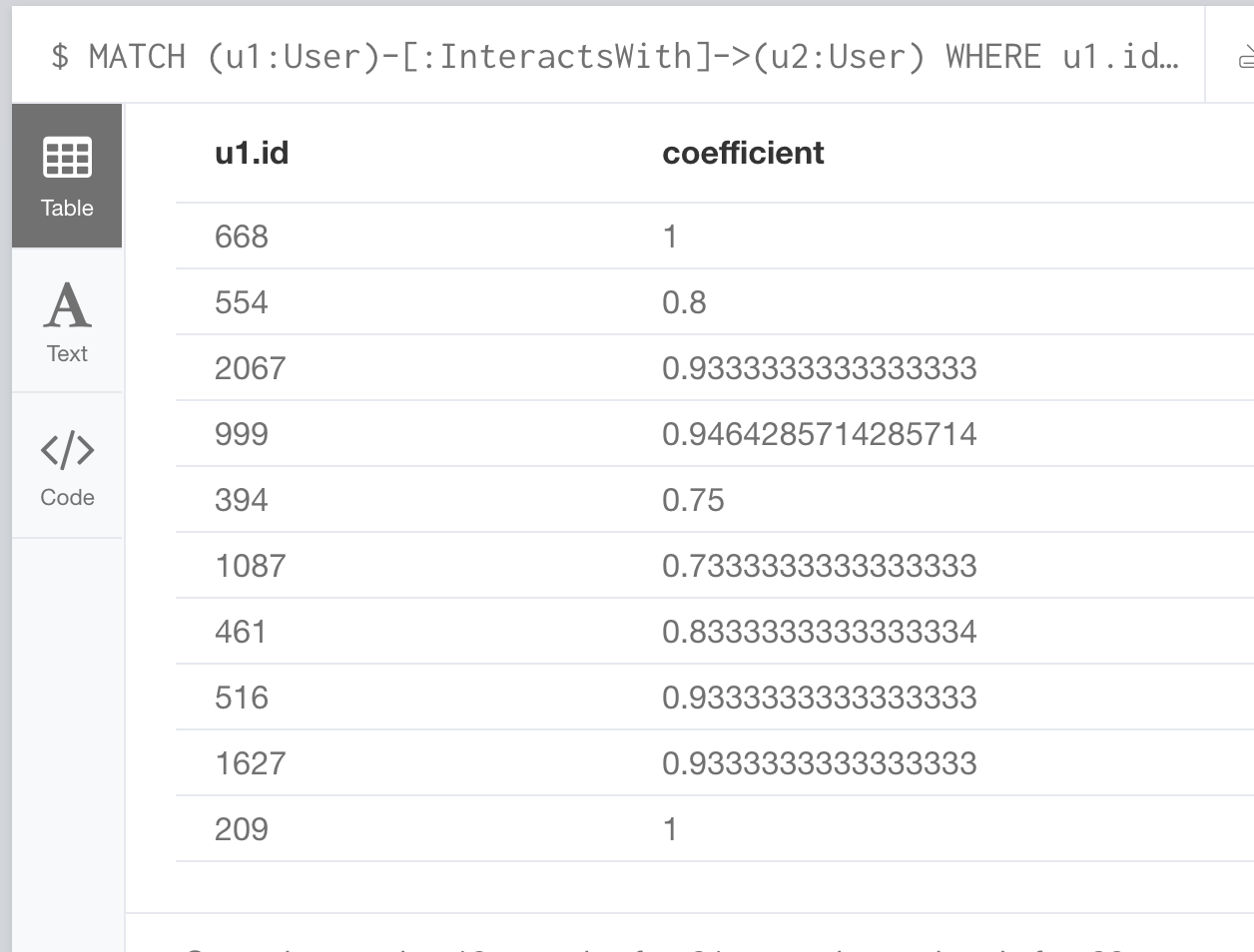
Calculate the coefficients:

MATCH (u1:User)-[:InteractsWith]->(u2:User) WHERE u1.id IN [394, 2067, 209, 1087, 554, 516, 1627, 999, 668, 461] WITH u1, collect(distinct u2.id) AS neighbors

MATCH (u3:User), (u4:User) WHERE u3.id IN neighbors AND u4.id IN neighbors AND u3<>u4

WITH u1, length(neighbors) AS k, sum(case when (u3)-[:InteractsWith]->(u4) then 1 else 0 end) as numEdge

RETURN u1.id, 1.0\*numEdge/(k\*(k-1)) as coefficient



**Most Active Users (based on Cluster Coefficients)**

|  |  |
| --- | --- |
| **User ID** | **Coefficient** |
| 209 | 1 |
| 668 | 1 |
| 2067 | 0.946 |