**Graph Analytics**

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

This graph model contains the chat data of the Catch the Pink Flamingo game. It basically includes the relations between the users of the game, the different teams formed by the users, and the chats between all users and between users in a team. Users can join and leave teams, create chats, respond to chats and, mention other users in a chat.

**Creation of the Graph Database for Chats**

The graph database is constructed from six .csv files. Below the characteristics of these files are described:

**File 1 chat\_create\_team\_chat.csv**

Description A line is added to this file when a player creates a new chat with their team.

**Columns** userid id of the user

teamid id of the team

TeamChatSessionID id of the chat session

timestamp datetime of creation team chat

**File 2 chat\_item\_team\_chat.csv**

**Description A line is added to this file when a player, who is part of a team, creates a chat item.**

Columns userid id of the user

TeamChatSessionID id of the chat session

chatitemid id of the chat item

timestamp datetime of creation chat item

**File 3 chat\_join\_team\_chat.csv**

Description A line is added when a player joins a team.

**Columns** userid id of the user

TeamChatSessionID id of the chat session

timestamp datetime of moment user joins team

**File 4 chat\_leave\_team\_chat.csv**

Description A line is added when a player leaves a team.

**Columns** userid id of the user

TeamChatSessionID id of the chat session

timestamp datetime of moment user leaves team

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

Description A line is added when a player is mentioned in a chat item

**Columns ChatItem id of the chat item**

userid id of the user

timeStamp datetime of moment user is mentioned

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

**Description** A line is added to this file when player responds to a chat post by another player.

**Columns** chatid1 **id of the chat of the responding user**

chatid2 **id of the chat which is responded to**

timestamp datetime of moment response is posted

Each file is loaded separately into the graph database. By loading the files Nodes and/or relationships are created. A sample LOAD command in Cypher language is show below:

**LOAD** **CSV** **FROM** "file:/chat-data/chat\_item\_team\_chat.csv" **AS** row

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

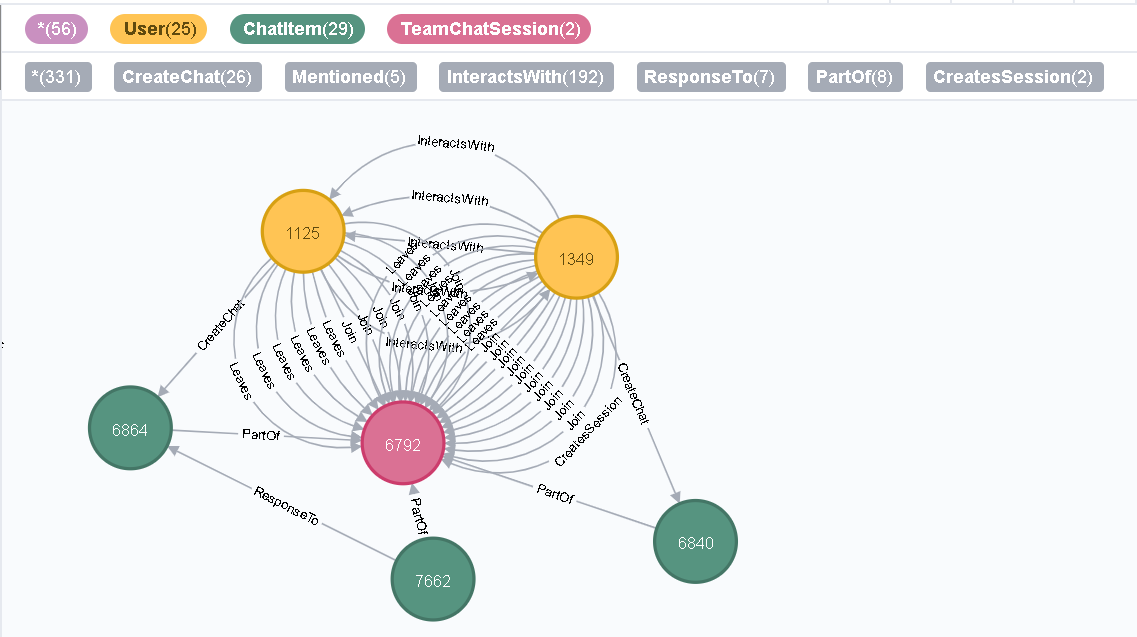
**MERGE** (c:TeamChatSession {id: toInteger(row[1])})

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

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

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

The first line loads the .csv file and works through it row by row. Line two to four creates the “User”, “TeamChatSession” and “ChatItem” nodes and line five and six create the “CreateChat” and “PartOf” relations between these nodes. A part of the created graph database with most of the nodes and relations is shown in the screenshot below.



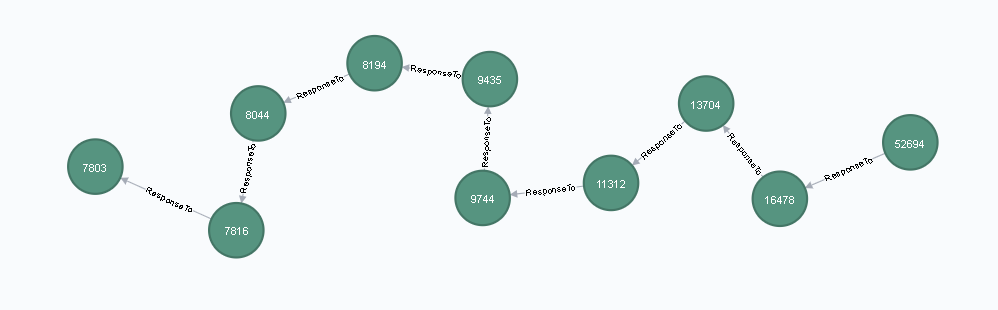
**Finding the longest conversation chain and its participants**

The longest conversation chain and its participants can be found using two queries. First, the longest conversation chain is determined using the “ResponseTo” relation, see code below.

**MATCH** p=(a)-[:ResponseTo\*]->(c)

**RETURN** length(p) **ORDER** **BY** length(p) **DESC** **LIMIT** 1

This query result in a path length of 9, which means 9 “ResponseTo” relations and thus 1 original chat an 9 reply chats. The path is show in the figure below.



Next, the path length of the longest conversation chain is used to find all participating users, see code below.

**MATCH** path = (a)-[:ResponseTo\*9]->(c)

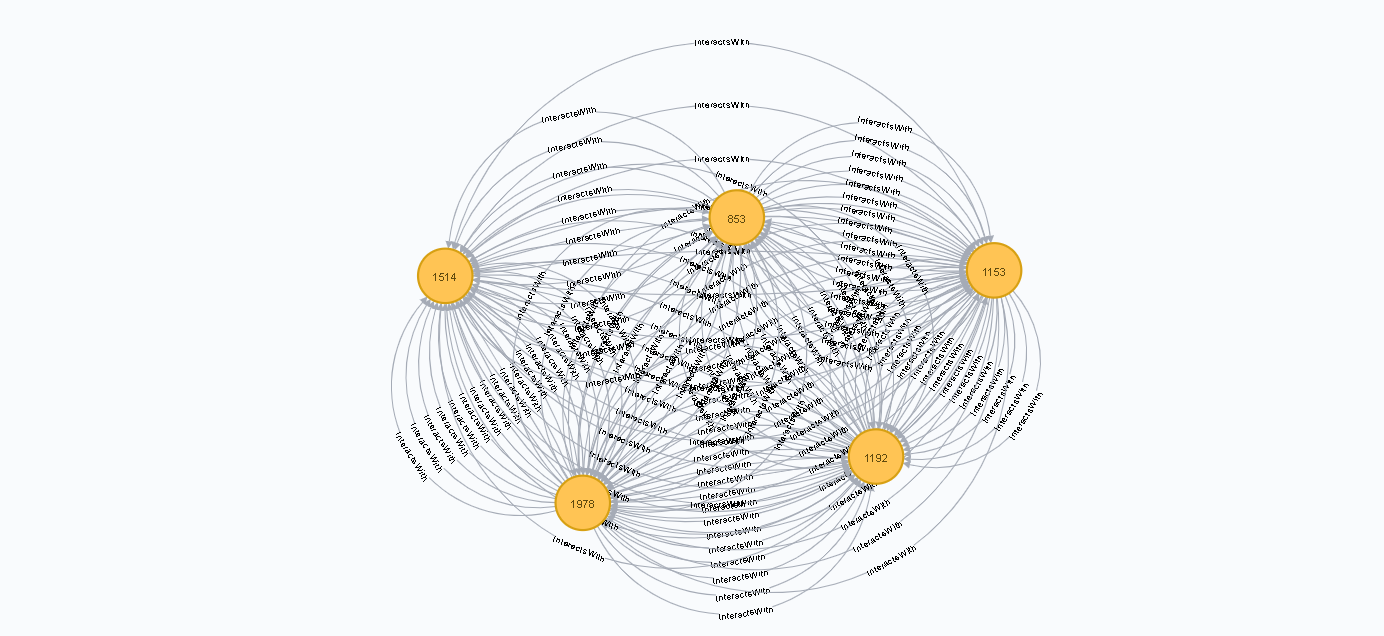
**WITH** NODES(path) **AS** ChatItems

**MATCH** (u:User) - [:CreateChat] -> (i:ChatItem)

**WHERE** i **in** ChatItems

**RETURN** count(**distinct**(u))

This query results in a number of distinct users of 5. The users are shown in the figure below.



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

The relation between the top 10 chattiest users and the top 10 chattiest teams can be investigated using three queries. First, the top 10 chattiest users are determined using the code below.

**MATCH** (u:User) - [:CreateChat] -> (i:ChatItem)

**RETURN** u.id **AS** UserId, count(i) **AS** nrOfChats **ORDER** **BY** count(i) **DESC** **LIMIT** 10

The top 3 chattiest users including their number of chats are presented in the table below.

**Chattiest Users**

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

In the second step the top 10 chattiest teams are determiner using the code below.

**MATCH** (i:ChatItem) - [e:PartOf] - (c:TeamChatSession) - [:OwnedBy] -> (t)

**RETURN** t.id **AS** TeamID, count(i) **AS** nrOfChats **ORDER** **BY** count(i) **DESC** **LIMIT** 10

The top 3 chattiest teams including their number of chats are presented in the table below.

**Chattiest Teams**

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

Finally, the relation between the top 10 chattiest users and top 10 chattiest teams is investigated using the following code:

**MATCH** (u:User) - [:CreateChat] - (i:ChatItem) - [:PartOf] - (c:TeamChatSession) - [:OwnedBy] -> (t)

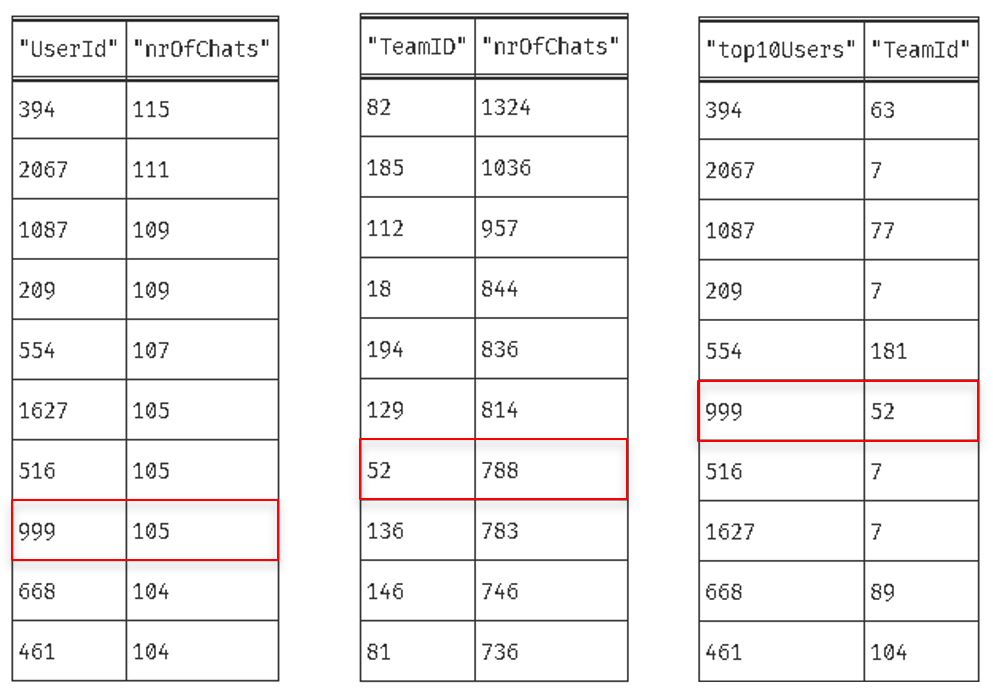
**WITH** count(i) **AS** nr, u.id **AS** top10Users, t.id **AS** TeamId

**ORDER** **BY** count(i) **DESC**

**LIMIT** 10

**RETURN** top10Users, TeamId

As show in the tables below, only one of the chattiest users (user 999) is in one of the chattiest teams (group 52).



**How Active Are Groups of Users?**

The activity of a group of users can be defined by how mutually interactive this group is. This can be determined by estimating the “density” of the neighborhood of a node.

First of all, a new relation type “InteractsWith” is added. This “InteractsWith” relation will form edges between users who mention other users and users who respond to a chat of another user.

In the code below the relations are created between users that mention other users.

**MATCH** (u1:User) - [:CreateChat] - (i:ChatItem) - [:Mentioned] -> (u2:User)

**CREATE** (u1) - [:InteractsWith] -> (u2)

In the code below the relations are created between users who respond to a chat of another user.

**MATCH** (u1:User) - [:CreateChat] - (i:ChatItem) - [:ResponseTo] - (i2:ChatItem) - [:CreateChat] - (u2:User)

**CREATE** (u1) - [:InteractsWith] -> (u2)

The code above has an undesirable side effect of creating a self-loop if a user has responded to their own chat. These self-loops are deleted using the code below.

**MATCH** (u1) - [r: InteractsWith] -> (u1) **DELETE** r

Now, with these new relations in place, the density of a neighborhood can be determined and expressed as a “clustering coefficient”. This coefficient is determined as follows:

clustering coefficient = e / (k \* (k – 1))  
 With:   
 e = number of unique interconnections between the nodes in the neighborhood  
 k = number of neighbors of a node

The density of the neighborhoods will be determined for the top 10 most chattiest users. Their ID’s are  
[394, 2067, 1087, 209, 554, 1627, 516, 999, 668, 461]. The code and a description is provided below.

**MATCH** (u1:User) - [r1:InteractsWith] -> (u2:User)

**WHERE** u1.id <> u2.id

**AND** u1.id **in** [ 394, 2067, 1087, 209, 554, 1627, 516, 999, 668, 461 ]

**WITH** u1, collect(u2.id) **AS** neighbors, count(**distinct**(u2)) **AS** k

**MATCH** (u3:User) - [r2:InteractsWith] -> (u4:User)

**WHERE** (u3.id <> u4.id) **AND** (u3.id **IN** neighbors) **AND** (u4.id **IN** neighbors)

**WITH** u1, u3, u4, k,

**CASE**

**WHEN** count(r2) > 0 **THEN** 1

**ELSE** 0

**END** **AS** value

**RETURN** u1.id **as** UserId, sum(value) \* 1.0 /(k \* (k - 1)) **AS** coefficient

**ORDER** **BY** coefficient **DESC**

Line 1 get all users that are connected to other user through a “InteractsWith” edge.  
Line 2 – 3 filter users that are not connected to themselves and part of the top 10 chattiest users.  
Line 4 store the neighbors as a list and calculate the number of neighbors.  
Line 5 – 6 get all users that are in the neighbors list and are not connected to themselves.   
Line 7 – 11 for each neighbor node calculate the unique connections to other neighbor nodes.  
Line 12 – 13 calculate the “clustering coefficient” and return it for each user in the top 10 chattiest users.

Below the three most active users in the top 10 chattiest users based on their cluster coefficient are provided.

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

|  |  |
| --- | --- |
| **User ID** | **Coefficient** |
| 209 | 0.952 |
| 554 | 0.905 |
| 1087 | 0.800 |