Graph Analytics

Modeling Chat Data using a Graph Data Model

The graph model for chat data models the entities and relationships involved in chat activity. The model allows us to analyze chat behavior to determine how users and teams chat.

For example, the model can tell us which users and teams chat the most frequently (or not at all), if the chats are conversational or not, which users are the most mentioned in a chat (the influencers).

This data could be used to improve the chat feature of the game or target users for promotion.

Creation of the Graph Database for Chats

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

i) Write the schema of the 6 CSV files

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

userId	The user creating the chat
teamId	The teamId of the chat
teamChatSessionId	The Id for the chat session
timestamp	The time the chatSession was created

chat_item_team_chat: A line is created for each chat item (a message posted to a chat)

userId	The user posting the chat item
teamChatSessionId	The session the item was posted to
chatItemId	The id for the chat item
timestamp	The time the item was posted

chat_join_team_chat : A line is created when a user joins a chat

userId	The user joining the chat
teamChatSessionId	The session id being joined

timestamp	The time the user joined

chat_leave_team_chat : A line is created when a user leaves a chat

userId	The user leaving the chat
teamChatSessionId	The session id being left
timestamp	The time the user left

chat_mention _team_chat : A line is created when a user is mentioned in a chat item

chatItemId	The id of the chat item which mentioned the user
userId	The user mentioned
timestamp	The time the mention occurred

chat_respond _team_chat : A line is created when a user responds to a chat item

chatId1	The chat item when initiated the response
chatId2	The response chat item
timestamp	The time the response occurred

ii) Explain the loading process and include a sample LOAD command

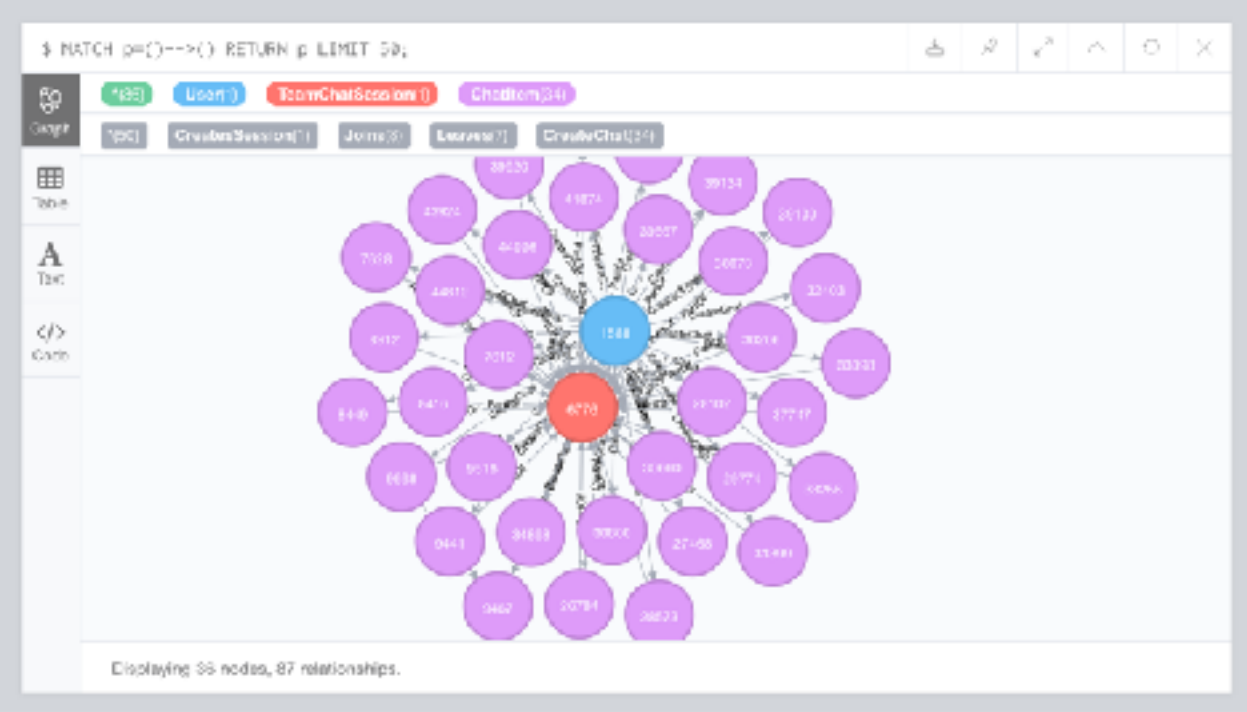
The loading process is the process of loading the graph database with data. In our case, data was loaded from comma separated files. When data is loaded, the entities and relationships are created in the graph database.

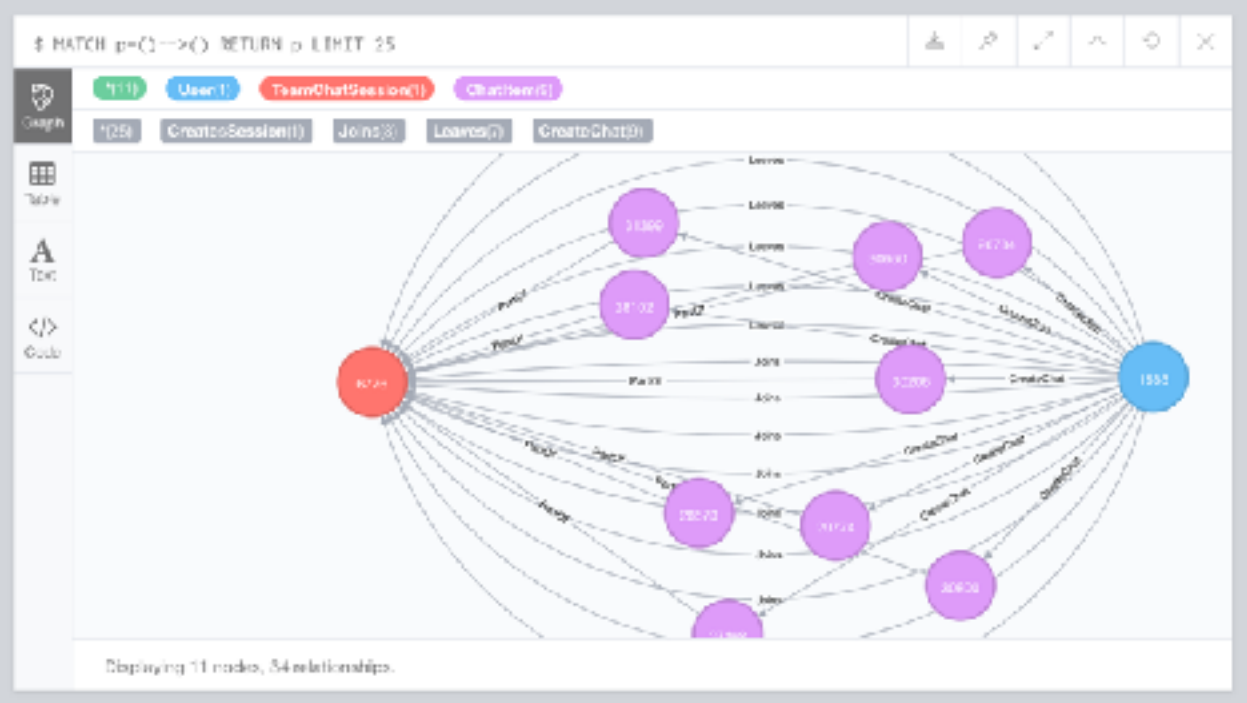
In this example, we are loading the team chats. We create the user, team, and teamChatSession entities, along with relationships telling us which user created the chat session and which team owns the chat session.

```
LOAD CSV FROM "file:///big-data/datasets/big-data-capstone/chat/
chat_create_team_chat.csv" AS row
MERGE (u:User {id: toInteger(row[0])})
MERGE (t:Team {id: toInteger(row[1])})
MERGE (c:TeamChatSession {id: toInteger(row[2])})
MERGE (u)-[:CreatesSession{timeStamp: row[3]}]->(c)
MERGE (c)-[:OwnedBy{timeStamp: row[3]}]->(t);
```

iii) 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 acceptable examples. The first example is a rendered in the default Neo4j distribution, the second has had some nodes moved to expose the edges more clearly. Both include examples of most node and edge types

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.

The longest conversation chain (by path length) is the set of chat items which respond to previous chat items. The longest chain in our graph is 9.

The steps to find the longest chain is to find all the paths with ResponseTo relationships between ChatItem nodes. Sort all the paths by their path lengths descending and select the first element in the list (the longest). Once you have that longest path, you can find the users who created the ChatItems in that particular path. The number of users in that particular chat is 5.

```
MATCH p=(:ChatItem)-[:ResponseTo*]->(:ChatItem)

WITH p as Path

ORDER BY LENGTH(p) DESC LIMIT 1

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

WHERE ci IN NODES(Path)

RETURN COUNT(DISTINCT u);
```

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.

Chattiest Users

Users	Number of Chats
394	115
2067	111
209	109

Chattiest Teams

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.

User 999 is one of the chattiest users is in one of the top 10 chattiest teams (team 52). In general, the chattiest users are **not** part of the chattiest teams.

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.

Users are active if their neighbors are active with each other. Users are less active if they have no neighbors or their neighbors are not active with eachother.

To find out active a group of users is, we find all the “neighbors” of a user. Given User 1, we first need to find all users who User 1 interacted with. These users are called “neighbors”. We then determine how active those users are between eachother using a “cluster coefficient”. If two neighbor users interact with eachother, they are considered “active”. The more active neighbors are, the higher the cluster coefficient.

Most Active Users (based on Cluster Coefficients)

User ID	Coefficient
209	0.9523809523809523
554	0.9047619047619048
1087	0.8