TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

VIỆN TRÍ TUỆ NHÂN TẠO

BÁO CÁO MÔN HỌC KỸ THUẬT VÀ CÔNG NGHỆ DỮ LIỆU LỚN:

Phân tích Chat Data bằng cơ sở dữ liệu Neo4j

Giảng viên hướng dẫn:

TS. Trần Hồng Việt

ThS. Ngô Minh Hương

Nhóm sinh viên thực hiện:

- Đàm Thái Ninh - 22022522
- Long Trí Thái Sơn - 22022653
- Hoàng Đăng Khoa - 22022548

Hà Nội - Việt Nam

05/2024

Mục lục

1 Big Data và Graph Databases

1.1 Định nghĩa Big Data

Big Data là thuật ngữ chỉ các tập dữ liệu có khối lượng lớn và phức tạp. Độ lớn đến mức các phần mềm xử lý dữ liệu truyền thống không có khả năng thu thập, quản lý và xử lý dữ liệu trong một khoảng thời gian hợp lý.

Các tập dữ liệu lớn này có thể có cấu trúc, không có cấu trúc và bán cấu trúc, cần được khai thác bằng các công nghệ xử lý Big Data để tìm hiểu insights. Một trong các công nghệ đó là Apache Spark được trình bày sau đây.

1.2 Giới thiệu về Graph Databases

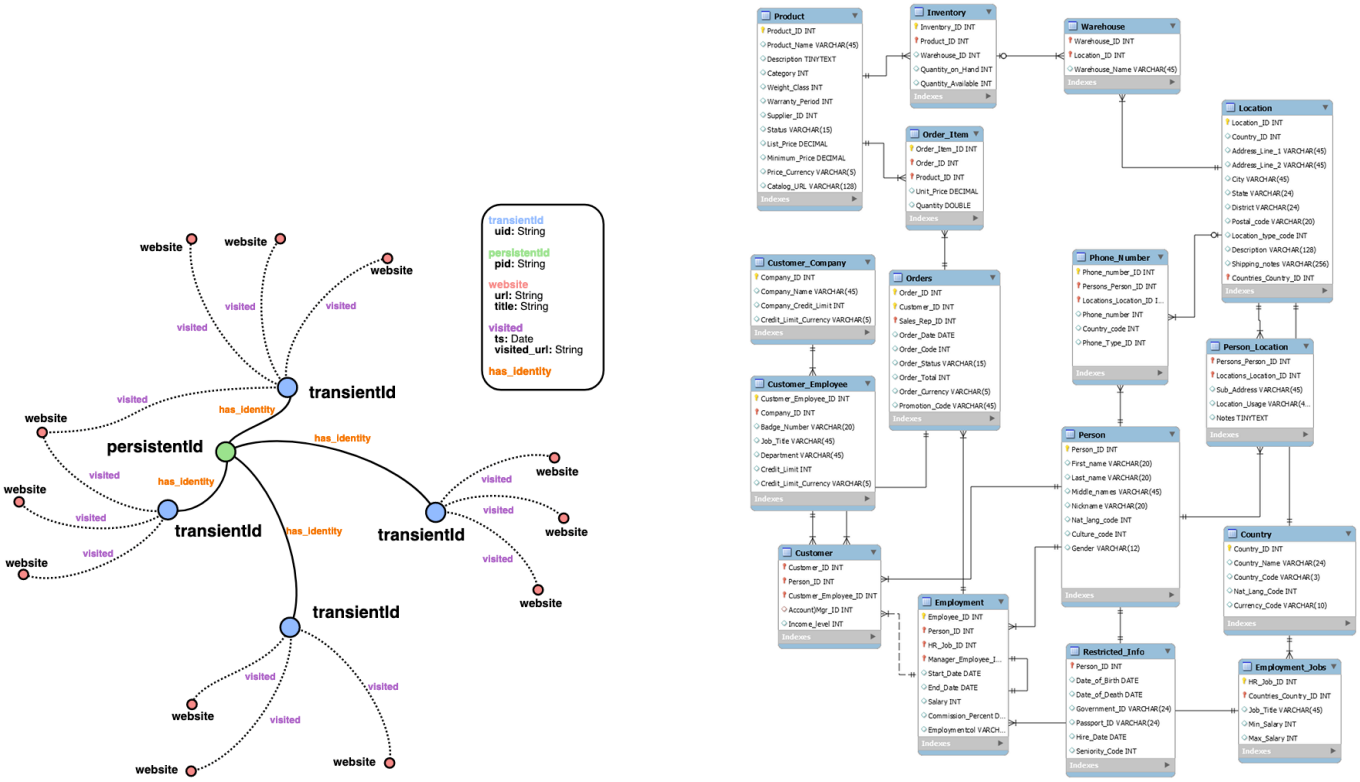
Trái lại, cơ sở dữ liệu đồ thị sử dụng cấu trúc đồ thị với các thuộc tính, biên và nút để biểu thị dữ liệu. Các nút là các đối tượng, còn các biên thể hiện mối quan hệ giữa các nút đó, và các thuộc tính mô tả thuộc tính của các nút và biên. Cấu trúc động này giúp cơ sở dữ liệu đồ thị hữu ích trong việc biểu thị dữ liệu được kết nối. Nó mang lại tính linh hoạt cao hơn về các mối quan hệ và kiểu dữ liệu.

Ví dụ: Dữ liệu cho ứng dụng mạng xã hội từ phần trước bây giờ sẽ được biểu thị như sau:

```
{
  customer_id: ‘‘C1’’,
  name: ‘‘Alejandro’’,
  location: ’’USA’’,
  friends: ‘‘C2,C3’’
}
```

Không còn sự trùng lặp hoặc dư thừa các bản ghi dữ liệu khi lập mô hình các mối quan hệ.

1.3 Điểm khác biệt chính giữa cơ sở dữ liệu đồ thị và cơ sở dữ liệu quan hệ



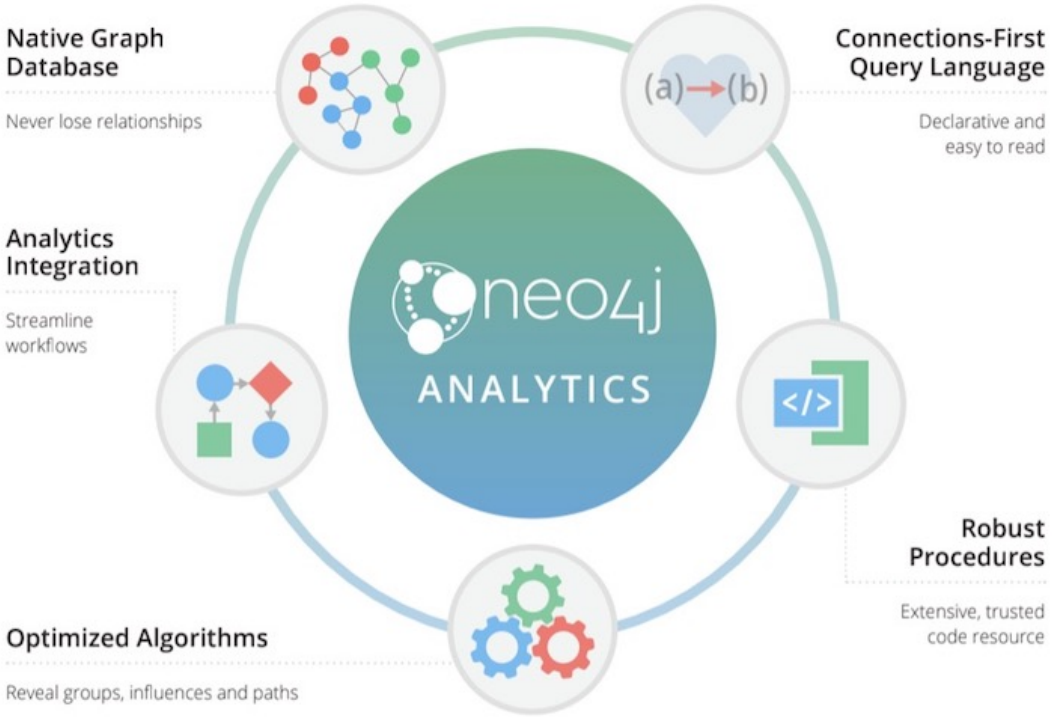
(a) Cơ sở dữ liệu đồ thị (b) Cơ sở dữ liệu quan hệ

Hình 1: Minh họa về hai loại cơ sở dữ liệu

- **Vận hành**
  - Cơ sở dữ liệu đồ thị sử dụng các thuật toán duyệt đồ thị để truy vấn dữ liệu. Các thuật toán này ưu tiên chiều sâu hoặc ưu tiên chiều rộng, điều này giúp nhanh chóng tìm và truy xuất dữ liệu được kết nối ⇒ Hữu ích cho các kết nối và truy vấn phức tạp vì chung có thể hiểu mối quan hệ giữa dữ liệu.
  - Cơ sở dữ liệu quan hệ sử dụng SQL để truy xuất và thao tác trên các bảng ⇒ Hiệu quả trong việc thực hiện các thao tác lọc, tổng hợp và kết nối phức tạp trên nhiều bảng.
- **Khả năng mở rộng**
  - Cơ sở dữ liệu quan hệ thường phải điều chỉnh quy mô theo chiều dọc (như nâng cấp CPU, bộ nhớ,...) vì việc phân phối dữ liệu quan hệ trên nhiều máy chủ sẽ làm tăng độ phức tạp của kho lưu trữ dữ liệu và có thể dẫn đến tính nhất quán. Điều chỉnh quy mô theo chiều dọc có thể tạo ra nhiều thách thức bên cạnh yêu cầu chi phí.
  - Ngược lại, cơ sở dữ liệu đồ thị rất phù hợp cho việc điều chỉnh quy mô theo chiều ngang và sử dụng phân vùng để điều chỉnh quy mô. Các phân vùng đều nằm trên các máy chủ khác nhau, cho phép nhiều máy chủ xử lý song song các truy vấn đồ thị. Bằng cách phân phối trên nhiều nút, công cụ cơ sở dữ liệu có thể truy vấn dữ liệu một cách hiệu quả, kể cả trên quy mô lớn.

- **Hiệu năng**
  - Cơ sở dữ liệu đồ thị cung cấp tính liền kề không có chỉ mục, giúp tăng hiệu năng. Tính liền kề không có chỉ mục cho phép hệ thống duyệt giữa các thực thể liên quan và các mối quan hệ trong thời gian không đổi, bất kể kích thước dữ liệu.
  - Cơ sở dữ liệu quan hệ sử dụng tra cứu chỉ mục và phải quét các bảng để xác định mối quan hệ giữa các thực thể. Bạn có thể kết nối nhiều bảng, nhưng việc này rất tốn thời gian vì hệ thống phải quét lượng chỉ mục lớn hơn trên nhiều dữ liệu hơn. Do đó, cơ sở dữ liệu quan hệ không đem lại hiệu năng giống như cơ sở dữ liệu đồ thị.
- **Dễ sử dụng**
  - Cơ sở dữ liệu đồ thị tập trung vào mối quan hệ, giúp bạn dễ dàng làm việc với chúng khi bạn sử dụng dữ liệu được kết nối. Các cơ sở dữ liệu này vượt trội trong các truy vấn nhiều bước nhảy, trong đó bạn duyệt các đường dẫn với nhiều mối quan hệ. Các ngôn ngữ truy vấn đồ thị thường được dùng có thể kể đến như Gremlin hoặc Cypher (sẽ nói chi tiết ở phần sau).
  - Cơ sở dữ liệu quan hệ sử dụng SQL, điều này có thể khiến bạn cảm thấy không tự nhiên khi quản lý các truy vấn nhiều bước nhảy. Một truy vấn có nhiều phép kết nối và bao gồm các truy vấn con lồng nhau sẽ rất cồng kềnh, khó đọc và khó duy trì.

2 Tổng quan về Neo4j

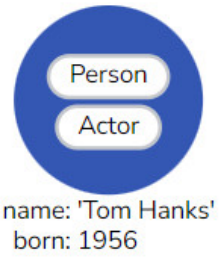


Neo4j là một loại hệ quản trị cơ sở dữ liệu đồ thị, được giới thiệu đầu tiên vào năm 2007 và công bố phiên bản 1.0 vào năm 2010.

2.1 Các thành phần chính của Neo4j

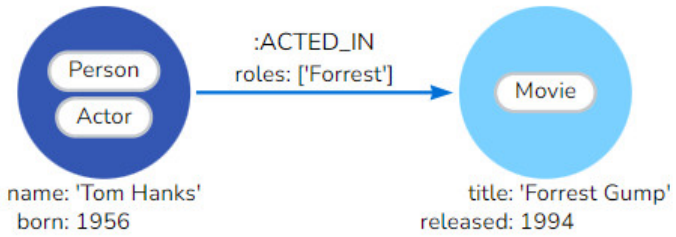
Khác với việc mô tả đối tượng (Subjects) và các đặc điểm của đối tượng (Properties) bằng các bảng dữ liệu như trong các cơ sở dữ liệu quan hệ, Neo4j mô tả đối tượng, quan hệ giữa các đối tượng và thuộc tính của chúng bằng đồ thị với các thành phần chính như sau:

- **Nodes (đỉnh):**
  - Đỉnh của đồ thị được sử dụng để biểu diễn các thực thể (entities).
  - Miền giá trị của các đỉnh có thể được mô hình hóa bằng Labels. Thông thường, chúng ta sẽ gộp nhóm các node có cùng kiểu dữ liệu thành một tập hợp rồi sau đó gắn label cho chúng. Một đỉnh có thể có nhiêu label.
  - Đồ thị đơn giản nhất là đồ thị mà trong đó chỉ có duy nhất một đỉnh.



Hình 2: Ví dụ một Node với 2 label và 2 thuộc tính

- **Relationships (quan hệ)**
  - Cạnh của đồ thị được sử dụng để biểu diễn mối quan hệ giữa các thực thể.
  - Các cạnh có thể mang theo các thuộc tính để mô tả thêm về mối quan hệ.
  - Tất cả các quan hệ đều phải có một và chỉ một kiểu quan hệ.



Hình 3: Ví dụ về quan hệ giữa 2 Node thuộc kiểu ACTED_IN

- **Properties (thuộc tính):**
  - Các thuộc tính được gắn vào đỉnh và cạnh để mô tả chi tiết hơn về các thực thể và mối quan hệ giữa chúng.
  - Thuộc tính được biểu diễn dưới dạng các cặp name-value và có thể lưu trữ các kiểu dữ liệu đa dạng khác nhau như number, string, boolean,...

2.2 Ngôn ngữ truy vấn Cypher

Cypher là ngôn ngữ truy vấn đồ thị của Neo4j, nó cho phép người dùng lưu trữ và truy xuất dữ liệu từ cơ sở dữ liệu đồ thị.. Cypher dựa trên nghệ thuật ASCII, vì vậy cú pháp của nó dễ hiểu và làm cho các truy vấn dễ hiểu hơn. Nó tập trung vào việc diễn đạt rõ ràng những gì cần truy xuất từ một biểu đồ, chứ không phải về cách truy vấn nó.

Cypher	SQL
Graph	Database
Node Label/Relationship	Table
Node	Row
Property	Column
MATCH...RETURN	SELECT
CREATE	INSERT
MATCH (node) SET (property)	UPDATE (table) SET (column)
MATCH (x:node)-[r:relationship]-(y:node) DELETE r	(delete a relationship) <=> DELETE FROM r WHERE r.x_id = x.id AND r.y_id = y.id
MATCH (x:node {property: "abc"}) DELETE x	(delete a node) <=> DELETE FROM x WHERE x.property = "abc"
MATCH (x:node {property: "abc"}) DETACH DELETE x	(delete node x and this relationship)

Hình 4: Tương quan giữa phương pháp truy vấn cơ bản của Cypher và SQL

Định dạng của Cypher

- **Định dạng Nodes:**
  - Các Nodes trong Cypher được ký hiệu bằng ngoặc tròn. VD: (varname:NodeName) là Node có nhãn là NodeName, tên biến là varname.
  - Truy vấn clean: Tên nhãn trong Cypher thường được đặt theo kiểu CamelCase (viết hoa chữ cái đầu), mang tính gợi nhớ tới đối tượng. Nhãn Node nên là một danh từ.
  - Tên nhãn có phân biệt chữ hoa chữ thường.
- **Định dạng Relationships (quan hệ):**
  - Các quan hệ trong Cypher được ký hiệu bằng ngoặc vuông. VD: [varname:RELATIONSHIP_NAME] là mối quan hệ có kiểu quan hệ là RELATIONSHIP_NAME và biến quan hệ là varname.
  - Truy vấn clean: Kiểu quan hệ thường được đặt theo kiểu UpperCase (tất cả viết hoa) và sử dụng dấu gạch nối giữa các từ. Kiểu quan hệ nên là động từ.
  - Kiểu quan hệ có phân biệt chữ hoa chữ thường.
- **Khóa thuộc tính, biến, tham số, bí danh và hàm:**
  - Ví dụ: title, size(), count(), firstName...
  - Thường được viết theo kiểu CamelCase, có phân biệt chữ hoa chữ thường
- **Mệnh đề và Keyword:**

- Ví dụ: MATCH, WHERE, WITH, UNWIND, MERGE,...
- Nên được viết in hoa, không cần đặt ở đầu dòng mới.
- Các keyword không phân biệt chữ hoa chữ thường

Ví dụ về các thao tác CRUD (Create, Read, Update, Delete) trong Cypher

Sau đây là ví dụ các thao tác trong CRUD của Cypher cùng với câu lệnh tương tự trong SQL để đối chiếu.

- **Tạo data mới (Create):** Ví dụ câu lệnh tạo node với label User và gán giá trị cho các thuộc tính first_name, last_name, email. Tương tự với INSERT một hàng vào một bảng trong SQL.
  - Bằng Cypher:

```
CREATE (:User {
    first_name: "John",
    last_name: "Doe",
    email: "john.doe@email.com"
})
```

  - Câu lệnh tương tự trong SQL:

```
INSERT INTO users (first_name, last_name, email)
VALUES ('John', 'Doe', 'john.doe@email.com');
```

- **Truy vấn data (Read):** Ví dụ câu lệnh truy vấn một node với label User có thuộc tính email là "john.doe@email.com". Tương tự với SELECT một hàng trong một bảng dựa vào điều kiện trong SQL.
  - Bằng Cypher:

```
MATCH (u:User {email: "john.doe@email.com"})
RETURN u
```

  - Câu lệnh tương tự trong SQL:

```
SELECT *
FROM users
WHERE email = 'john.doe@email.com';
```

- **Sửa đổi data (Update)** Ví dụ câu lệnh tìm một node với label User có thuộc tính email là "john.doe@email.com"và sửa thuộc tính first_name của nó thành "Johny". Tương tự với việc Update một giá trị trong một hàng trong SQL.
  - Bằng Cypher:

```
MATCH (u:User {email: "john.doe@email.com"})
SET u.first_name = "Johnny"
RETURN u
```

– Câu lệnh tương tự trong SQL:

```
UPDATE users
SET first_name = 'Johnny'
WHERE email = 'john.doe@email.com';
```

- **Xóa data (Delete):** Ví dụ câu lệnh xóa node với label User có email là "john.doe@email.com"và xóa nó khỏi đồ thị. Tương tự với việc Delete một hàng trong bảng của SQL.

  – Bằng Cypher:

```
MATCH (u:User {email: "john.doe@email.com"})
DELETE u
```

  – Câu lệnh tương tự trong SQL:

```
DELETE FROM users
WHERE email = 'john.doe@email.com';
```

3 Ứng dụng Neo4j để phân tích bộ dữ liệu Chat Data

3.1 Tổng quan về bộ dữ liệu

Bộ dữ liệu Chat Data cung cấp thông tin về tương tác của người dùng trong các phiên trò chuyện nhóm. Bộ dữ liệu bao gồm 6 file csv với các cột như sau:

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

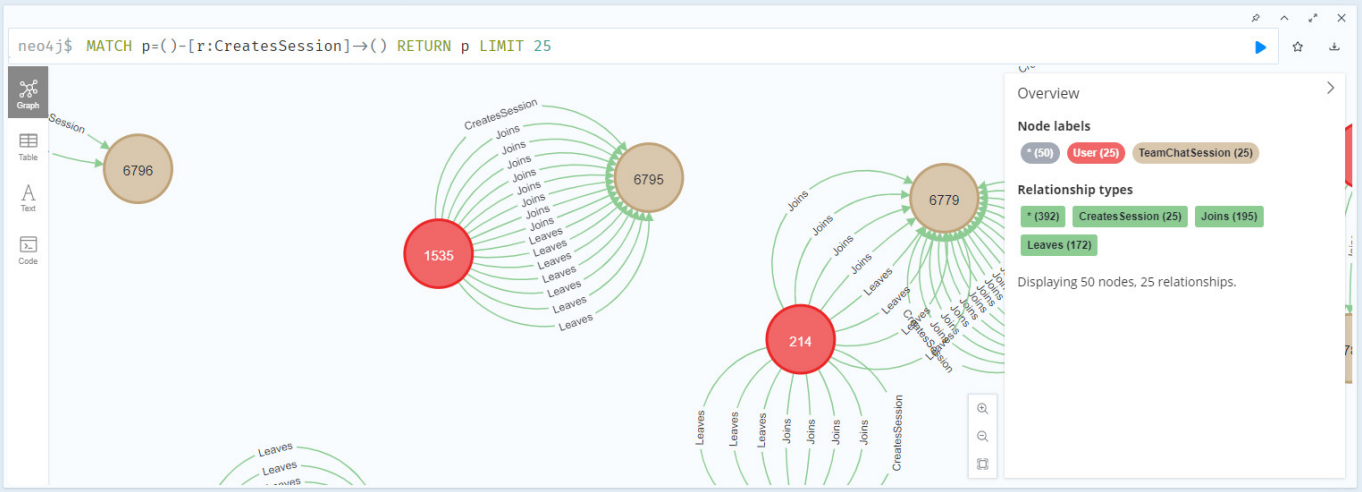
Hình 5: Thông tin về 6 file csv trong bộ dữ liệu Chat Data

3.2 Mô hình bộ dữ liệu dưới dạng Graph

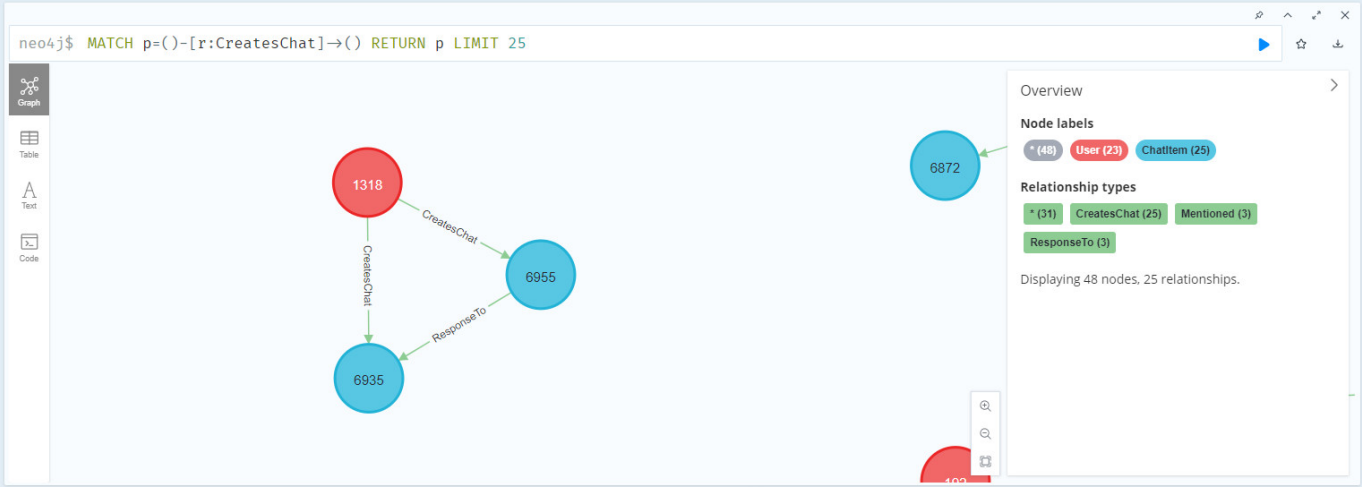
Bộ dữ liệu Chat Data sẽ được mô hình dưới dạng Graph như sau:

- Sử dụng 4 Label (User, TeamChatSession, ChatItem, Team) cho các Node để biểu diễn 4 đối tượng chính trong bộ dữ liệu.
- Sử dụng 8 kiểu quan hệ được mô tả trong các file:
  - User **CreatesSession** TeamChatSession
  - User **Joins** TeamChatSession
  - User **Leaves** TeamChatSession
  - User **CreatesChat** ChatItem
  - TeamChatSession **OwnedBy** Team
  - ChatItem **Mentioned** User
  - ChatItem **PartOf** TeamChatSession
  - ChatItem **ResponseTo** ChatItem

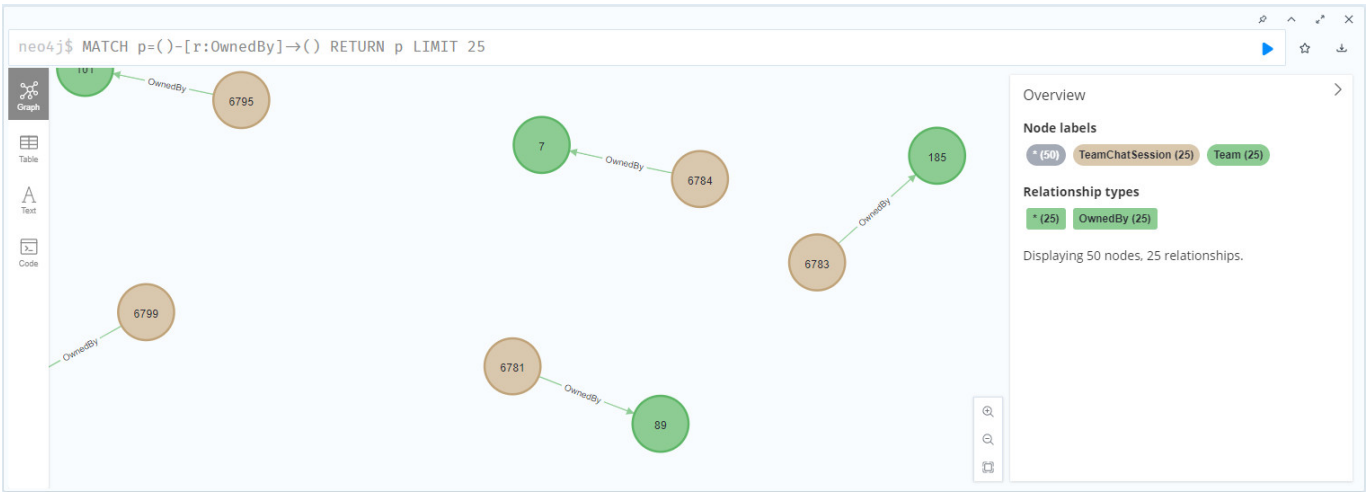
Một số hình ảnh của đồ thị



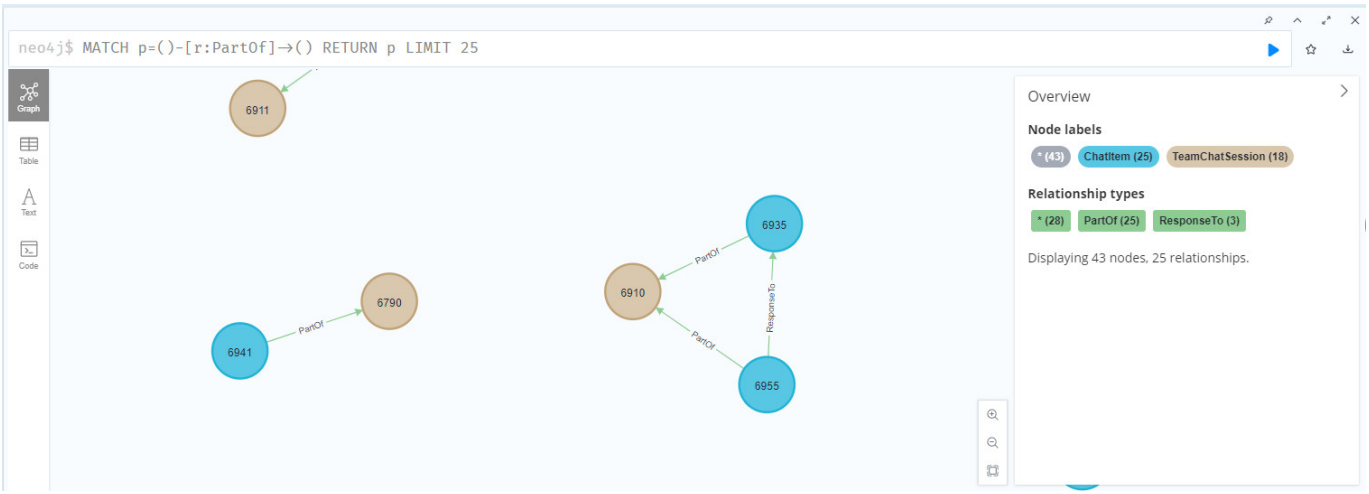
Hình 6: User **CreatesSession, Joins, Leaves** TeamChatSession



Hình 7: User **CreatesChat** ChatItem; ChatItem **ResponseTo** ChatItem; ChatItem **Mentioned** User



Hình 8: TeamChatSession **OwnedBy** Team

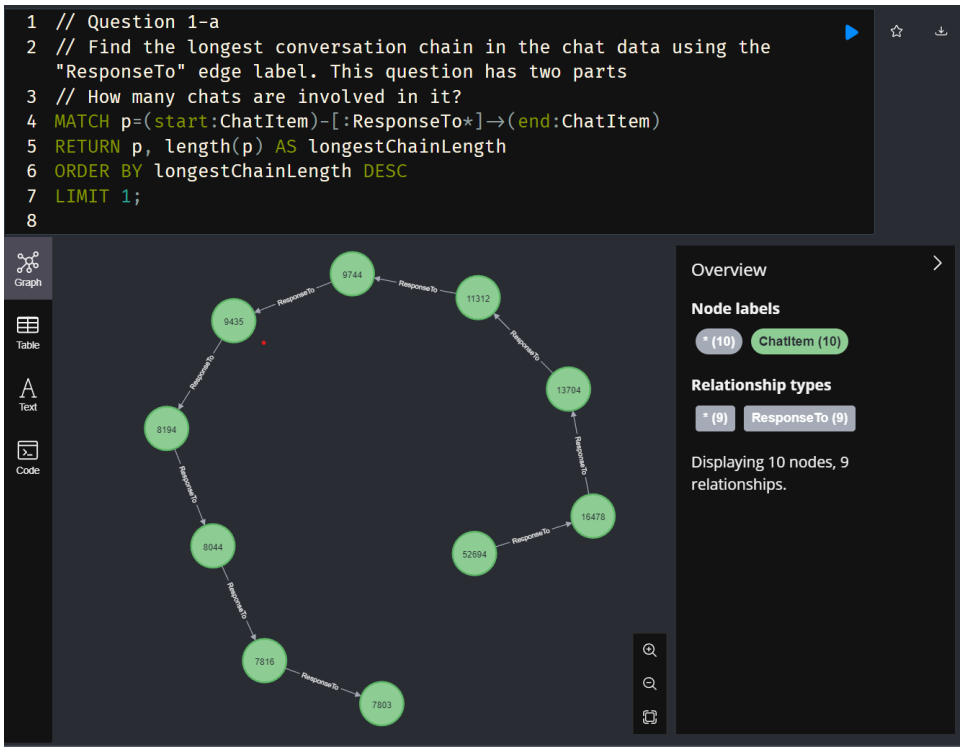


Hình 9: ChatItem **PartOf** TeamChatSession

3.3 Phân tích dữ liệu

Chúng tôi phân tích dữ liệu thông qua việc trả lời 3 câu hỏi sau đây:

1. Tìm chuỗi hội thoại dài nhất (sử dụng kiểu quan hệ ResponseTo). Có bao nhiêu ChatItem trong chuỗi? Có bao nhiêu User tham gia chuỗi hội thoại đó



Hình 10: Chuỗi hội thoại dài nhất

Hình 11: Số lượng Users tham gia vào chuỗi hội thoại này

2. Tìm top 10 Users và Teams tạo nhiều chat nhất.

```
// Chattiest Users

// Query to find the top 10 chattiest users
MATCH (u:User)-[:CreatesChat]→(c:ChatItem)
RETURN u.id AS userId, count(c) AS chatCount
ORDER BY chatCount DESC
LIMIT 10;
```

	userId	chatCount
1	394	115
2	2067	111
3	1087	109
4	209	109
5	554	107
6	1627	105

Started streaming 10 records after 7 ms and completed after 40 ms.

Hình 12: Top 10 Users tạo nhiều chat nhất

```
// Q2-b

// Chattiest Teams

// Query to find the top 10 chattiest teams
MATCH (ci:ChatItem)-[:PartOf]→(tcs:TeamChatSession)-[:OwnedBy]→
(t:Team)
RETURN t.id AS teamId, count(ci) AS chatCount
ORDER BY chatCount DESC
LIMIT 10;
```

	teamId	chatCount
1	82	1324
2	185	1036
3	112	957
4	18	844
5	194	836
6	129	814

Hình 13: Top 10 Teams tạo nhiều chat nhất

3. Tìm những người hoạt động tích cực nhất.

```
MATCH
(u:User)-[c:CreatesChat]→()
WITH u, COUNT(c) as Chats
ORDER BY Chats DESC LIMIT 10 WITH [u] as ChattiestUsers
//Getting the neighbours of all Users and the count
MATCH (u1:User)-[:InteractsWith]→(u2:User)
WHERE u1 in ChattiestUsers
WITH u1.id AS UserID, COLLECT(DISTINCT u2.id) AS Neighbours RETURN
UserID, Neighbours, SIZE(Neighbours) AS k
```

	UserID	Neighbours	k
1	394	[1012, 2011, 1997, 1782]	4
2	2067	[63, 516, 1265, 1672, 209, 1627]	6
3	1087	[1311, 426, 929, 772, 1879, 1098]	6
4	209	[63, 516, 1627, 2067, 1672]	5
5	554	[2018, 1959, 1687, 1096, 1010]	5
6	1627	[516, 2067, 209, 63, 1672, 1265]	6
7			

Started streaming 10 records after 11 ms and completed after 74 ms.

Hình 14: Danh sách những người tương tác ứng với từng người

```
// Getting TOP 10 Chattiest Users
MATCH (u:User)-[c:CreatesChat]→()
WITH u, COUNT(c) AS Chats
ORDER BY Chats DESC
LIMIT 10
WITH COLLECT(u) AS ChattiestUsers

// Getting the neighbours of TOP 10 Users and the count
MATCH (u1:User)-[:InteractsWith]→(u2:User)
WHERE u1 IN ChattiestUsers
WITH u1.id AS UserID, COLLECT(DISTINCT u2.id) AS Neighbours,
SIZE(COLLECT(DISTINCT u2.id)) AS k
```

	UserID	NUM	DENUM	ClusteringCoefficient
1	668	6	6	1.0
2	209	20	20	1.0
3	999	53	56	0.9464285714285714

Hình 15: Top những người hoạt động tích cực nhất

4 Kết luận

4.1 Kết luận chính

Neo4j, một cơ sở dữ liệu đồ thị hàng đầu, đã chứng minh sức mạnh của mình trong việc xử lý và quản lý dữ liệu đồ thị. Với kiến trúc đồ thị linh hoạt và hiệu suất cao, Neo4j không chỉ là một công cụ lưu trữ dữ liệu mà còn là một nền tảng mạnh mẽ cho việc phân tích dữ liệu và trí tuệ nhân tạo.

Một trong những điểm đáng chú ý của Neo4j là khả năng xử lý các tương tác phức tạp giữa các thực thể trong dữ liệu, cho phép phân tích các mô hình quan hệ phức tạp và trích xuất thông tin quan trọng từ các mạng liên kết phức tạp.

Bên cạnh đó, sự tích hợp của Neo4j với các công nghệ như Spark và PySpark mở ra những cơ hội mới trong việc xử lý dữ liệu lớn và phân tán. Kết hợp sức mạnh của Neo4j với các khả năng tính toán và xử lý song song của Spark mang lại hiệu suất cao và khả năng mở rộng linh hoạt cho các ứng dụng dựa trên dữ liệu đồ thị.

Tóm lại, Neo4j không chỉ là một công cụ lưu trữ dữ liệu đồ thị mạnh mẽ mà còn là một nền tảng đa năng cho việc phân tích dữ liệu, trích xuất thông tin quan trọng và xây dựng các ứng dụng thông minh dựa trên dữ liệu đồ thị.

4.2 Hướng phát triển

Dự định kết hợp Neo4j và Spark để xử lí các đồ thị lớn hơn.

Neo4j là một cơ sở dữ liệu đồ thị nổi tiếng có khả năng xử lý các tương tác phức tạp giữa các mục dữ liệu một cách vượt trội. Để xây dựng các kỹ thuật và thuật toán xử lý đồ thị về tầm quan trọng của nút, trích xuất đồ thị con, phân cụm và dự đoán liên kết, chúng tôi sẽ đặc biệt tập trung vào việc sử dụng Neo4j trên Spark và PySpark.

Các biểu đồ lớn có thể được lưu trữ và xử lý hiệu quả bằng cách sử dụng cơ sở dữ liệu phân tán như Neo4j trên Spark và PySpark. Hơn nữa, các phương pháp tiên tiến như bộ nhớ đệm, phân tách dữ liệu, xử lý song song và chiến thuật tối ưu hóa có thể cải thiện tốc độ xử lý đồ thị phân tán hơn nữa. Sự tích hợp của Neo4j với Spark và PySpark mang đến những cơ hội mới để trích xuất nhanh chóng và hiệu quả thông tin chi tiết từ các bộ dữ liệu biểu đồ được liên kết phức tạp khi chúng tôi tiếp tục vượt qua các giới hạn của các ứng dụng dựa trên dữ liệu.

Nhiệm vụ của các thành viên

Họ và tên	**Công Việc**
Đàm Thái Ninh	+ Làm báo cáo + Tìm hiểu về Graph Databases + Viết Query trả lời câu hỏi 2, 3 + Làm Slide thuyết trình
Long Trí Thái Sơn	+ Làm báo cáo + Viết Query trả lời câu hỏi 1 + Tìm hiểu về Neo4j Connector cho Apache Spark
Hoàng Đăng Khoa	+ Làm báo cáo + Tìm hiểu về Neo4j và Cypher + Viết Scrip load dữ liệu vào Neo4j + Làm Slide thuyết trình

Tài liệu tham khảo

[**1**] Neo4j Documentation. https://neo4j.com/docs/getting-started/

[**2**] Neo4j Connector for Apache Spark Documentation. https://neo4j.com/docs/getting-started/

[**3**] https://aws.amazon.com/vi/compare/the-difference-between-graph-and-relational-database/

[**4**] https://github.com/AlessandroCorradini/University-of-California-San-Diego-Big-Data-Specialization/tre
%20Big%20Data%20-%20Capstone%20Project/04%20-%20Graph%20Analytics%20With%20Chat%20Da