



Neo4j Luan-hoang - Các hệ cơ sở dữ liệu

Công tác kỹ sư (Trường Đại học Công nghệ Thành phố Hồ Chí Minh)

BỘ GIÁO DỤC VÀ ĐÀO TẠO



HUTECH
Đại học Công nghệ Tp.HCM

BÁO CÁO MÔN HỌC
CÁC HỆ CƠ SỞ DỮ LIỆU NÂNG CAO

NGHIÊN CỨU VÀ ỨNG DỤNG NEO4J

GVHD : PGS.TS. NGUYỄN THỊ THÚY LOAN

HVTH : Phạm Thành Luân – 2241860007

Võ Trọng Hoàng – 2241860020

TP. HCM, năm 2022

[illegible]

Giảng viên hướng dẫn

MỤC LỤC

Chương 1. Giới thiệu chung về cơ sở dữ liệu đồ thị.....	4
1.1 Đồ thị là gì.....	4
1.2 Dữ liệu đồ thị là gì.....	5
1.3 Vì sao nên chọn dữ liệu đồ thị.....	5
1.4 So sánh dữ liệu đồ thị và dữ liệu quan hệ.....	6
1.5 So sánh dữ liệu đồ thị và dữ liệu tài liệu.....	9
1.6 Ví dụ về dữ liệu đồ thị.....	10
Chương 2. Tìm hiểu về neo4j.....	13
2.1 Giới thiệu về Neo4j.....	13
2.2 Các thành phần chính của cơ sở dữ liệu Neo4j.....	15
2.3 Lợi ích của cơ sở dữ liệu Neo4j.....	15
2.4 Cài đặt Neo4j.....	16
Chương 3. Ngôn ngữ truy vấn cypher.....	22
3.1 Giới thiệu.....	22
3.2 Mô hình dữ liệu đồ thị.....	23
3.3 Cú pháp cơ bản.....	26
3.4 Các lệnh Cypher cơ bản.....	28
Chương 4. Lĩnh Vực ứng dụng.....	43
4.1 Ngăn chặn gian lận.....	43
4.2 Mạng và hoạt động công nghệ thông tin.....	44
4.3 Công cụ khuyến nghị.....	45
4.4 Các ứng dụng mạng xã hội.....	45
4.5 AI và phân tích.....	46
4.6 Những lĩnh vực sử dụng Neo4j nhiều nhất.....	47
Chương 5. Ứng dụng demo.....	48
5.1 Mục tiêu.....	48
5.2 Chức năng.....	48
5.3 Thiết kế cơ sở dữ liệu.....	50
5.4 Giao diện.....	50
Chương 6. Khó khăn và kết quả đạt được.....	52

6.1	Khó khăn.....	52
6.2	Kết quả đạt được.....	52
	Tài liệu tham khảo.....	53

DANH MỤC HÌNH

Hình 1.	Cấu trúc đồ thị
Hình 2.	Đồ thị mạng xã hội đơn giản
Hình 3.	Đồ thị mạng xã hội sau khi được mở rộng
Hình 4.	Dữ liệu đồ thị trong thể thao
Hình 5.	Thể hiện mối liên kết giữa các bảng bằng khóa ngoại
Hình 6.	Cấu trúc tài liệu
Hình 7.	Dữ liệu đồ thị tình hình nhân sự ở các công ty
Hình 8.	Dữ liệu đồ thị phân quyền
Hình 9.	Dữ liệu đồ thị thể hiện tuyến đường và các trạm trung chuyển trong TP

DANH MỤC BẢNG

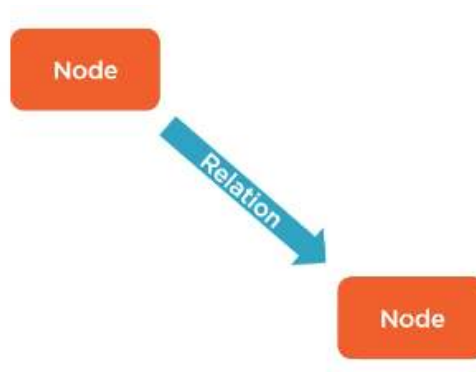
Bảng 1.	So sánh Dữ liệu đồ thị và Dữ liệu quan hệ
Bảng 2.	Kết quả thực nghiệm hiệu quả xử lý dữ liệu giữa Neo4j và Rel.Db
Bảng 3.	So sánh Dữ liệu đồ thị và Dữ liệu tài liệu

DANH MỤC THUẬT NGỮ, TỪ VIẾT TẮT

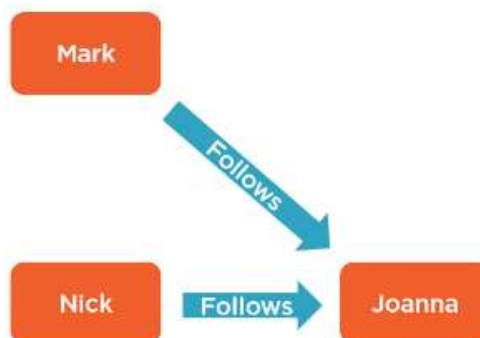
Ký hiệu	Diễn giải

CHƯƠNG 1. GIỚI THIỆU CHUNG VỀ CƠ SỞ DỮ LIỆU ĐỒ THỊ

1.1 Đồ thị là gì



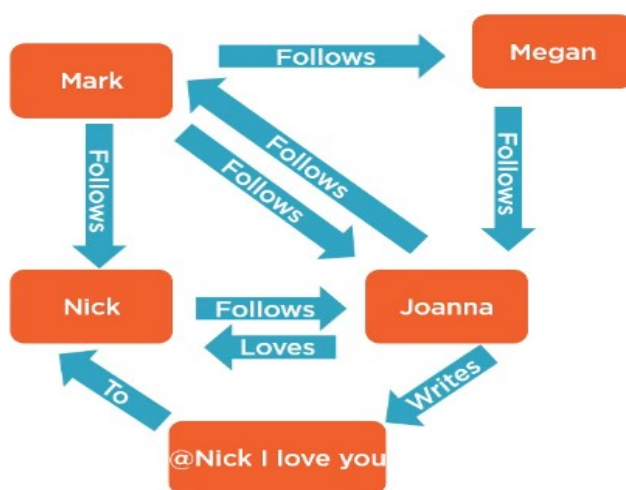
Hình 1. Cấu trúc đồ thị



Hình 2. Đồ thị mạng xã hội đơn giản

Đồ thị bao gồm nhiều node mà những node này được liên kết trực tiếp với nhau bằng những quan hệ. Mỗi node đại diện cho một thực thể, một thực thể có thể là bất cứ một thứ gì đó trong thực tế như một khách hàng, một giao dịch, một người, một đồ vật... Hình bên trên là ví dụ thực tế cho một đồ thị, các node và quan hệ khi kết hợp lại với nhau sẽ tạo nên một đồ thị.

Đồ thị có thể dễ dàng mở rộng bằng cách thêm vào các node và quan hệ mới. Đồ thị cho người sử dụng một cái nhìn tổng thể và thân thiện hơn các dạng khác.



Hình 3. Đồ thị mạng xã hội sau khi được mở rộng

Đồ thị có thể dễ dàng mở rộng bằng cách thêm vào các node và quan hệ mới. Đồ thị cho người sử dụng một cái nhìn tổng thể và thân thiện hơn các dạng khác.

1.2 Dữ liệu đồ thị là gì

Tương tự như cơ sở dữ liệu quan hệ, dữ liệu đồ thị cũng là dữ liệu nhưng sử dụng cấu trúc đồ thị để thể hiện và lưu trữ dữ liệu. Dữ liệu đồ thị có ưu thế so với các cấu trúc khác trong việc lưu trữ và truy vấn những thông tin có sự liên kết cao. Dù chứa rất nhiều

quan hệ trong cấu trúc dữ liệu nhưng qua những nghiên cứu và trải nghiệm dữ liệu đồ thị vẫn giữ được hiệu quả hoạt động cao dù có truy vấn lên đến hàng triệu node. Tính linh hoạt của dữ liệu đồ thị cao hơn dữ liệu quan hệ vì không bị ràng buộc bởi các khóa, chúng ta có thể thêm, sửa, xóa node và các thuộc tính của node mà không ảnh hưởng đến các node khác. Dữ liệu đồ thị không bị ràng buộc bởi ngôn ngữ truy vấn và lưu trữ dữ liệu, có nhiều ngôn ngữ được hỗ trợ để làm việc trên cơ sở dữ liệu đồ thị. Ngoài node đại diện cho các thực thể và quan hệ thể hiện sự liên kết giữa các node, dữ liệu đồ thị có thuộc tính cho cả node và quan hệ.



Hình 4. Dữ liệu đồ thị trong thể thao

1.3 Vì sao nên chọn dữ liệu đồ thị

Trước đây khi chưa có những cấu trúc dữ liệu mới chúng ta thường sử dụng dữ liệu quan hệ cho tất cả các ứng dụng vì sự phổ biến cũng như đa dụng. Bởi vì hầu như dữ liệu quan hệ có thể đáp ứng phần lớn các loại ứng dụng, điểm khác biệt lớn nhất làm cho các cấu trúc dữ liệu mới được chọn là hiệu quả sử dụng nhằm tiết kiệm chi phí và thời gian. Dữ liệu quan hệ vẫn hoạt động tốt trong một số ứng dụng nhưng nhược điểm sẽ thể hiện rõ khi truy vấn và xử lý những dữ liệu có tính liên

kết cao và đây là lúc dữ liệu đồ thị được cần đến. Do đó chúng ta cần lựa chọn cấu trúc dữ liệu cho phù hợp với mục đích công việc hoặc phần mềm được viết ra.

Qua những thông tin trên chúng ta sẽ thấy được 3 lý do quan trọng khi cần chọn dữ liệu đồ thị:

- Dữ liệu có tính liên kết cao
- Không bị ràng buộc bởi schema
- Dễ nhìn sơ đồ tổng quát của dữ liệu

1.4 So sánh dữ liệu đồ thị và dữ liệu quan hệ

Quan hệ	Đồ thị
Bảng	Node
Schema cho phép null	Không có schema
Quan hệ dựa trên khóa ngoại	Quan hệ được định nghĩa
Dữ liệu liên kết được đưa ra bằng phép kết	Dữ liệu liên kết dựa trên các mẫu

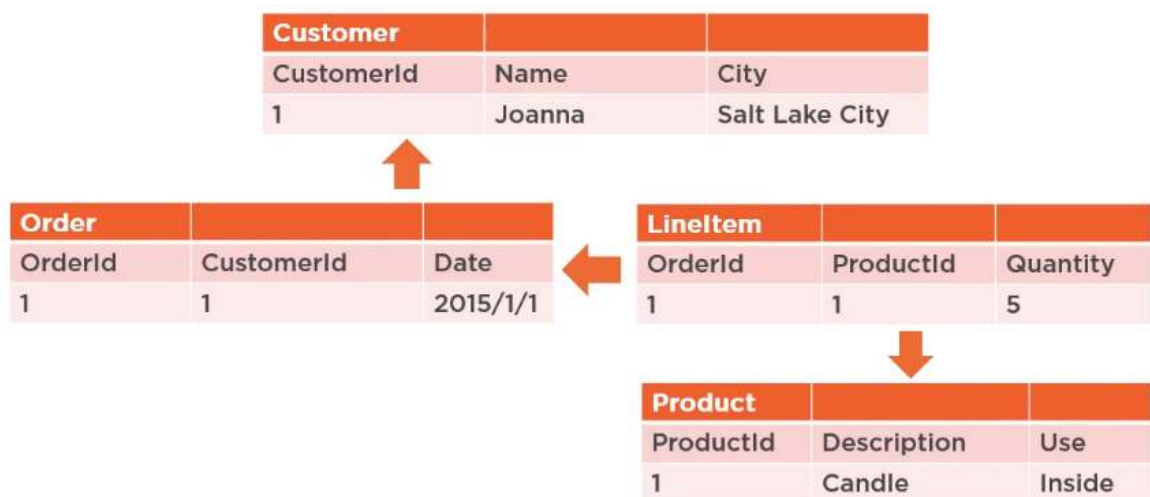
Bảng 1. So sánh Dữ liệu đồ thị và Dữ liệu quan hệ

Dữ liệu quan hệ:

- Dữ liệu quan hệ sử dụng bảng để chứa dữ liệu. Các bảng được định dạng dựa trên dòng và cột, mỗi cột là một thuộc tính của bảng, mỗi dòng sẽ chứa dữ liệu của mỗi cột thuộc tính.
- Tất cả các dữ liệu được đưa vào bảng phải tuân thủ chính xác những quy định của schema trên bảng đó. Nếu có bất kỳ dòng dữ liệu nào được thêm vào bảng mà dữ liệu không đúng với thuộc tính, dữ liệu tại cột đó có thể được gán giá trị null. Do đó nếu một thuộc tính không có giá trị trên một thực thể thì sẽ xuất hiện một thuộc tính có giá trị null. Các truy vấn hoặc ứng dụng khi chạy cần kiểm tra các giá trị null để

có kết quả chính xác và sẽ mất thời gian để duyệt qua các thuộc tính null.

- Quan hệ giữa các dòng trên các bảng khác nhau được tạo và thể hiện bằng các khóa ngoại, khóa ngoại cũng được xác định là một phần dữ liệu của bảng. Khóa ngoại của một bảng là một thuộc tính copy từ một thuộc tính duy nhất trên bảng có quan hệ.
- Để lấy dữ liệu giữ các bảng có liên quan với nhau, ta cần thực hiện các truy vấn có toán tử join. Nếu dữ liệu được lấy từ nhiều bảng khác nhau chúng ta cần thực hiện nhiều toán tử join. Truy vấn hoặc ứng dụng có càng nhiều toán tử join, hiệu quả xử lý càng bị ảnh hưởng.
- Dữ liệu quan hệ vẫn giữ được vai trò quan trọng đối với các dạng dữ liệu cần sự tổ chức cao, có nhiều tính toán trong bảng, xây dựng các bảng báo cáo.



Hình 5. Thể hiện mối liên kết giữa các bảng bằng khóa ngoại

Dữ liệu đồ thị:

- Dữ liệu đồ thị sử dụng các node để lưu trữ dữ liệu, các node không bị ràng buộc bởi bảng hay bất cứ định dạng nào. Các node được tạo và định dạng hoàn toàn tự do.
- Node thể hiện cho các thực thể, mỗi thực thể có thuộc tính riêng biệt. Đó đó chúng ta có thể thêm hoặc xóa thuộc tính của một thực thể mà không ảnh hưởng đến các thuộc tính hoặc thực thể khác. Nếu một thuộc tính chỉ có trên thực thể này mà không có trên thực thể khác, thuộc tính đó có thể xóa trên những thực thể không có giá trị. Từ đó, dữ liệu đồ thị sẽ không tồn tại những giá trị null, các truy vấn và xử lý sẽ giảm được thời gian cho việc tính toán.
- Quan hệ trong dữ liệu đồ thị được định nghĩa riêng như một thuộc tính giữa các node. Mỗi quan hệ trong dữ liệu đồ thị cũng có tầm quan trọng như các node và tách biệt hoàn toàn với nhau nhờ đó mà dữ liệu đồ thị hạn chế được sự lặp lại dữ liệu.
- Ngôn ngữ truy vấn trong dữ liệu đồ thị được tối ưu hóa để truy vấn các mối quan hệ, những dữ liệu mang tính liên kết cao. Truy vấn trong dữ liệu đồ thị hoạt động theo cách đi tìm mẫu, nếu mẫu đưa ra trong truy vấn khớp với dữ liệu, dữ liệu sẽ được trả về. Truy vấn node liên kết đến các node khác để thực hiện nhưng cần nhiều thời gian để thực thi nhưng thời gian thực thi vẫn tốt hơn so với việc thực hiện toán tử join trên dữ liệu quan hệ.

Jonas Partner and Aleksa Vukotic đã viết một quyển sách có tên Neo4j, trong đó có một thực nghiệm để so sánh hiệu quả xử lý dữ liệu giữa dữ liệu đồ thị và dữ liệu quan hệ như sau:

- Một mạng xã hội
- Bao gồm bạn của những người bạn

- So sánh mySql và Neo4j
- 1.000.000 người được tạo ra trong dữ liệu
- Mỗi người trong 1.000.000 người trên có khoảng 50 người bạn
- Liên hệ 2 bước: tìm bạn của những người bạn của một người trong dữ liệu trên
- Liên hệ 3: bước: tìm bạn của những người bạn, của những người bạn của một người trong dữ liệu trên và tăng dần số bước liên hệ

Thực nghiệm trên cho ra kết quả như bảng sau:

Depth	Rel. Db (s)	Neo4j (s)	# records
2	0,016	0,01	~2500
3	30,267	0,168	~110000
4	1543,505	1,359	~600000
5	Unfinished	2,132	~8000000

Bảng 2. Kết quả thực nghiệm hiệu quả xử lý dữ liệu giữa Neo4j và Rel.Db

Kết quả này không chứng tỏ cho việc dữ liệu quan hệ xử lý kém hơn dữ liệu đồ thị mà dữ liệu quan hệ không phù hợp xử lý dữ liệu có sự kết nối và liên kết cao.

1.5 So sánh dữ liệu đồ thị và dữ liệu tài liệu

Tài liệu	Đồ thị
Tài liệu	Node
Không có schema	Không có schema
Quan hệ dựa trên khóa ngoại hoặc cấu trúc nhúng	Quan hệ được định nghĩa
Dữ liệu liên kết được đưa ra bằng phép kết hoặc cấu trúc nhúng	Dữ liệu liên kết dựa trên các mẫu

Bảng 3. So sánh Dữ liệu đồ thị và Dữ liệu tài liệu

Ví dụ về dữ liệu tài liệu:



Hình 6. Cấu trúc tài liệu

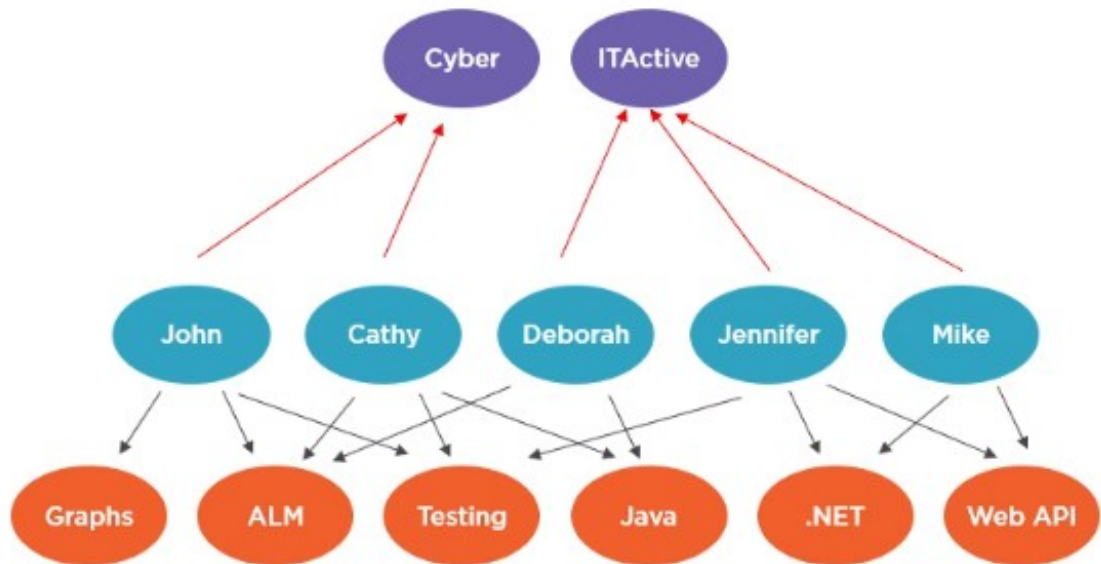
Toàn bộ dữ liệu về một giao dịch của khách hàng được hiển thị trong một tài liệu. Trong ví dụ này tài liệu được lưu trữ trong kho tài liệu khách hàng. Dữ liệu tài liệu được khuyến khích sử dụng cho các dữ liệu không có sự chuẩn hóa. Dữ liệu tài liệu hoạt động tốt nhất khi các dữ liệu liên quan và các thuộc tính của một thực thể được lưu trữ trong một tài liệu. Dữ liệu tài liệu cần có tài liệu gốc để thực hiện việc truy vấn, như ví dụ bên trên sản phẩm cần được lưu trữ trong một kho riêng để lấy thông tin lên tài liệu khách hàng.

Dữ liệu tài liệu sẽ có sự lặp lại dữ liệu tương tự như dữ liệu quan hệ, khi một sản phẩm được đưa vào hóa đơn của một khách hàng, sản phẩm đó được copy từ sản phẩm vào tài liệu khách hàng.

Tương tự như dữ liệu quan hệ, dữ liệu tài liệu cũng không phù hợp với những dữ liệu mang tính liên kết cao. Nhưng lại phù hợp với những dạng văn bản như lưu trữ mã code của một ứng dụng.

1.6 Ví dụ về dữ liệu đồ thị

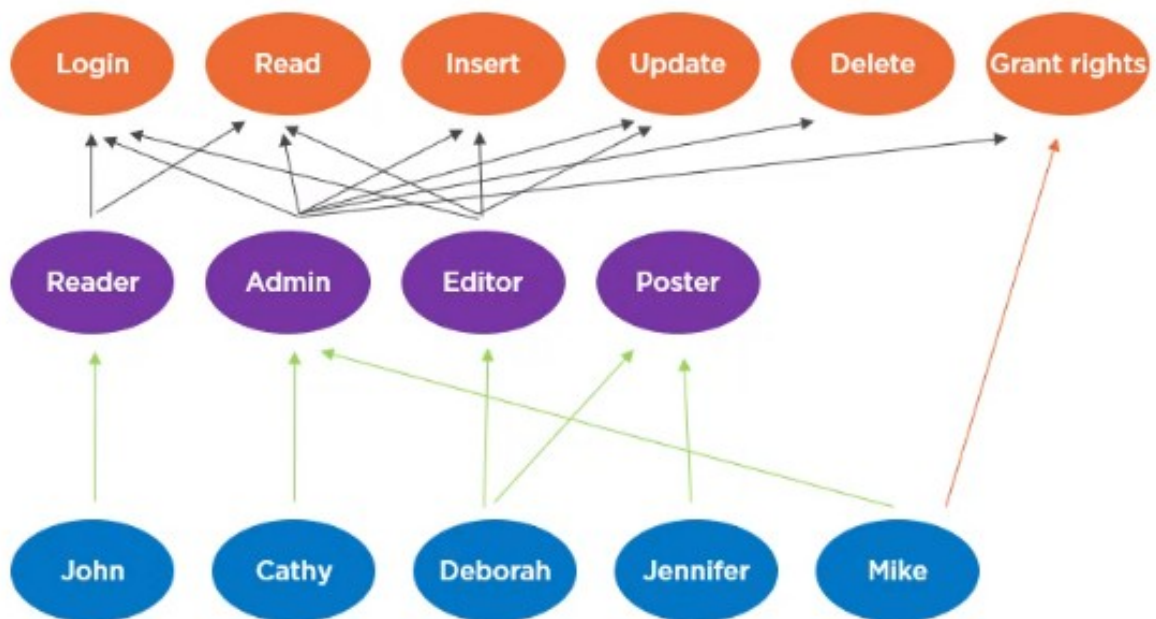
Ví dụ 1:



Hình 7. Dữ liệu đồ thị tình hình nhân sự ở các công ty

Ví dụ này có 3 lớp thực thể: đầu tiên là công ty, thứ 2 là con người, và cuối cùng là kỹ năng, mũi tên màu đỏ thể hiện mối quan hệ làm việc, mũi tên màu đen thể hiện mối quan hệ có kỹ năng. Dựa trên dữ liệu này công ty có thể thống kê được những người có cùng kỹ năng hoặc kỹ năng nào công ty chưa có, những người làm chung công ty. Ngoài ra, như đã đề cập bên trên các mối quan hệ trong dữ liệu đồ thị đều có thể gán thuộc tính, quan hệ màu đỏ có thể thêm thuộc tính làm việc từ khi nào, quan hệ màu đen có thể thêm thuộc tính trình độ của mỗi người là sơ cấp hay cao cấp. Những truy vấn trên dễ dàng thực hiện và có tốc độ xử lý nhanh đối với dữ liệu dạng đồ thị.

Ví dụ 2:

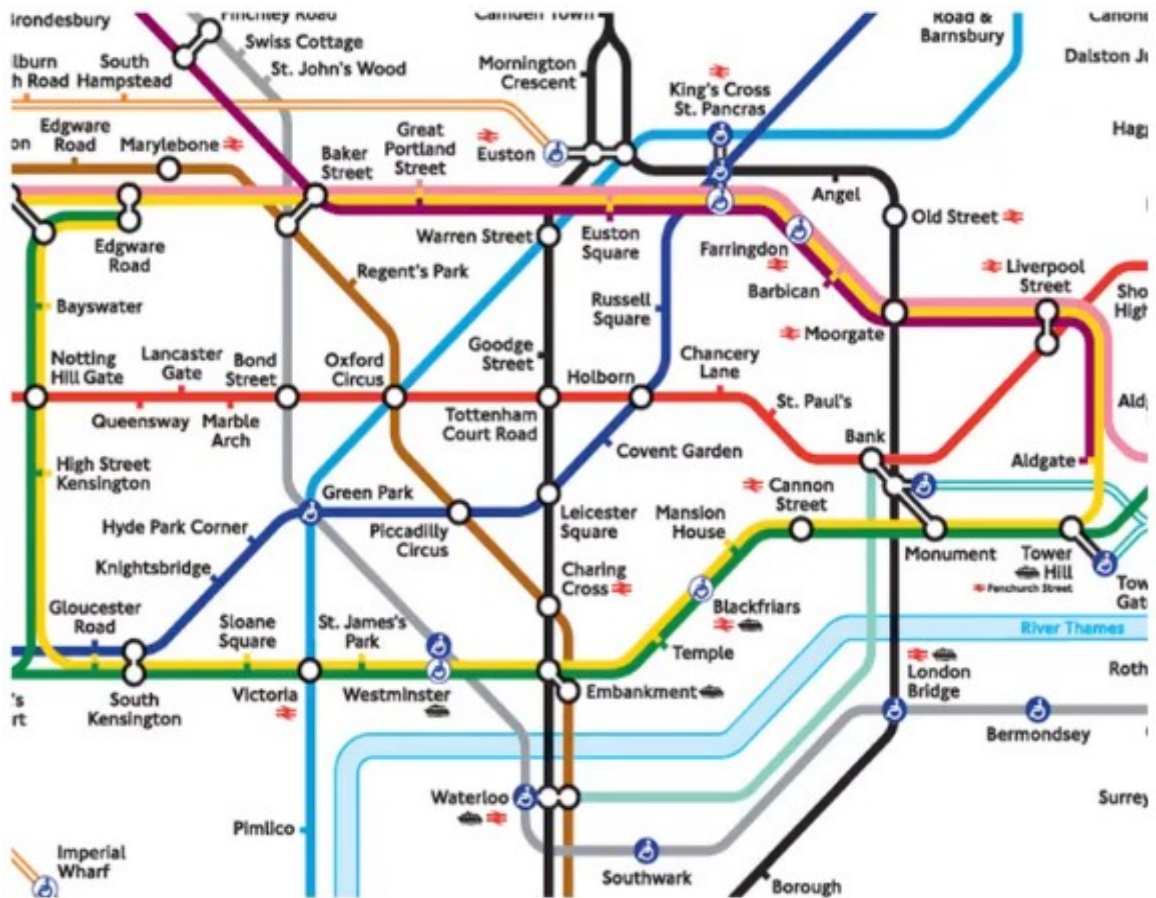


Hình 8. Dữ liệu đồ thị phân quyền

Ví dụ này là sơ đồ phân quyền dựa trên vai trò của mỗi người thường sử dụng trong lĩnh vực công nghệ thông tin. Ví dụ này có 3 lớp thực thể: đầu tiên là quyền, thứ 2 là nhóm hoặc vai trò, cuối cùng là người và một người có thể thuộc một hoặc nhiều nhóm, mũi tên màu đen thể hiện quan hệ được phép, mũi tên màu xanh thể hiện quan hệ thuộc về. Một câu truy vấn trong dữ liệu quan hệ sẽ tìm ra được toàn bộ quyền mà một người được phép thực hiện.

Dữ liệu quan hệ dễ dàng mở rộng mà không ảnh hưởng đến các node và quan hệ được tạo trước đó. Nếu như có người mới hoặc nhóm mới ta chỉ việc thêm vào dữ liệu mà không cần kiểm tra thuộc tính của các node trước đó.

Ví dụ 3:



Hình 9. Dữ liệu đồ thị thể hiện tuyến đường và các trạm trung chuyển trong thành phố

Đây là một ví dụ về cách tính toán lộ trình để chọn đường vận chuyển hiệu quả và tiết kiệm chi phí nhất. Bản đồ trên thể hiện tuyến đường và các trạm trung chuyển trong một thành phố. Mỗi trạm là một node, đoạn đường giữa các trạm thể hiện cho mỗi quan hệ giữa các node.

Đoạn đường hoặc mỗi quan hệ có thể chứa thuộc tính thời gian vận chuyển từ đó chúng ta có thể thực hiện truy vấn để tìm ra đoạn đường ngắn nhất.

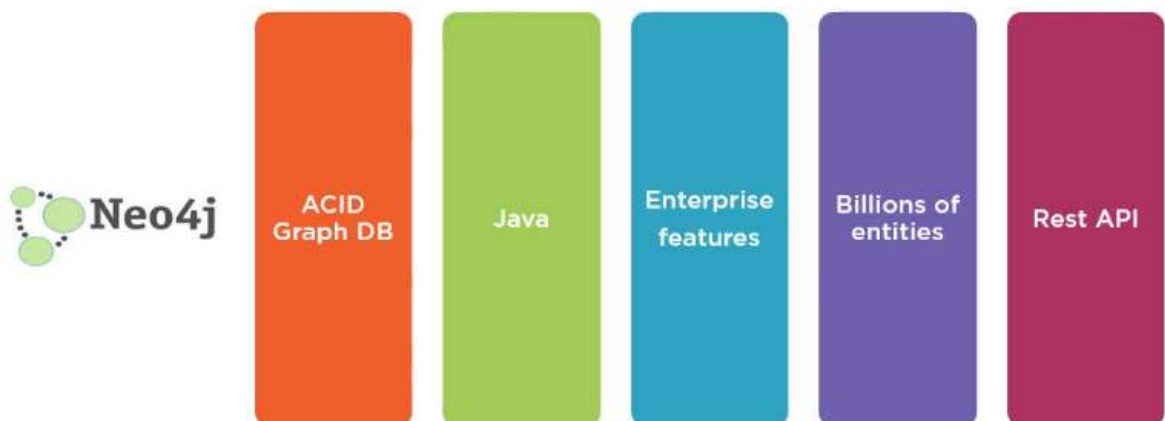
CHƯƠNG 2. TÌM HIỂU VỀ NEO4J

2.1 Giới thiệu về Neo4j

Cơ sở dữ liệu đồ thị như Neo4j chủ yếu dựa trên lý thuyết đồ thị, là một lý thuyết toán học. Đồ thị là cấu trúc bao gồm hai tham số chính: đỉnh và cạnh.

Các nhà phát triển ứng dụng có thể sử dụng các cấu trúc này để mô hình hóa các tình huống được xác định bởi các mối quan hệ. Ví dụ, một cơ sở dữ liệu đồ thị đơn giản cho phép các nhà phát triển lập mô hình mạng xã hội bao gồm người dùng là các nút và các mối quan hệ là kết nối giữa những người dùng. Một ví dụ khác có thể là một mạng lưới giao thông trong đó các thành phố, thị trấn hoặc làng mạc là các đỉnh và mặt khác, các con đường là các cạnh nối các đỉnh với trọng số cho biết khoảng cách.

Neo4j không phải là đồ thị dữ liệu duy nhất, mỗi nhà phát triển đều có công cụ riêng để thực hiện truy vấn đồ thị nhưng hiện tại Neo4j là một trong những lựa chọn phổ biến nhất cho dữ liệu mang tính liên kết cần những điểm mạnh của dữ liệu dạng đồ thị.



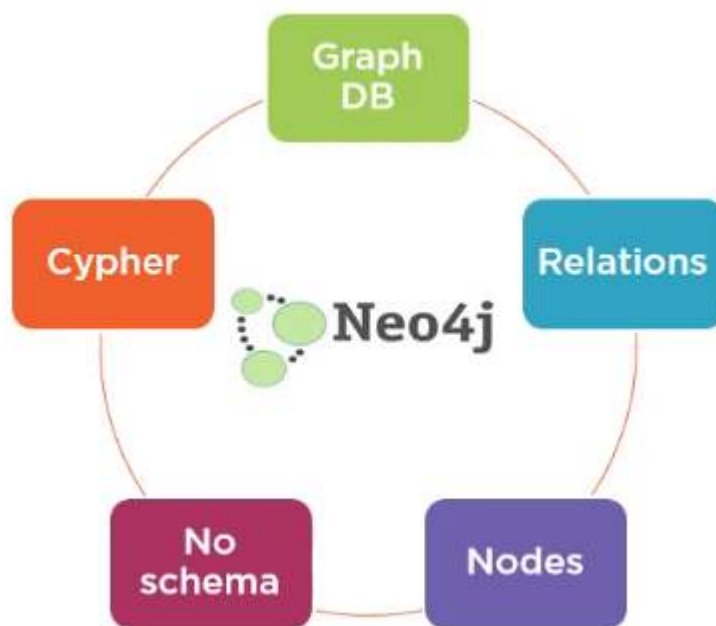
Tương tự như các dạng dữ liệu khác Neo4j đảm bảo được các tính chất về ACID. Mỗi thay đổi dữ liệu đều phải kết thúc trong một giao dịch nếu có giao dịch hoặc truy vấn nào xảy ra lỗi, Neo4j sẽ quay trở lại trạng thái dữ liệu chính xác trước đó.

Ví dụ khi có giao dịch giữa 2 tài khoản ngân hàng, số tiền sẽ đi từ tài khoản người chuyển đến tài khoản người nhận. Cần có 2 bước được thực hiện trong giao dịch này, đầu tiên trừ tiền từ tài khoản người gửi và cộng tiền vào tài khoản người nhận. 2 bước này phải đồng thời xảy ra hoặc cùng không xảy ra.

Neo4j được triển khai và phát triển bởi Java, nhưng nó hỗ trợ các ngôn ngữ lập trình khác, để các lập trình viên có thể sử dụng nền tảng mà họ yêu thích. Neo4j hoàn toàn tương thích với Windows, Linux hoặc MacOS.

Ngoài phiên bản tiêu chuẩn, Neo4j còn có phiên bản cao cấp hỗ trợ dữ liệu lớn, những máy chủ lớn hơn cho những truy vấn cần hiệu quả cao, sao lưu dữ liệu và theo dõi hệ thống. Neo4j hỗ trợ mở rộng và nâng cao phần cứng khi hệ thống xảy ra quá tải.

Neo4j hỗ trợ REST API nhằm tương thích với các nền tảng lập trình khác nhau.



Neo4j là loại dữ liệu No SQL

Neo4j hỗ trợ nhiều loại ngôn ngữ truy vấn, nhưng hiệu quả nhất và phổ biến nhất là Cypher. Cypher là ngôn ngữ được phát triển riêng cho Neo4j.

Community Edition

Base functionality

GPL v3 licensed

Enterprise Edition

Extra scale and availability features

Commercial license

Developer license

Evaluation license

Educational license

2.2 Các thành phần chính của cơ sở dữ liệu Neo4j

Mô hình Neo4j chủ yếu bao gồm các thành phần chính sau:

- Node: Các thực thể chính được kết nối với nhau bằng các mối quan hệ. Các node có thể có nhãn và thuộc tính.
- Các mối quan hệ: Mô tả các kết nối giữa các node và kết nối chúng với nhau. Các mối quan hệ có thể có một hoặc nhiều thuộc tính.
- Nhãn: Thể hiện vai trò của các node. Nhãn được sử dụng để nhóm các node. Mỗi node có thể có nhiều nhãn. Nhãn cũng được lập index để đẩy nhanh quá trình tìm kiếm các node trong biểu đồ.
- Thuộc tính: Thuộc tính của các node và các mối quan hệ liên quan đến các cặp tên hoặc giá trị.

Cơ sở dữ liệu Neo4j cho phép lưu trữ dữ liệu dưới dạng các cặp khóa giá trị, nghĩa là các thuộc tính có thể có bất kỳ loại giá trị nào chuỗi, số hoặc boolean. Cấu trúc dữ liệu biểu đồ ban đầu có vẻ hơi phức tạp, nhưng nó rất đơn giản và tự nhiên.

2.3 Lợi ích của cơ sở dữ liệu Neo4j

Mô hình Neo4j được thiết kế đặc biệt để xử lý lượng dữ liệu được kết nối đáng kể. Những mô hình này cung cấp cho người dùng một số lợi thế chính, bao gồm những điều sau:

- **Hiệu suất cao**

Đây là một trong những ưu điểm lớn nhất của mô hình dữ liệu đồ thị. Hiệu suất của cơ sở dữ liệu quan hệ giảm đi đáng kể khi số lượng và độ sâu của các mối quan hệ tăng lên. Mặt khác, hiệu suất của cơ sở dữ liệu đồ thị như cơ sở dữ liệu Neo4j vẫn hoạt động hiệu quả ngay cả khi lượng dữ liệu và mối quan hệ tăng lên.

Nhóm nghiên cứu đằng sau mô hình Neo4j cũng đã phát hành một thư viện cho phép các nhà phát triển và lập trình viên chạy các thuật toán đồ thị đồng thời trên dữ liệu có hàng tỷ node và mối quan hệ trong thời gian dài. Cơ sở dữ liệu Neo4j mở rộng theo chiều ngang. Điều đó có nghĩa là hiệu suất truy vấn không phụ thuộc vào kích thước của cơ sở dữ liệu. Neo4j có thể truyền tải dữ liệu lớn và cung cấp các tính năng thương mại như giao dịch ACID và sao lưu hoặc phục hồi tự động.

- **Linh hoạt**

Cấu trúc và lược đồ của mô hình đồ thị như Neo4j có thể dễ dàng thay đổi theo mục tiêu phát triển ứng dụng mà không phụ thuộc vào sự ràng buộc về khóa, làm cho Neo4j trở thành một cơ sở dữ liệu có tính linh hoạt cao. Neo4j có thể dễ dàng nâng cấp cấu trúc dữ liệu mà không làm ảnh hưởng đến chức năng đang sử dụng.

2.4 Cài đặt Neo4j

- Tải về file cài đặt Neo4j từ đường link sau:

<https://neo4j.com/download-thanks-desktop/?edition=desktop&flavour=winstall64&release=1.5.6&offline=true#installation-guide>

- Cài đặt theo hướng dẫn từ trang Neo4j
- Kích hoạt license Neo4j ở lần đầu tiên mở ứng dụng

Software registration

Neo4j Desktop is always free. Registration lets us know who has accepted this gift of graphs.

Register yourself with the following contact information.

Name *

Luan

Email *

luanpham91@gmail.com

Organization *

Organization

[Read about our privacy policy.](#)

Register later

Already registered? Add your software key here to activate this installation.

Software key *

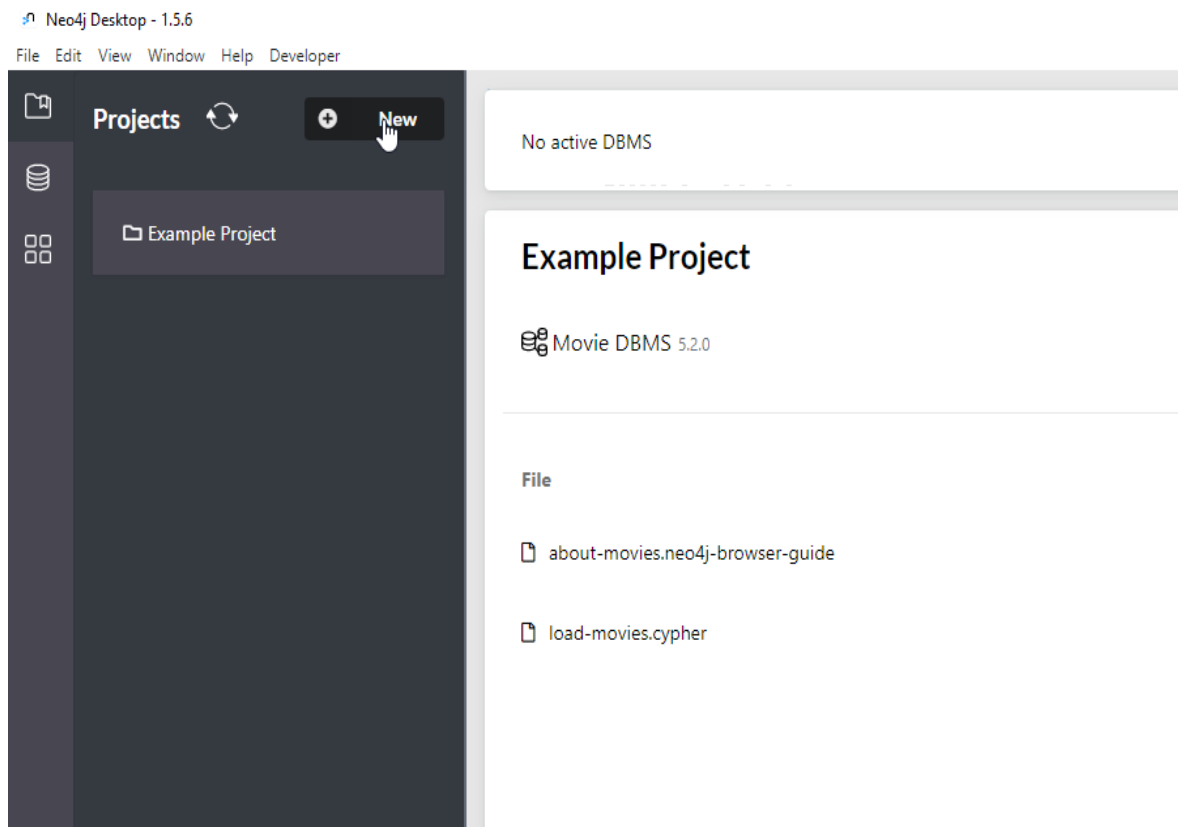
TK2J2YTZEJI0slcnLv5M8VQ2OpluvhDBXW
a1yjFsMir4yDql57B-
KZ2jf3TZAndw3_mkUWxaVSU4ySMMvtC_8
XK7HR-N8x5HM0Q92X9TtAUCFO2TMWM-
KawRb0MQObKdjgzhcyVWlyGEzy0AWRhYic
rZC21Sok9EGzo39MZss8Ertq5ee5xgK3P65x
ysF9ZVfUuBIWqLZ2zEGdLJp9Xwl17tC0_23--
umLtdSCICgQ8WvYficGKrol-
lsg70EBPNilJQRVuNfJlbifsw

OR

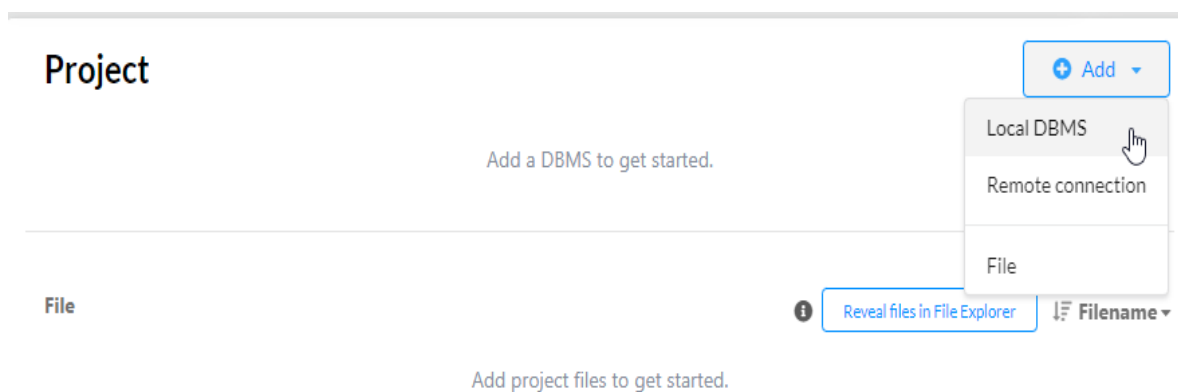
Register with Email

Key active được gửi kèm khi tải Neo4j.

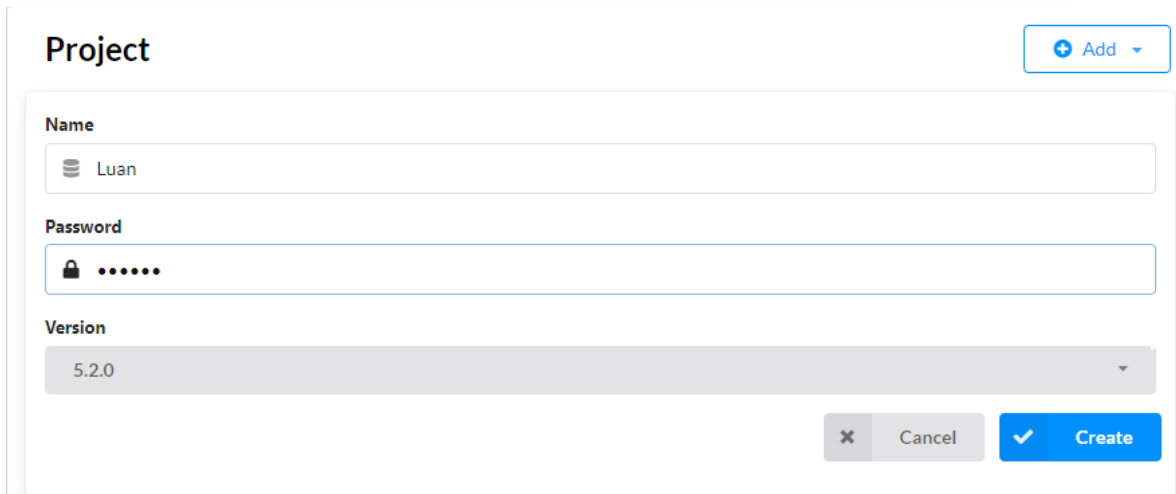
- Tạo project mới



- Add server cho project vừa tạo



- Đặt tên database và mật khẩu cho project vừa tạo



Project + Add ▾

Name

Luan

Password

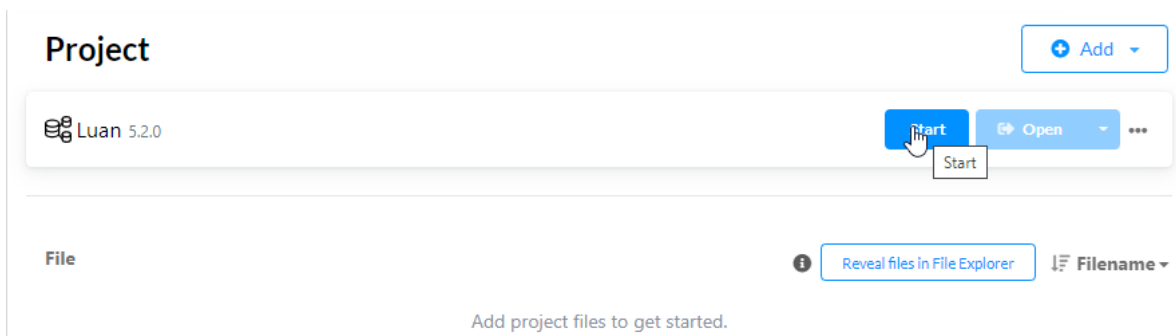
.....

Version

5.2.0 ▾

✕ Cancel ✓ Create

- Sau khi tạo xong, chọn start để chạy project vừa tạo



Project + Add ▾

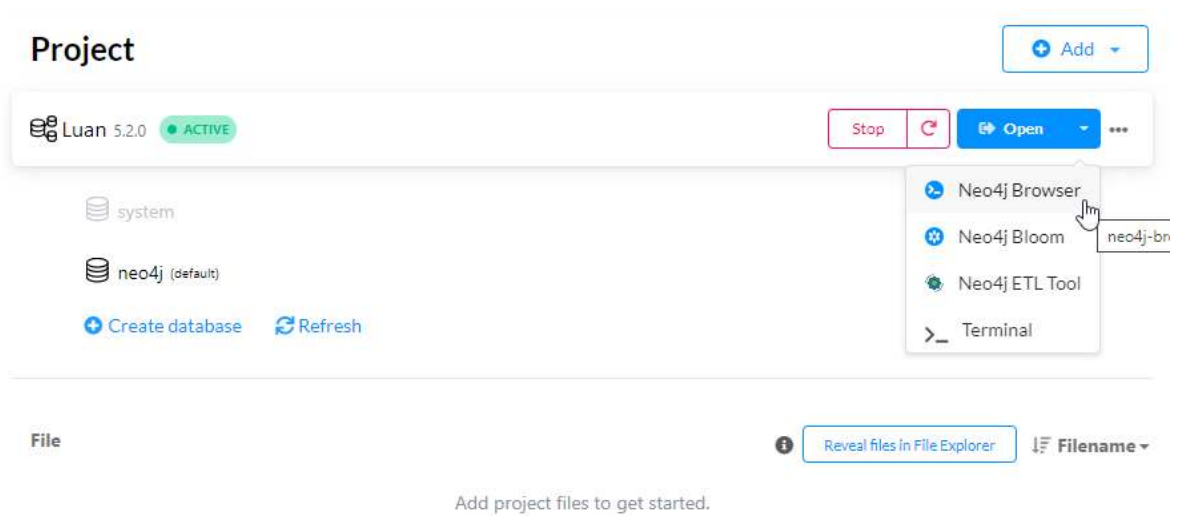
Luan 5.2.0 Start Open ▾ ⋮

Start

File Reveal files in File Explorer Filename ▾

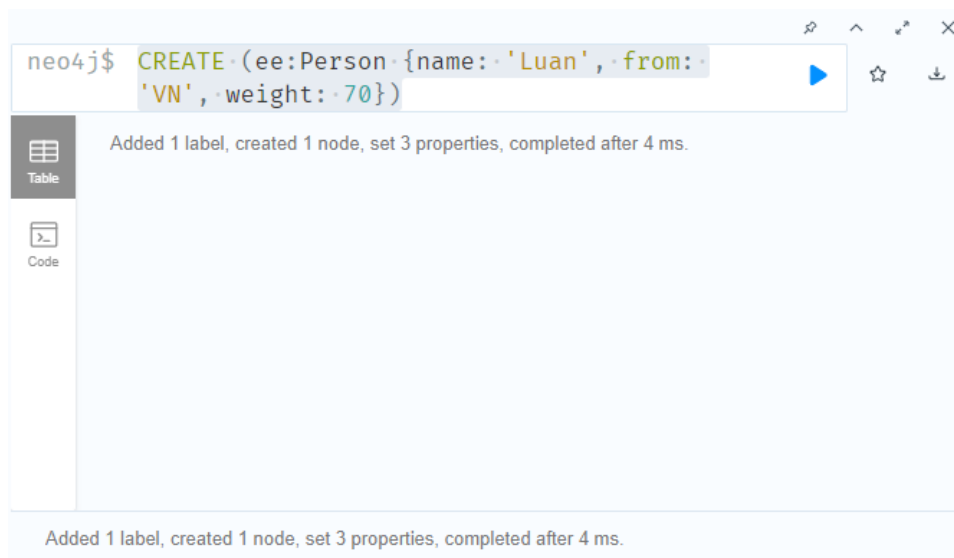
Add project files to get started.

- Kết nối đến cơ sở dữ liệu vừa tạo để nhập và truy vấn dữ liệu



- Tạo dữ liệu hoặc sử dụng dữ liệu mẫu của Neo4j
- Neo4j có sẵn 2 dữ liệu mẫu là movie và northwind, chúng ta có thể sử dụng lệnh sau để tạo một node.

CREATE (ee:Person {name: 'Luan', from: 'VN', weight: 70})



- Xem lại node vừa tạo

MATCH (ee:Person) **RETURN** ee;

MATCH (ee:Person) **WHERE** ee.name = 'Luan' **RETURN** ee;



CHƯƠNG 3. NGÔN NGỮ TRUY VẤN CYPHER

3.1 Giới thiệu

Cypher là ngôn ngữ được sử dụng phổ biến và hiệu quả nhất trong cơ sở dữ liệu Neo4j, Cypher vẫn đang được thay đổi liên tục bởi các nhà phát triển.

Cypher truy vấn theo mong muốn, cách viết của người truy vấn. Cypher gần gũi với ngôn ngữ của con người, dễ hiểu khi xem tìm hiểu các câu truy vấn vì Cypher truy vấn dựa trên mẫu dữ liệu để lấy dữ liệu cần thiết.

Cypher hỗ trợ truy vấn theo kiểu mệnh đề tương tự như dữ liệu quan hệ. Ví dụ như mệnh đề WHERE và ORDER BY.

Cách sử dụng Cypher là chúng ta tưởng tượng ra một hình ảnh về dữ liệu chúng muốn tìm, sau đó đồ thị hóa lên một trang giấy hoặc một bảng vẽ, sau đó đưa thông tin vào câu truy vấn. Một ví dụ đơn giản chúng ta muốn lấy dữ liệu như đồ thị sau.

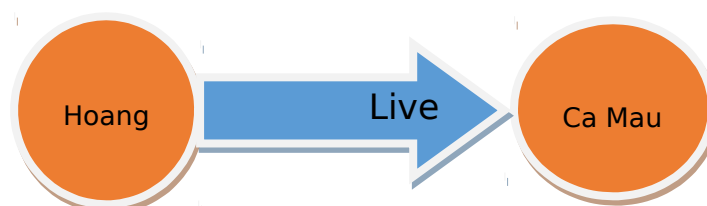


Một node có mối quan hệ Played với một node khác. Chúng ta cần chuyển ngôn ngữ từ lời nói sang ASCII như sau.

`()-[:PLAYED]->()`

Chúng ta có thể thấy câu truy vấn này rất dễ hiểu () tượng trưng cho 2 node, -> thể hiện chiều của mối quan hệ, [] thể hiện tên mối quan hệ.

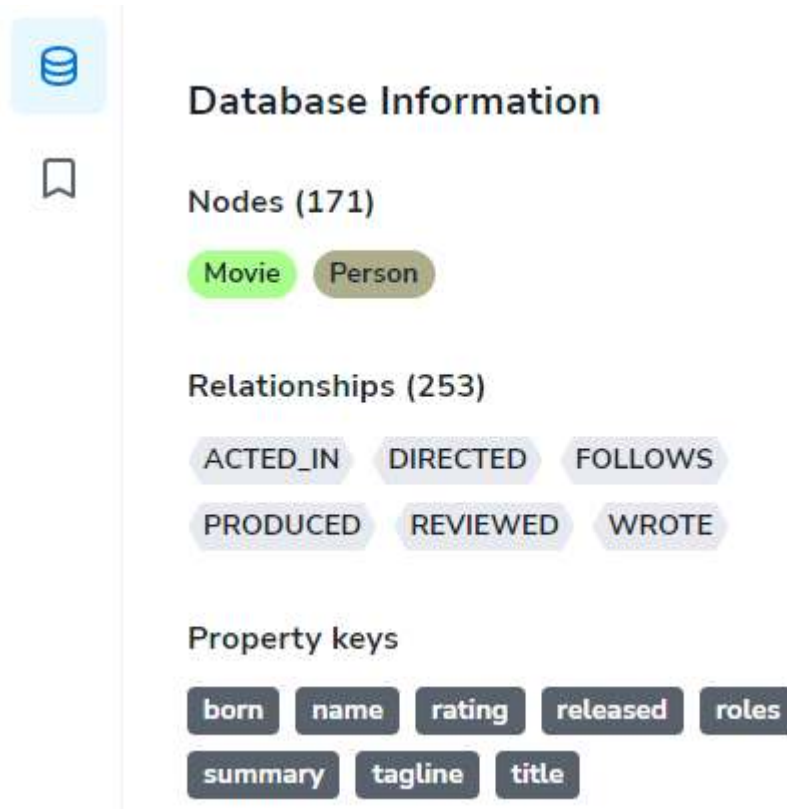
Một ví dụ để có thể hiểu rõ hơn



`(:Hoang)-[:Live in]->(:Ca Mau)`

3.2 Mô hình dữ liệu đồ thị

Khi xem thông tin một cơ sở dữ liệu trong Neo4j ta sẽ thấy được mô hình dữ liệu gồm những gì: số lượng node, số lượng quan hệ, thuộc tính...



Để xem thông tin các node, chúng ta có thể chọn vào node bất kỳ. Ví dụ như node Movie như hình bên dưới. Khi đó sẽ hiện ra những node đang có nhãn là Movie. Dữ liệu sẽ có nhiều hơn nhưng giới hạn ở mức 25 sẽ cho chúng ta thấy về mô hình cơ bản của một dữ liệu đồ thị trên Neo4j

Database Information

Nodes (171)

Movie Person

Relationships (253)

ACTED_IN DIRECTED FOLLOWS
PRODUCED REVIEWED WROTE

Property keys

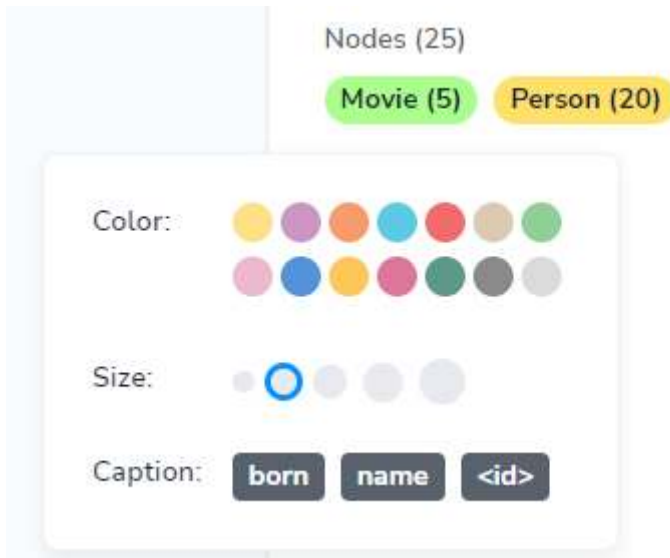
born name rating released roles
summary tagline title



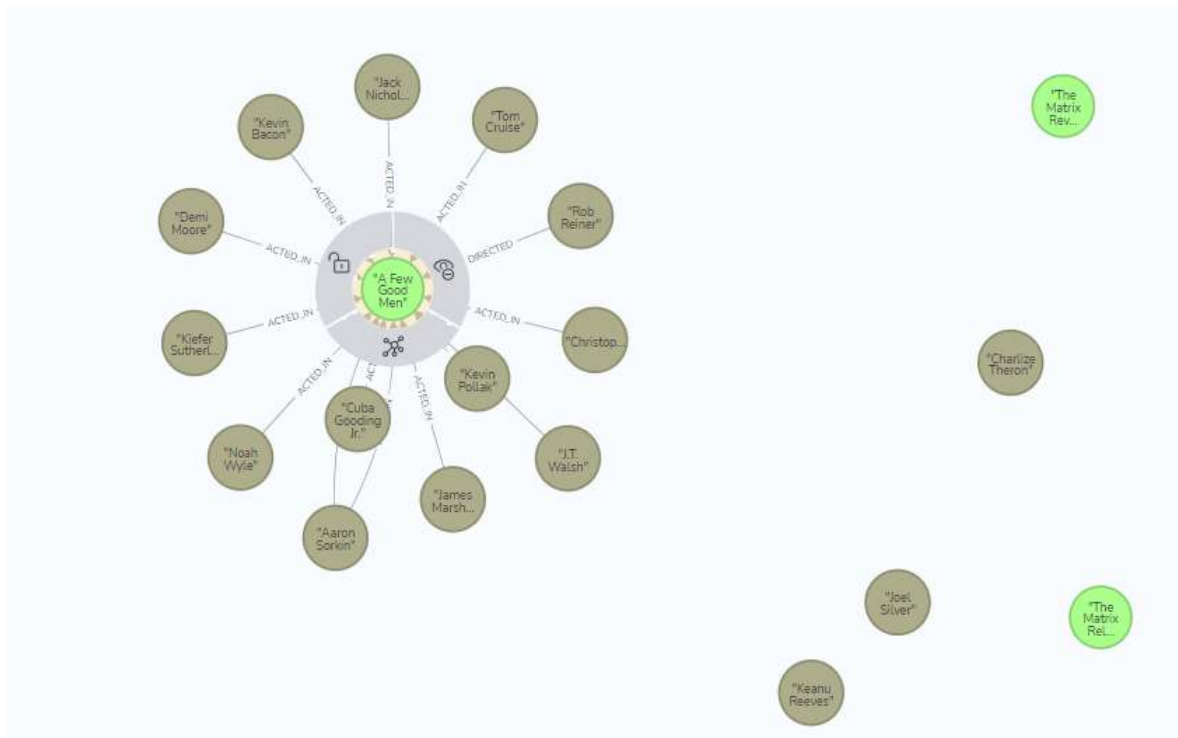
Ngoài ra chúng ta có thể xem được thông tin số lượng node nào đang dán nhãn gì được hiển thị. Như hình bên dưới lấy ngẫu nhiên 25 node trong đó có 5 node có nhãn Movie và 20 node có nhãn Person. Các node cũng được phân biệt bằng màu sắc khác nhau theo nhãn



Trên mỗi nhãn chúng ta có thể thay đổi màu sắc, kích thước và xem các thuộc tính của node có nhãn đó.



Để xem thông tin của một node chúng ta có thể double click để xem các mối quan hệ và các node liên kết đến node này.



3.3 Cú pháp cơ bản

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

Định dạng nút (nodes)

- `()`: nút rỗng
- `(varname:NodeName)`: Node có nhãn là `NodeName`, tên biến của node là `varname`. Nút có thể không có tên biến.
- Truy vấn clean: Gợi ý sử dụng cách đặt tên nhãn theo kiểu viết hoa chữ cái đầu.
- Nên đặt tên nhãn 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 quan hệ (relationship)

- `[varname:RELATIONSHIP_NAME]`: mỗi quan hệ có nhãn là `RELATIONSHIP_NAME` và biến quan hệ là `varname`.

- Truy vấn clean: Gợi ý cách đặt tên nhân theo kiểu `upper_case`, tất cả được viết hoa và sử dụng dấu gạch nối giữa các từ. Nhân của các quan hệ nên là động từ. Tên nhân có phân biệt chữ hoa chữ thường.

Khóa thuộc tính, biến, tham số, bí danh và hàm:

- Ví dụ: `title`, `size()`, `count()`, `firstName...`
- Có phân biệt chữ hoa chữ thường.
- Truy vấn sạch: nên viết theo định dạng `camelCase` (chữ cái đầu viết thường).

Mệnh đề:

- Ví dụ: `MATCH`, `WHERE`, `WITH`, `UNWIND...`
- Các mệnh đề không phân biệt chữ hoa chữ thường.
- Truy vấn sạch: các mệnh đề nên được tạo kiểu là tất cả các chữ in hoa, được đặt ở đầu mỗi dòng để dễ đọc và dễ truy vấn.

Keyword:

- Ví dụ: `AND`, `OR`, `IN`, `NOT`, `DISTINCT`, `STARTS WITH`, `CONTAINS`, `ENDS WITH...`
- Các keyword không phân biệt chữ hoa chữ thường.
- Truy vấn sạch: Nên được viết in hoa, không cần đặt ở đầu dòng mới.

Dấu chấm phẩy

- Dùng trong trường hợp có 1 tập các câu lệnh Cypher và cần phân tách giữa các lệnh. Không nên dùng khi chỉ có 1 lệnh.

Metacharacter:

- Dấu nháy đơn: nên dùng cho các giá trị chuỗi bằng chữ.

- Ví dụ: 'Mats\' quote: 'statement', 'Cypher's a nice language'

3.4 Các lệnh Cypher cơ bản

2 lệnh quan trọng nhất được sử dụng trong Cypher là MATCH và RETURN.

Ví dụ 1: Cho label Person có 1 node với property {name:'Jennifer'}

Cypher command:

MATCH (p: Person {name: 'Jennifer'}) RETURN p



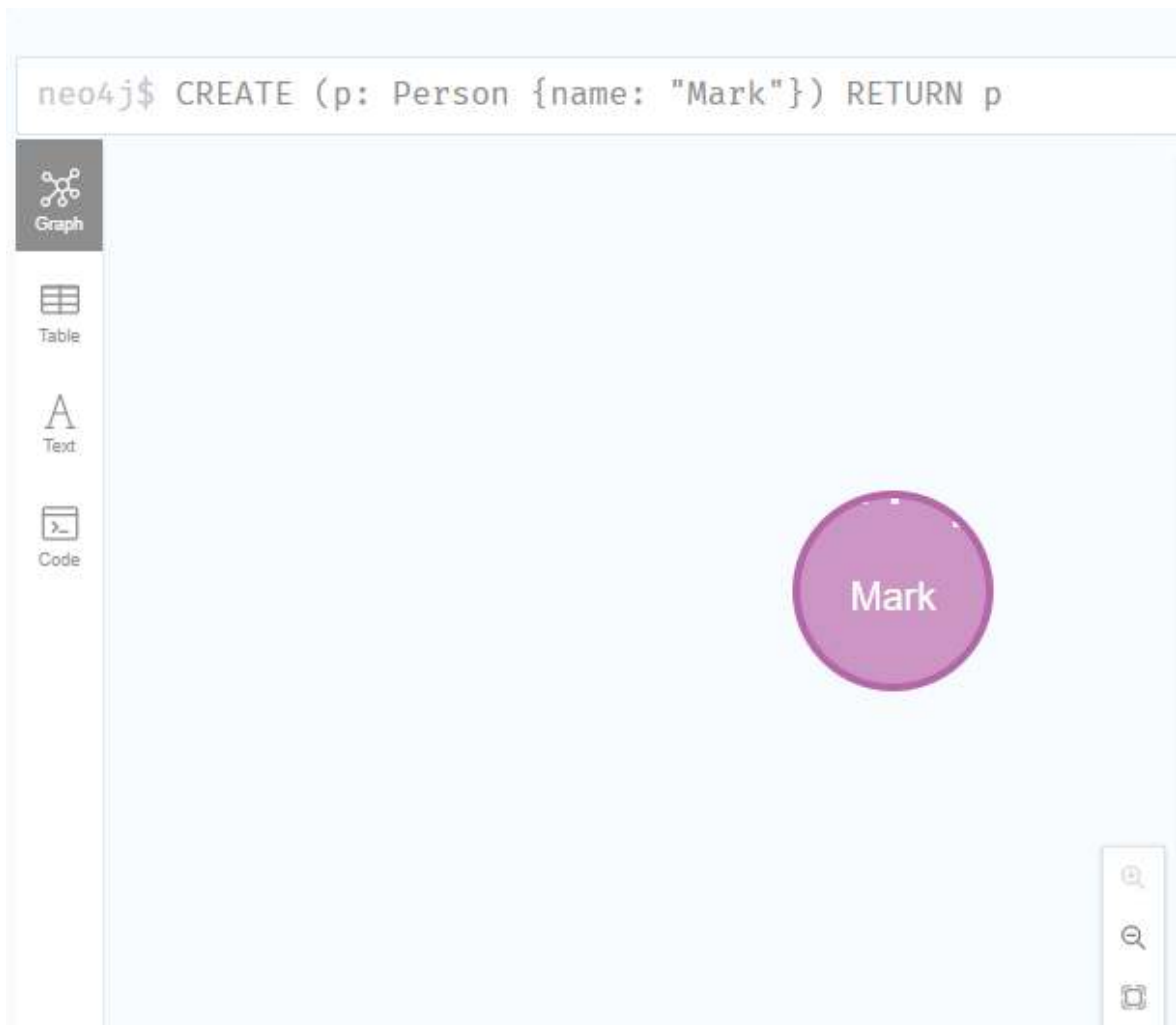
Để tạo mới một đối tượng trong Cypher ta dùng lệnh Create tương tự với Insert trong dữ liệu quan hệ.

Ví dụ 2: Tạo thêm một node có nhãn(label) Person với property {name: 'Mark'}

Cypher command:

```
CREATE (p: Person {name: 'Mark'}) RETURN p
```

Lệnh return không bắt buộc, chỉ dùng để kiểm tra node vừa tạo có đúng hay không.



Để thiết lập mối quan hệ cho nó, ta phải sử dụng MATCH để tránh lặp lại nút đã tạo.

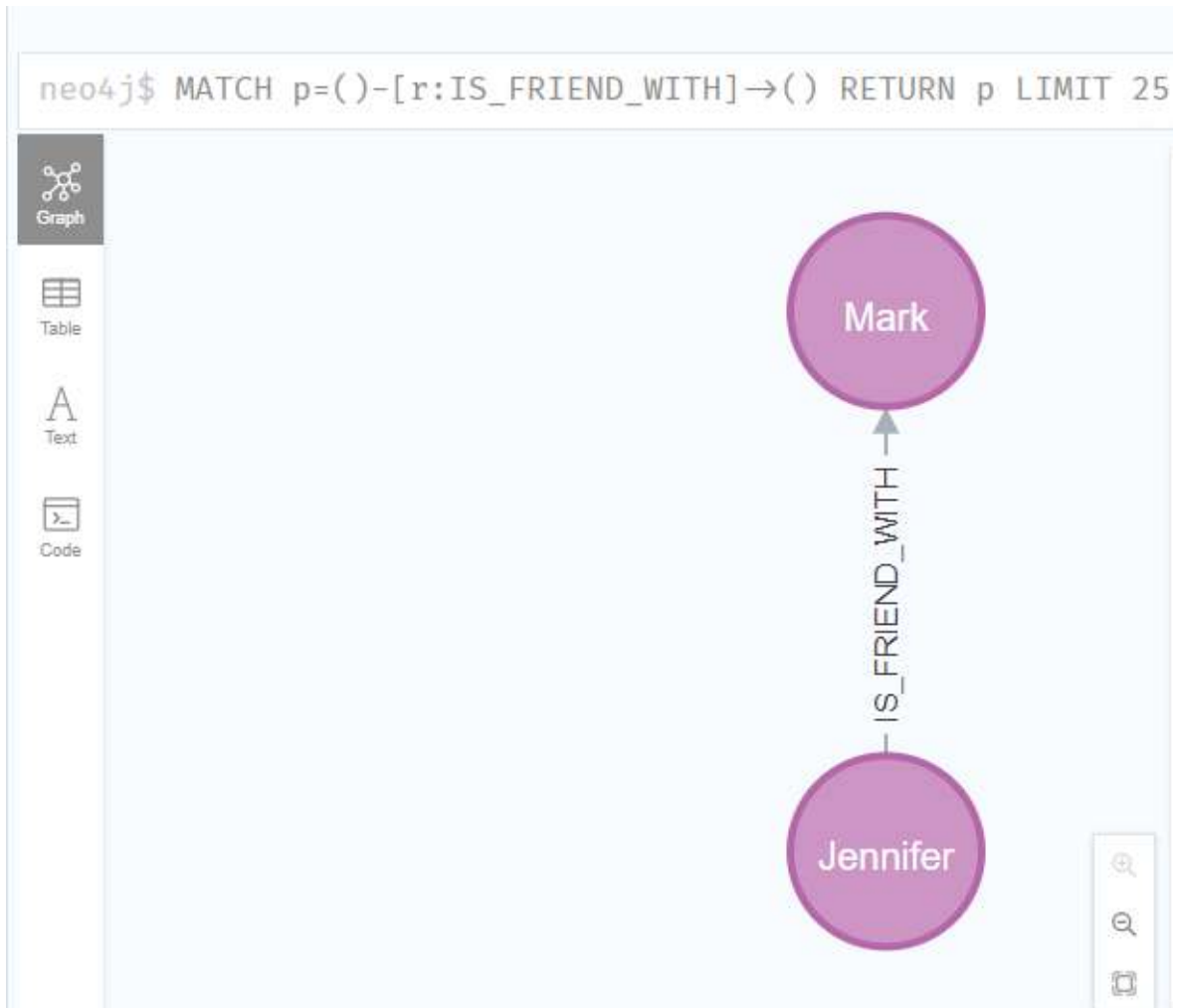
Cypher command:

```
MATCH (jennifer: Person {name: 'Jennifer'})
```

```
MATCH (mark: Person {name: 'Mark'})
```

```
CREATE (jennifer)-[rel:IS_FRIEND_WITH]->(mark)
```

Nếu không có 2 lệnh MATCH như trên, Cypher sẽ tự động tạo các node mới mà không kiểm tra xem nó đã tồn tại trong csdl hay chưa.



Để thêm, sửa, xóa các thuộc tính của một node ta sử dụng lệnh MATCH ... SET lệnh này sẽ tìm ra node cần sửa sau đó gán lại thuộc tính cần thay đổi.

Ví dụ 3: Thêm thuộc tính birthday cho Person Jennifer

Cypher command:

```
MATCH (p: Person {name: 'Jennifer'})
```

```
SET p.birthday = date('1999-01-01')
```



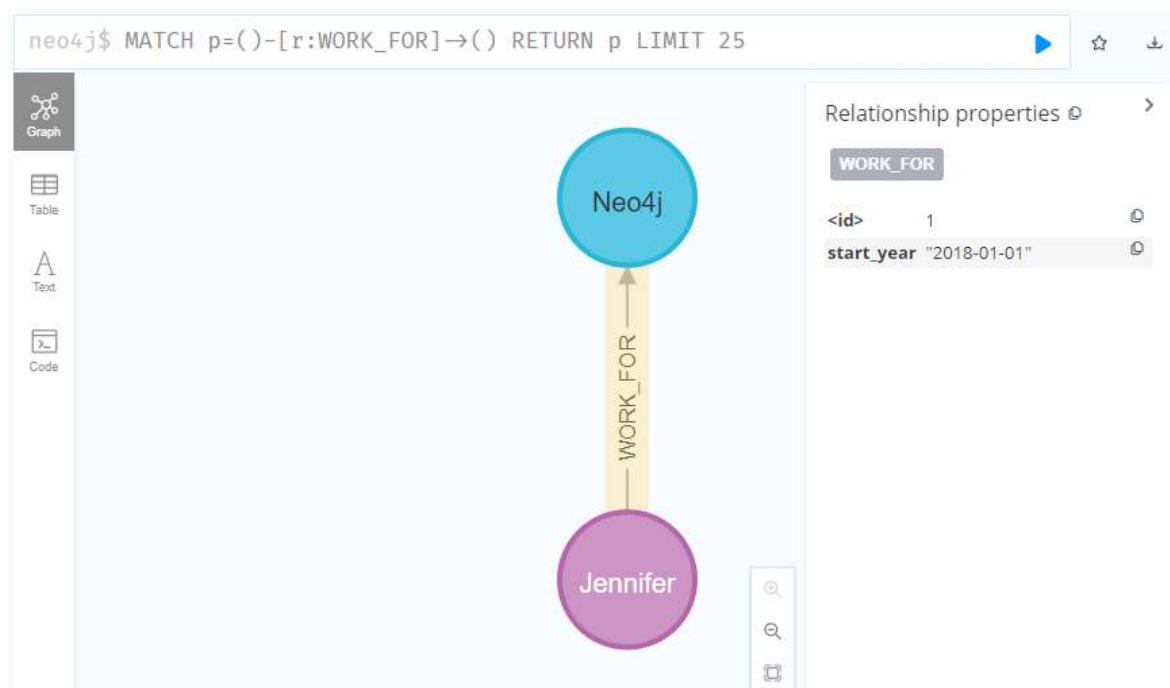
Ví dụ 4: Jennifer làm việc ở công ty 'Neo4j' từ năm 2018

Cypher command:

```
MATCH (:Person {name: 'Jennifer'})
```

```
-[rel:WORK_FOR]->(:Company {name: 'Neo4j'})
```

```
SET rel.start_year = date({year:2018})
```



Để xóa node hoặc quan hệ trong Cypher ta dùng lệnh `MATCH ... DELETE`. Tương tự như sửa thuộc tính, lệnh `MATCH` được dùng để tìm đến dữ liệu cùng xóa.

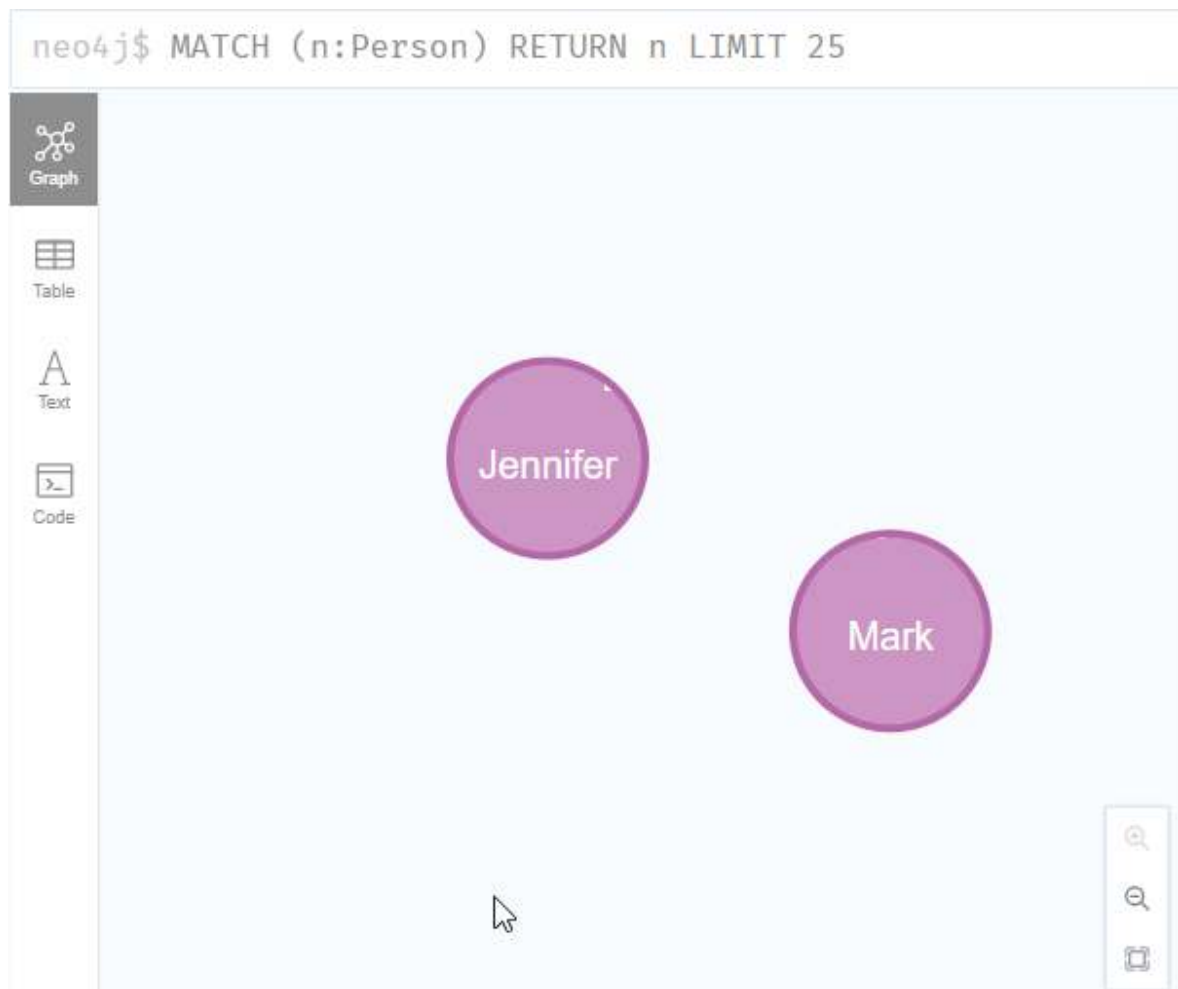
Ví dụ 5: xóa mối quan hệ bạn bè giữa Jennifer và Mark

Cypher command:

```
MATCH (j: Person {name: 'Jennifer'})
```

```
-[friend: IS_FRIEND_WITH]->(m: Person {name: 'Mark'})
```

```
DELETE friend
```



Ví dụ 6: xóa nút Person có property {name: 'Mark'}

Cypher command:

```
MATCH (p: Person {name: 'Mark'}) DELETE p
```



Ví dụ 7: Xóa đồng thời nút và mối quan hệ của nó

Cypher command:

```
MATCH (p: Person {name: 'Mark'}) DETACH DELETE p
```



Để xóa thuộc tính của node trong Cypher ta dùng lệnh `MATCH ... REMOVE`.

`REMOVE`: xóa hoàn toàn thuộc tính khỏi nút và không lưu trữ nó nữa.

Ví dụ 8: Xóa thuộc tính birthday của Jennifer

```
MATCH (p: Person {name: 'Jennifer'})
```

```
REMOVE p.birthday
```



```
1 MATCH (p: Person {name: 'Jennifer'})
2 REMOVE p.birthday
3
```

Table

Set 1 property, completed after 2 ms.

Node properties

Person

<id> 1
name Jennifer

Node Jennifer không còn thuộc tính birthday

SET: Sử dụng từ khóa SET để đặt giá trị thuộc tính thành null (trong mô hình cơ sở dữ liệu Neo4j không lưu trữ giá trị null).

Ví dụ 9: Đặt giá trị của thuộc tính birthday thành null

MATCH (p: Person {name: 'Jennifer'})

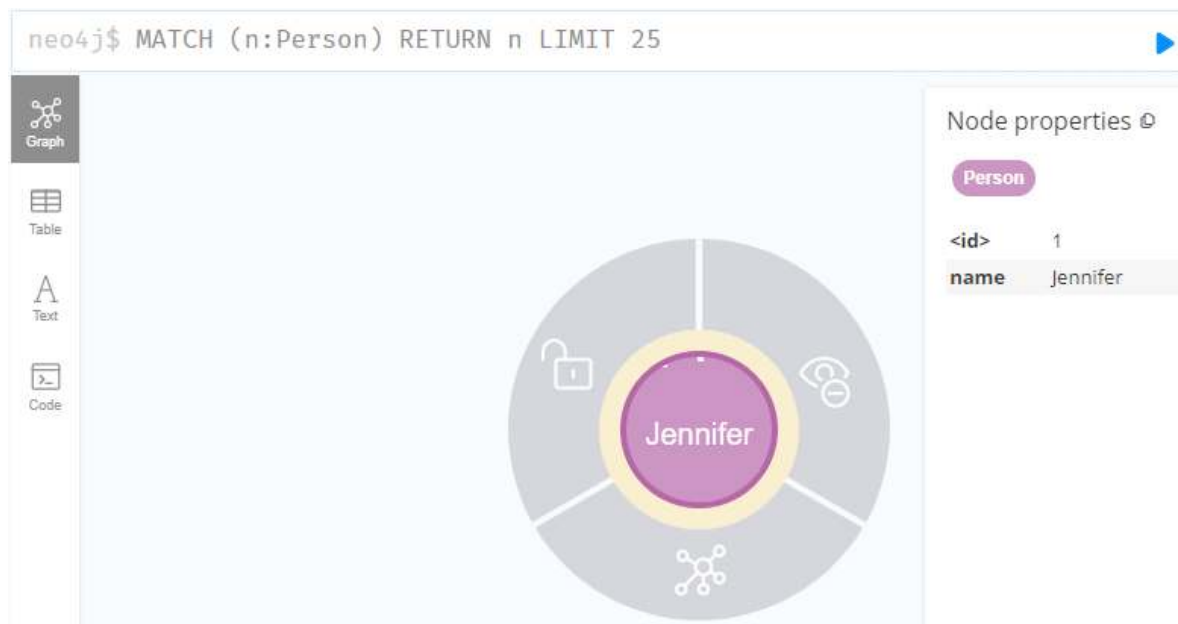
SET p.birthday = null

```
1 MATCH (p: Person {name: 'Jennifer'})
2 SET p.birthday = null
3
```

Table

Set 1 property, completed after 1 ms.

Code



Do Neo4j không lưu giá trị null, nên thuộc tính là null sẽ không được lưu trữ hay hiển thị khi truy vấn.

Merge thực hiện chọn lọc và kiểm tra dữ liệu có tồn tại trong csdl hay không, trước khi chèn vào cơ sở dữ liệu.

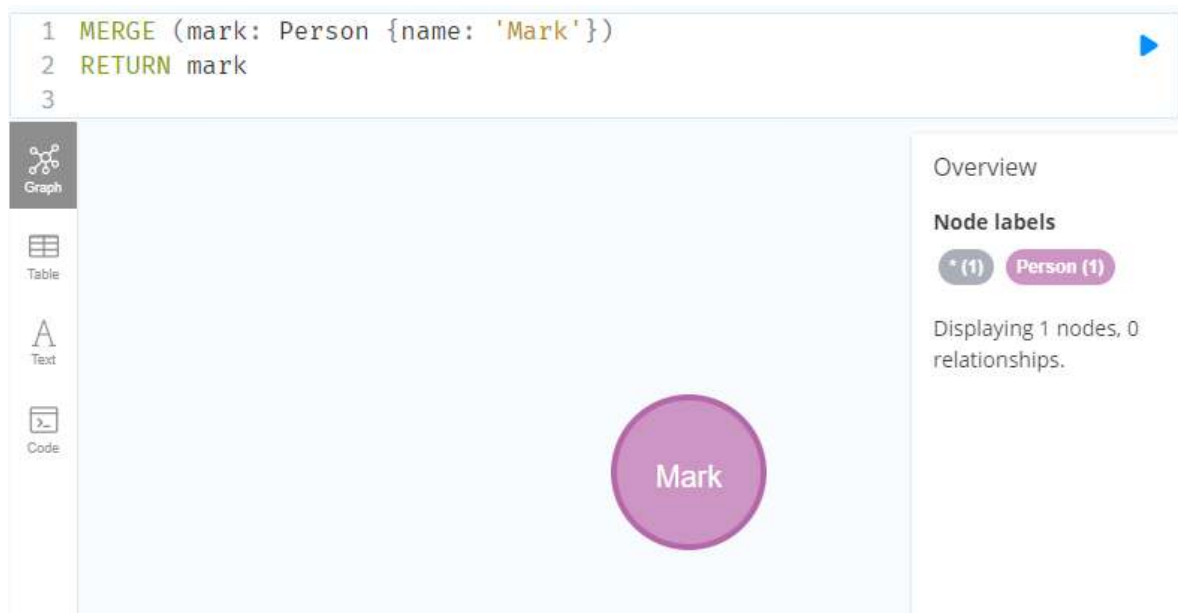
Ví dụ 10: Chèn Person Mark vào csdl bằng Merge

Cypher command:

```
MERGE (mark: Person {name: 'Mark'})
```

```
RETURN mark
```

Nút mark đã tồn tại trong cơ sở dữ liệu trước đó, nên câu lệnh trên sẽ không tạo thêm nút mark mới mà chỉ trả về nút mark đã có.



Merge trên 1 mối quan hệ: Sử dụng tương tự như Create. Nếu mối quan hệ chưa được thiết lập, merge sẽ thực hiện tạo mới toàn bộ (mặc dù nút đã tồn tại).

Ví dụ 11: Thực hiện tạo mối quan hệ bạn bè giữa Jennifer và Mark

Cypher command:

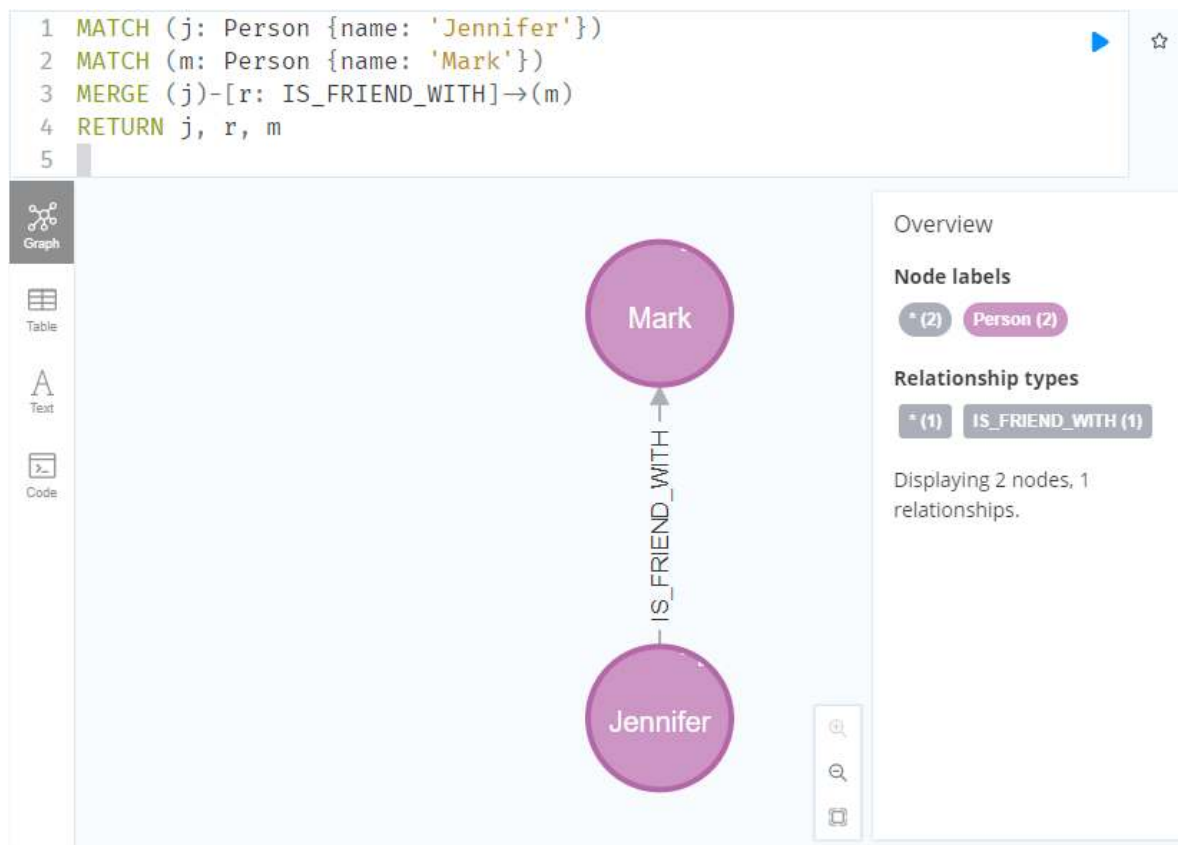
```
MATCH (j: Person {name: 'Jennifer'})
```

```
MATCH (m: Person {name: 'Mark'})
```

```
MERGE (j)-[r: IS_FRIEND_WITH]->(m)
```

```
RETURN j, r, m
```

Sử dụng MATCH để thực hiện khớp dữ liệu trước khi tạo mối quan hệ. Mối quan hệ này đã được tạo trước đó nên Merge chỉ cần trả về dữ liệu đã tồn tại.



Lưu ý: Nếu chỉ sử dụng MERGE mà không khớp dữ liệu sẽ dẫn đến việc lệnh MERGE tạo lại các nút đã tạo nếu không tìm thấy mối quan hệ giữa các nút đó dẫn đến bị lặp dữ liệu.

Nếu muốn sử dụng MERGE để đảm bảo không tạo ra các bản sao, đồng thời khởi tạo một số thuộc tính mới hoặc cập nhật lại các thuộc tính khác nếu nó chỉ được khớp, trong trường hợp này, ta sử dụng ON CREATE hoặc ON MATCH với SET.

Cypher command:

```
MERGE (m: Person {name: 'Mark'})
-[r:IS_FRIEND_WITH]-(j: Person {name: 'Jennifer'})
ON CREATE SET r.since = date('2018-01-01')
ON MATCH SET r.update = date()
```

```

1 MERGE (m: Person {name: 'Mark'})
2 -[r:IS_FRIEND_WITH]-(j: Person {name: 'Jennifer'})
3 ON CREATE SET r.since = date('2018-01-01')
4 ON MATCH SET r.update = date()
5

```



Set 1 property, completed after 5 ms.

Relationship properties >

IS_FRIEND_WITH

<id>	0	
update	"2022-12-31"	

Sử dụng mệnh đề WHERE trong Cypher, tương tự với mệnh đề WHERE trong SQL mệnh đề WHERE trong Cypher cũng hỗ trợ cho việc lọc dữ liệu cần truy vấn.

So sánh cú pháp sử dụng truy vấn trước với cú pháp sử dụng với WHERE:

Cypher command before:

```
MATCH (person: Person {name: 'Jennifer'})
```

```
RETURN person
```

Cypher command with WHERE:

```
MATCH (person: Person)
```

```
WHERE person.name = 'Jennifer'
```

```
RETURN person
```

Cả hai truy vấn đều trả về cùng 1 kết quả. WHERE có thể làm nhiều hơn thế, WHERE có thể sử dụng kết hợp cùng các toán tử khác để tối ưu truy vấn.

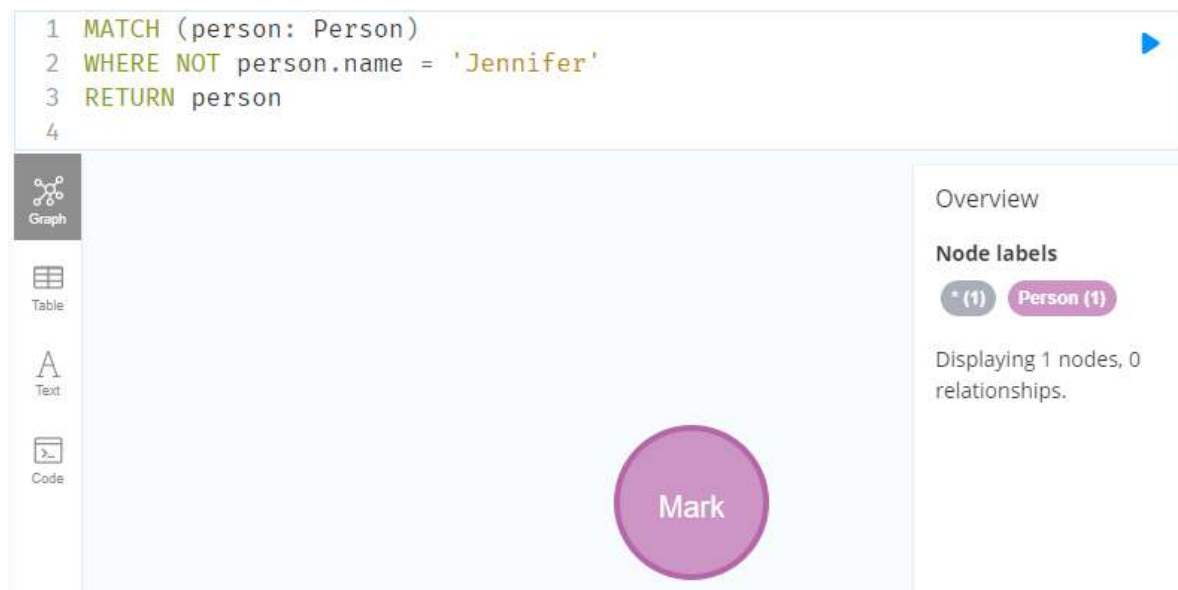
WHERE NOT: trả về thuộc tính không khớp với mẫu.

Cypher command:

```
MATCH (person: Person)
```

```
WHERE NOT person.name = 'Jennifer'
```

```
RETURN person
```



Ngoài ra, WHERE còn có thể đi cùng với AND, OR, XOR.

Truy vấn trong một phạm vi nhất định với WHERE:

```
MATCH (person: Person)
```

```
WHERE 1999 <= person.yearBirthday <= 2000
```

```
RETURN person
```

WHERE exists(property): kiểm tra xem thuộc tính có tồn tại trong nút hoặc một mối quan hệ có tồn tại trong mẫu.

Cypher command:

```
MATCH (varname: NodeLabel)
```

```
WHERE exists(varname.property)
```

RETURN varname

WHERE ... IN: Kiểm tra xem giá trị thuộc tính có phải là giá trị trong danh sách đã cho.

Cypher command:

MATCH (varname: NodeLabel)

WHERE varname.property IN your_array

RETURN varname

STARTS WITH: tìm chuỗi bắt đầu bằng chuỗi bạn chỉ định.

MATCH (varname: NodeLabel)

WHERE varname.property STARTS WITH your_string

RETURN varname.property

CONTAINS: kiểm tra xem chuỗi được chỉ định có phải 1 phần của giá trị thuộc tính không.

MATCH (varname: NodeLabel)

WHERE varname.property CONTAINS your_string

RETURN varname.property

ENDS WITH: tìm chuỗi có kết thúc bằng chuỗi bạn chỉ định.

MATCH (varname: NodeLabel)

WHERE varname.property ENDS WITH your_string

RETURN varname.property

Regular expressions: kiểm tra một phần giá trị của chuỗi bằng biểu thức chính quy.

Ví dụ: Tìm tất cả các nút Person có bắt đầu bằng 'L'.

Cypher command:

MATCH (p: Person)

WHERE p.name =~ 'L.*'

RETURN p.name

```
1 MATCH (p: Person)
2 WHERE p.name =~ 'L.*'
3 RETURN p.name
4
```

	p.name
1	"Luan"

Trả về kết quả kể cả khi chúng không khớp với toàn bộ mẫu hoặc tất cả các tiêu chí. Nó tương tự như outer join trong SQL.

OPTIONAL MATCH: nếu không tìm thấy kết quả, hàng được khớp sẽ trả về null.

Ví dụ: Tìm những người có tên bắt đầu bằng 'J' và làm việc trong công ty.

MATCH (p: Person) WHERE p.name STARTS WITH 'L'

OPTIONAL MATCH (p)-[:WORK_FOR]->(c:Company)

RETURN p.name, c.name

```
1 MATCH (p: Person) WHERE p.name STARTS WITH 'L'
2 OPTIONAL MATCH (p)-[:WORK_FOR]->(c:Company)
3 RETURN p.name, c.name
4
```

	p.name	c.name
1	"Luan"	"Katalon"

Lệnh Cypher trên sẽ trả về tất cả những người có tên bắt đầu bằng 'L', và tên công ty mà họ đang làm việc. Những người không làm việc ở công ty, tên công ty mà họ làm việc sẽ trả về giá trị null.

CHƯƠNG 4. LĨNH VỰC ỨNG DỤNG

4.1 Ngăn chặn gian lận

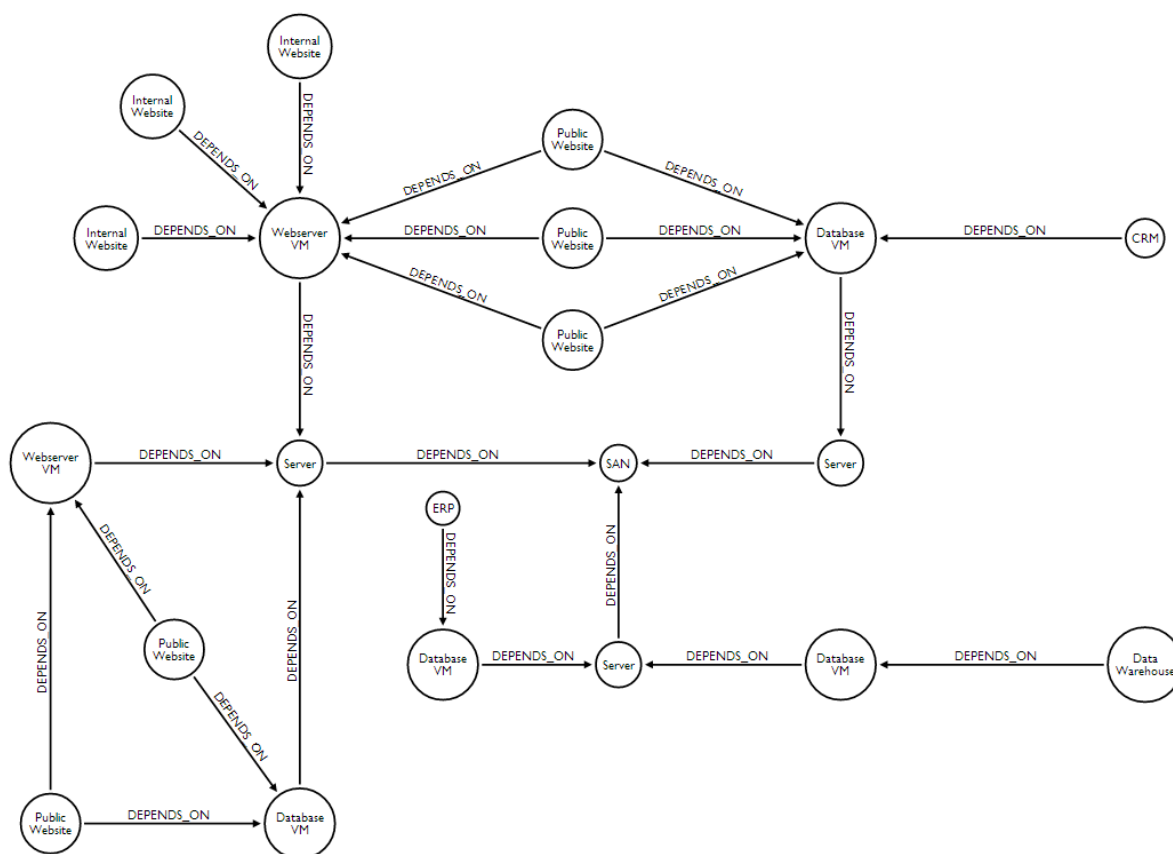
Các biện pháp phòng chống gian lận truyền thống tập trung sâu vào các điểm dữ liệu riêng biệt, bao gồm tài khoản, cá nhân, thiết bị hoặc địa chỉ IP. Vấn đề ở đây là tội phạm hiện đại ngày nay có thể thoát khỏi các phương pháp phát hiện này chỉ đơn giản bằng cách tạo ra các vòng lừa đảo với danh tính không có thật. Để ngăn chặn việc thoát như vậy, cần phải xem xét các kết nối liên kết các điểm dữ liệu riêng lẻ.



Mặc dù không có biện pháp phòng chống gian lận nào là hoàn hảo, nhưng bạn có thể nâng cao quy trình bằng cách phân tích các kết nối giữa các dữ liệu riêng lẻ. Đây là lúc mà mô hình Neo4j có ích để phát hiện các mẫu khó mà cơ sở dữ liệu quan hệ khó có thể phát hiện ra.

Các tổ chức doanh nghiệp sử dụng cơ sở dữ liệu Neo4j để tăng cường khả năng phát hiện gian lận của họ nhằm ngăn chặn các hành vi gian lận tài chính khác nhau, bao gồm gian lận ngân hàng bên thứ nhất, gian lận thương mại điện tử, gian lận thẻ tín dụng, gian lận bảo hiểm và gian lận rửa tiền ngay lập tức.

4.2 Mạng và hoạt động công nghệ thông tin

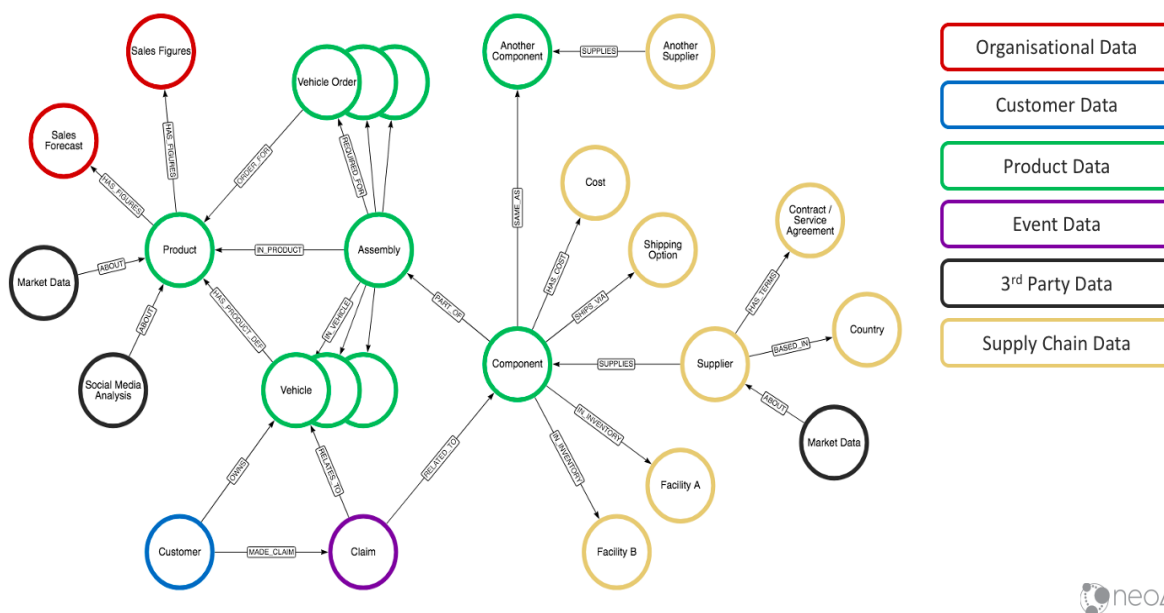


Cơ sở hạ tầng mạng và CNTT cực kỳ phức tạp và yêu cầu cơ sở dữ liệu quản lý cấu hình (CMDB) vượt xa cơ sở dữ liệu quan hệ. Cơ sở dữ liệu đồ thị Neo4j CMDB giúp bạn tương quan mạng, trung tâm dữ liệu và tài sản CNTT để đơn giản hóa việc khắc phục sự cố, phân tích tác động và lập kế hoạch dung lượng hoặc ngừng hoạt động.

Cơ sở dữ liệu đồ thị như Neo4j cho phép bạn kết nối các công cụ giám sát và đạt được những hiểu biết quan trọng về các mối quan hệ phức tạp giữa các hoạt động mạng hoặc trung tâm dữ liệu khác nhau. Sử dụng không giới hạn cho đồ thị trong các hoạt động mạng và CNTT.

4.3 Công cụ khuyến nghị

Supply Chain Example Graph



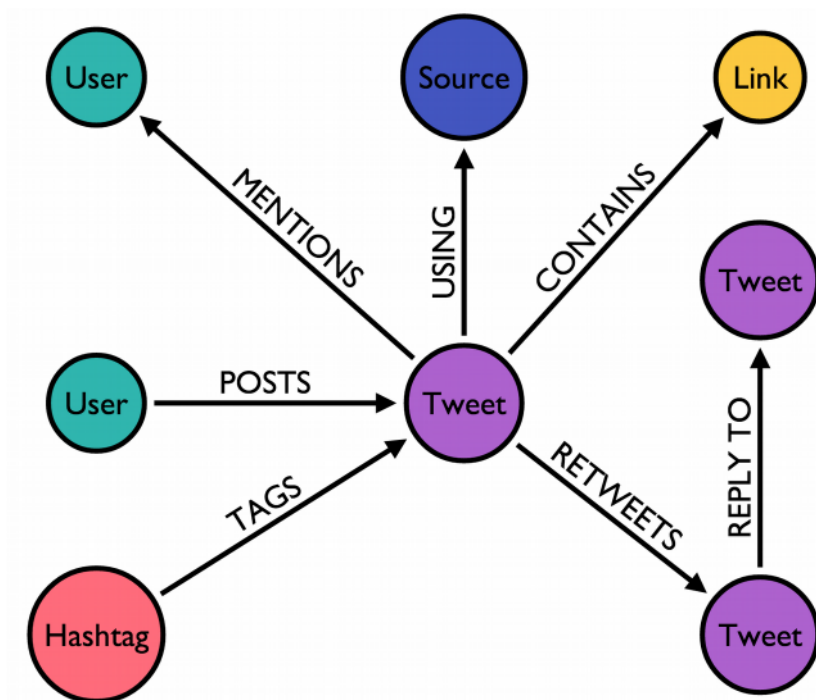
Công cụ đề xuất thời gian thực phải có khả năng tương quan với dữ liệu sản phẩm, hàng tồn kho, khách hàng, nhà cung cấp, hậu cần và thậm chí cả tình cảm xã hội để hoạt động hiệu quả nhất. Hơn nữa, họ sẽ có thể nắm bắt ngay lập tức bất kỳ sở thích mới nào theo lượt truy cập của khách hàng mới.

Công nghệ quan trọng cho phép các công cụ khuyến nghị làm như vậy là cơ sở dữ liệu đồ thị như Neo4j. Nó nhanh chóng bỏ lại các cơ sở dữ liệu quan hệ truyền thống và kết nối lượng lớn dữ liệu sản phẩm và khách hàng.

4.4 Các ứng dụng mạng xã hội

Cơ sở dữ liệu biểu đồ xã hội như Neo4j giúp tạo ra các mạng xã hội sáng tạo hoặc tích hợp các biểu đồ xã hội hiện tại vào một ứng dụng doanh nghiệp. Sự thật là các mạng truyền thông xã hội đã được xây dựng bằng các biểu đồ và các mối quan hệ; vì vậy không ích gì khi thay đổi chúng từ đồ thị thành bảng và sau đó quay lại lần nữa.

Mô hình dữ liệu có thể khớp trực tiếp với mô hình miền của bạn sẽ giúp bạn hiểu cơ sở dữ liệu của mình, giao tiếp tốt hơn và giảm bớt công việc không cần thiết. Neo4j đẩy nhanh hiệu suất của ứng dụng mạng xã hội của bạn bằng cách giảm thời gian cần thiết cho việc lập mô hình dữ liệu.



4.5 AI và phân tích

Các doanh nghiệp hiện đại ngày nay đang phải đối mặt với những thách thức vô cùng phức tạp và đòi hỏi những công nghệ thông minh. Neo4j trong trường hợp này cải thiện các dự đoán giúp đưa ra quyết định tốt hơn và có thể đổi mới. Nó kết hợp sức mạnh dự đoán của các mối quan hệ và cấu trúc mạng trong dữ liệu hiện tại để trả lời các câu hỏi khó và tăng độ chính xác của dự đoán.

Các thuật toán đồ thị Neo4j tìm ra các mẫu thiết yếu trong cấu trúc toàn cục và đưa ra dự đoán về đồ thị có thể với sự trợ giúp của những đồ thị và đào tạo máy học cơ sở dữ liệu đồ thị bên trong không gian làm việc phân tích. Đó là cách doanh

ng nghiệp có thể tăng cường các mối quan hệ có tính dự đoán cao và cấu trúc mạng lưới để trả lời các câu hỏi không phổ biến.

Các tổ chức sử dụng kết quả của các thuật toán đồ thị và các tính năng dự đoán của Neo4j để phân tích sâu hơn, học máy hoặc hỗ trợ các hệ thống trí tuệ nhân tạo. Đồ thị thường mang lại giá trị tuyệt vời cho phân tích nâng cao, máy học và AI.

4.6 Những lĩnh vực sử dụng Neo4j nhiều nhất

Ngày nay, các tổ chức lớn nhất trên thế giới đang sử dụng Neo4j để tối ưu hóa việc quản lý vô số điểm dữ liệu của họ. Neo4j là nhà cung cấp hàng đầu thế giới về công nghệ đồ thị có thể mở rộng, giúp 75% các công ty trong danh sách Fortune 100 nâng cao các ứng dụng dữ liệu được kết nối của họ.

Các ngành và công ty lớn nhất sử dụng công nghệ Neo4j hiện nay như sau:

- 7 trong số 10 nhà bán lẻ hàng đầu thế giới như eBay, ADEO và ATPCO
- 3 trong số 5 nhà sản xuất máy bay hàng đầu thế giới như Airbus
- 8 trong số 10 công ty bảo hiểm hàng đầu thế giới như Bayerische và Allianz
- Tất cả 20 ngân hàng hàng đầu của Bắc Mỹ như JP Morgan, Citi, Chase và UBS
- 8 trong số 10 nhà sản xuất ô tô hàng đầu thế giới như Volvo, Toyota và Daimler
- 3 trong số 5 khách sạn hàng đầu thế giới như Marriott và AccorHotels
- 7 trong số 10 công ty viễn thông hàng đầu thế giới như Verizon, Orange, AT&T và Comcast

CHƯƠNG 5. ỨNG DỤNG DEMO

5.1 Mục tiêu

Sử dụng một ứng dụng web đơn giản để chạy các truy vấn trên Neo4j. Tìm các phim có trong dữ liệu và hiển thị những thông tin có liên quan đến phim tương đương với các mối quan hệ trong dữ liệu.

Cho phép tìm phim theo tên phim, liệt kê toàn bộ các phim, hiển thị quan hệ dữ liệu dưới dạng list

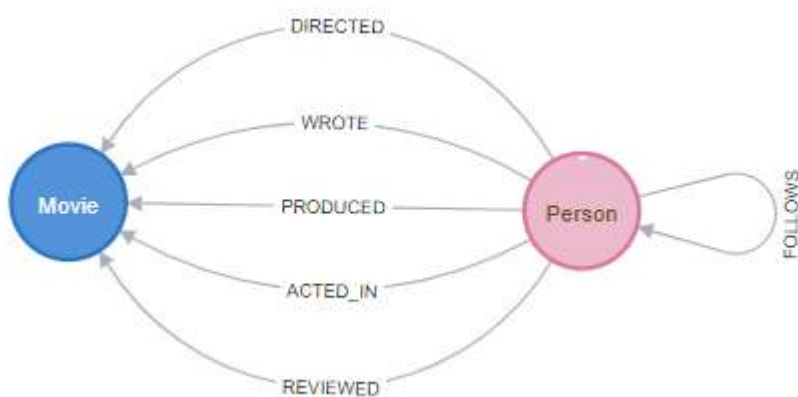
Yêu cầu để chạy được ứng dụng:

- Application Type: Java-Web Application
- Web framework: Spark-Java (Micro-Webframework)
- Neo4j Database Connector: Neo4j-Java-Driver for Cypher Docs
- Database: Neo4j-Server (4.x) with multi-database
- Frontend: jquery, bootstrap, d3.js

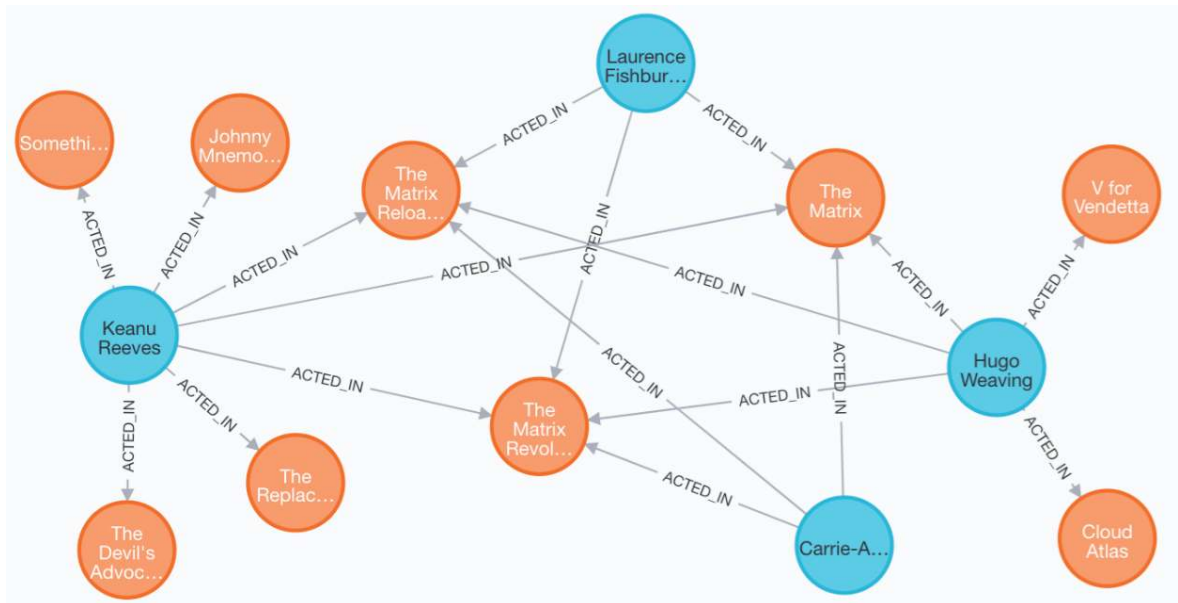
5.2 Chức năng

Sử dụng dữ liệu movie để xem mối quan hệ giữa phim và người trong đó bao gồm: diễn viên, đạo diễn, nhà sản xuất...

`(:Person {name: string})-[:ACTED_IN {roles: [string]}]->(:Movie {title: string, released: number})`



Một phần đồ thị trong dữ liệu được sử dụng.



Chức năng tìm kiếm phim:

MATCH (movie:Movie)

WHERE lower(movie.title) CONTAINS \$query

RETURN movie

Dữ liệu trả về khi tìm kiếm là The Matrix:

```

{"title": "The Matrix",
 "cast": [{"job": "acted", "role": ["Emil"], "name": "Emil Eifrem"},
 {"job": "acted", "role": ["Agent Smith"], "name": "Hugo Weaving"}, ...
 {"job": "directed", "role": null, "name": "Andy Wachowski"},
 {"job": "produced", "role": null, "name": "Joel Silver"}]}
  
```

Khi ứng dụng tìm được phim có tựa đề khớp với dữ liệu tìm kiếm là tên phim, toàn bộ thông tin của những người liên quan đến phim sẽ được trả về.

MATCH (movie:Movie {title:\$title})

OPTIONAL MATCH (movie)--[rel]--(person:Person)

RETURN movie.title as title,

```
collect( {name:person.name, role:rel.roles,  
job:head(split(toLower(type(rel)),'_'))}) as cast  
LIMIT 1
```

5.3 Thiết kế cơ sở dữ liệu

Các thực thể trong node trong nhãn trong neo4j gồm: Movie và Person.

Mối quan hệ giữa Person -> Movie

Nodes (171)

Movie Person

Relationships (253)

ACTED_IN DIRECTED FOLLOWS
PRODUCED REVIEWED WROTE

Mỗi người có thể đóng vai trong phim, đạo diễn, theo dõi, nhà sản xuất, biên kịch hoặc nhà phê bình.

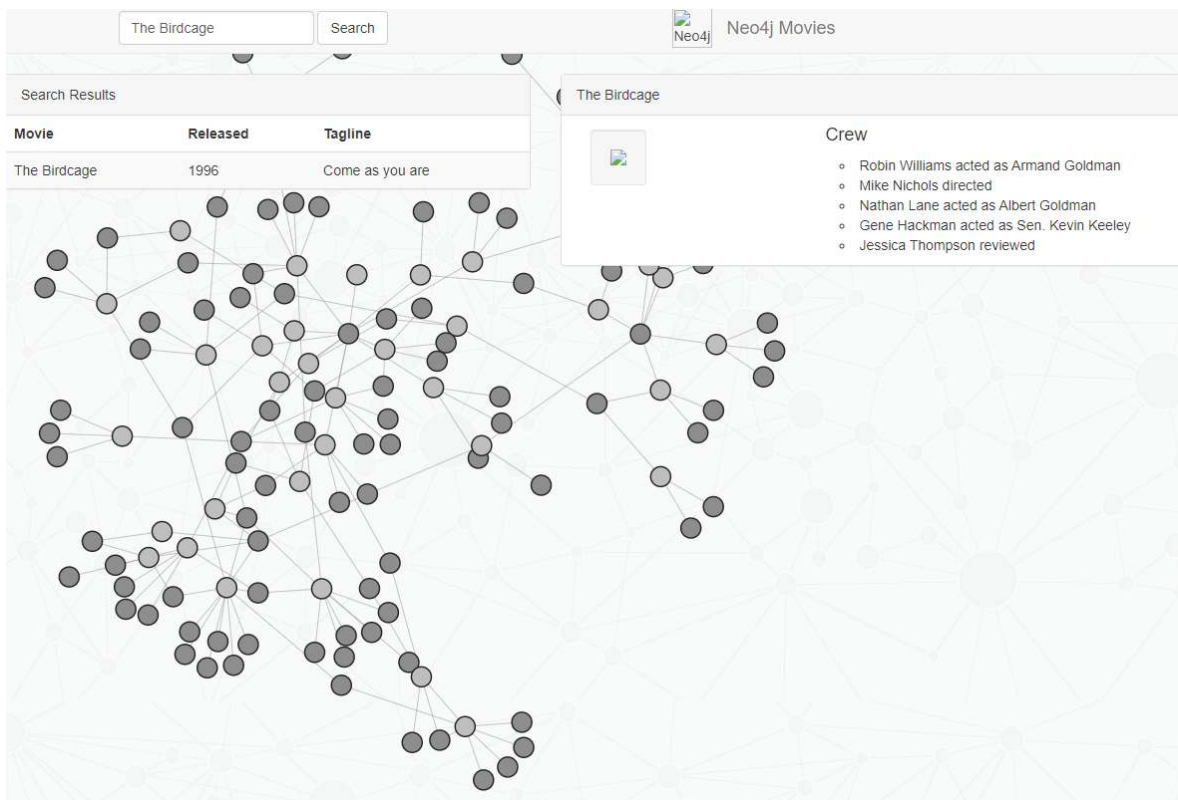
Các thuộc tính được sử dụng trong dữ liệu bao gồm: năm sản xuất, tên, đánh giá, vai diễn, ...

Property keys

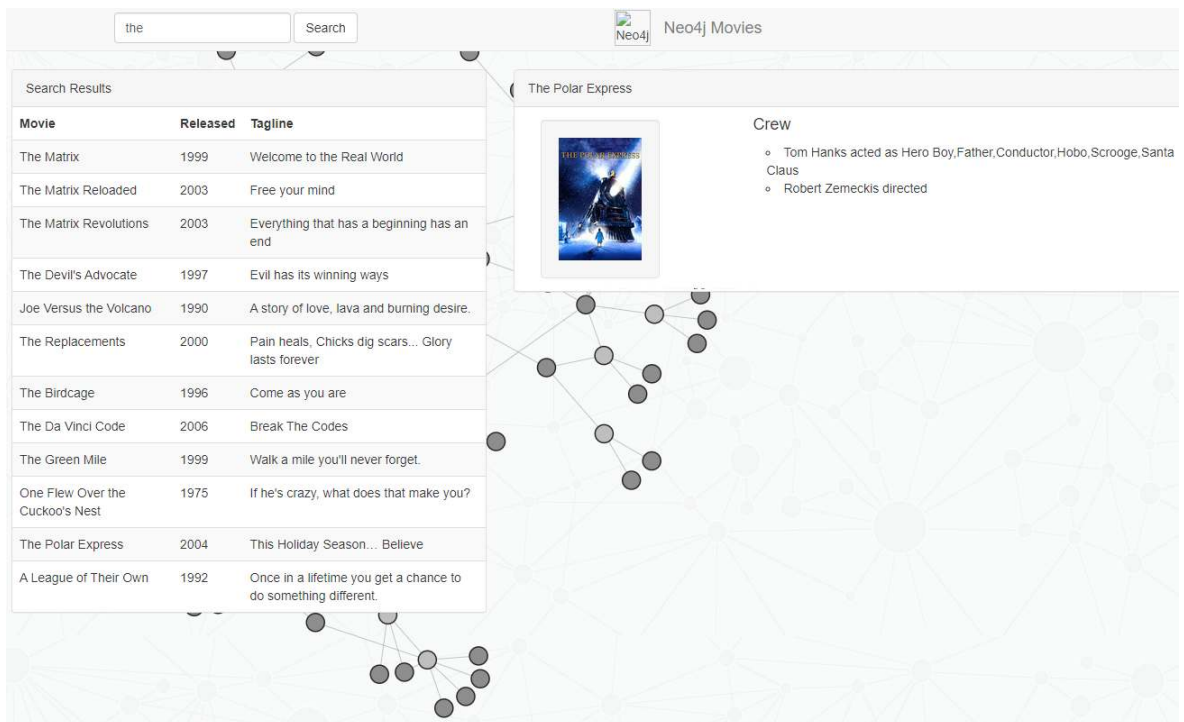
born name rating released roles
summary tagline title

5.4 Giao diện

Trang chủ của ứng dụng web



Tìm tên phim bắt đầu bằng “The”



CHƯƠNG 6. KHÓ KHĂN VÀ KẾT QUẢ ĐẠT ĐƯỢC

6.1 Khó khăn

Do hạn chế về thời gian cũng như lần đầu tiên tìm hiểu về Neo4j, nên bài báo cáo còn nhiều hạn chế cũng như chưa tìm hiểu sâu được khả năng ứng dụng và mở rộng của Neo4j

Chưa tìm hiểu được các ứng dụng cao cấp của phiên bản trả phí.

6.2 Kết quả đạt được

Tìm hiểu được cách sử dụng và ứng dụng Neo4j.

Tìm hiểu được quy trình chạy Neo4j, tạo dữ liệu và sao lưu dữ liệu.

Tìm hiểu được khả năng ứng dụng của dữ liệu đồ thị trong lĩnh vực IT hiện nay.

Phân tích ưu điểm và nhược điểm của dữ liệu đồ thị so với các dạng dữ liệu khác từ đó có thể lựa chọn cơ sở dữ liệu phù hợp cho việc phát triển ứng dụng

TÀI LIỆU THAM KHẢO

- [1] Tài liệu Hệ cơ sở dữ liệu nâng cao, PGS. TS. Nguyễn Thị Thúy Loan.
- [2] <https://neo4j.com/developer/cypher/#:~:text=Cypher%20is%20Neo4j's%20graph%20query,other%20standard%20data%20access%20languages.> (20/12/2022)
- [3] https://go.neo4j.com/rs/710-RRC-335/images/EMA_Neo4j_Two_Deployments-WP.pdf?__gl=1*19h1ofz*_ga*OTY0NjMzODYxLjE2Njg2OTg4MDc.*_ga_DL38Q8KGQC*MTY3MjQ1ODQ5MC4xMC4xLjE2NzI0NTg3NDcuMC4wLjA.&_ga=2.29712575.1420071836.1672418341-964633861.1668698807 (20/12/2022)
- [4] Neo4j in Action By [Aleksa Vukotic](#), [Nicki Watt](#), [Tareq Abedrabbo](#), [Dominic Fox](#) and [Jonas Partner](#) (10/12/2022)
- [5] <https://app.pluralsight.com/library/courses/graph-databases-neo4j-introduction/table-of-contents> (15/12/2022)