ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

KHOA HỆ THỐNG THÔNG TIN

□□□

Báo cáo thực hành:

BÀI TẬP LỚN 2

CƠ CHẾ PHÂN TÁN TRONG HỆ QUẢN TRỊ NOSQL - NEO4J

Môn học:

CƠ SỞ DỮ LIỆU PHÂN TÁN

Giảng viên giảng dạy: **Nguyễn Minh Nhựt**

Lớp: IS211.N11.HTCL

Sinh viên thực hiện:

Nguyễn Thị Mỹ Trân	-	20520322
Tôn Nữ Tú Quyên	-	20520296
Thái Tăng Đức	-	20521203
Trần Anh Huy	-	20520551

TP.HCM, tháng 12 năm 2022

LỜI CẢM ƠN

Lời đầu tiên, nhóm xin cảm ơn thầy Nguyễn Minh Nhựt đã cung cấp kiến thức để chúng tôi có thể thực hiện bài tập này cũng như những lời khuyên nhiệt tình, chân thật và luôn hữu ích của thầy. Kịp thời trả lời các câu hỏi của chúng tôi. Nếu không có sự hướng dẫn của anh Nhựt, chúng tôi nghĩ phần báo cáo rất khó hoàn thành. Đây cũng là cơ hội để mỗi thành viên trong nhóm làm việc với những người bạn mới, học hỏi thêm kỹ năng làm việc nhóm, học hỏi lẫn nhau và quan trọng là có cơ hội thực hiện sản phẩm thông qua khóa học.

Trong quá trình thực hiện dự án, nhóm áp dụng những điều đã học được đồng thời áp dụng những điều mới với mong muốn hoàn thành công việc một cách hoàn hảo nhất. Nhưng thời gian, kiến thức và kinh nghiệm còn hạn chế, không tránh khỏi những thiếu sót, chúng tôi rất mong nhận được sự góp ý quý báu của các thầy cô, các anh chị đi trước để nhóm bổ sung và hoàn thiện kiến thức, phục vụ tốt hơn cho đồ án và thực tiễn sau này công việc.

MỤC LỤC

A. GIỚI THIỆU VỀ HỆ QUẢN TRỊ CSDL NOSQL

1. Lịch sử ra đời và nguồn gốc

***Nguồn gốc:**

Neo4j là hệ quản trị cơ sở dữ liệu đồ thị, được phát triển bởi Neo4j, Inc. Hiện nay, neo4j là một trong các cơ sở dữ liệu đồ thị mã nguồn mở phổ biến nhất.

Neo4j có sẵn trong một GPL3-licensed mã nguồn mở "phiên bản cộng đồng", với sao lưu trực tuyến và sẵn sàng cao mở rộng cấp phép theo giấy cấp phép thương mại mã nguồn đóng. Neo cũng cung cấp cho Neo4j với các tiện ích mở rộng này theo các điều khoản thương mại nguồn đóng

Neo4j được triển khai bằng Java và có thể truy cập được từ phần mềm được viết bằng các ngôn ngữ khác bằng ngôn ngữ truy vấn Cypher thông qua điểm cuối HTTP giao dịch hoặc thông qua giao thức "bu lông " nhị phân.

Neo4j là một cơ sở dữ liệu đồ thị, lưu trữ dữ liệu trong các node và các quan hệ của một đồ thị. Các cấu trúc dữ liệu chung nhất là đồ thị, biểu diễn bất kỳ kiểu dữ liệu nào, đảm bảo cấu trúc tự nhiên của một cơ sở dữ liệu. Neo4j không giống như cơ sở dữ liệu truyền thống, nó là NOSQL(Not Only SQL), nghĩa là không chỉ truy vấn dữ liệu bằng các câu sql thông thường, chúng ta truy vấn trong một cơ sở dữ liệu đồ thị dựa trên phép duyệt đồ thị.

Neo4j một dự án mã nguồn mở dùng trong cộng đồng GPLv3, được hỗ trợ bởi công ty Neo Technology.

***Lịch sử:**

- Năm 2000: Những người sáng lập Neo4j là Emil, Johan và Peter đã gặp phải các vấn đề về hiệu suất với RDBMS và bắt đầu xây dựng Neo4j prototype đầu tiên

- Năm 2002: Đã phát triển phiên bản Neo4j đầu tiên
- Năm 2003: Triển khai Neo4j sản xuất 24 × 7 đầu tiên
- Năm 2007: Thành lập một công ty có trụ sở tại Thụy Điển đứng sau Neo4j. Cũng mở nguồn cơ sở dữ liệu đồ thị đầu tiên, Neo4j, theo GPL
- Năm 2009: Tăng vốn tài trợ 2,5 triệu đô la, từ Sunstone và Conor và tiếp tục phát triển. 2000 khách hàng toàn cầu đầu tiên.
- Năm 2010: Phát hành phiên bản Neo4j 1.0 vào tháng 2
- Năm 2011: Tăng một vòng và chuyển trụ sở đến Thung lũng Silicon
- Năm 2012: Đã huy động được 11 triệu đô la Series B từ Fidelity, Sunstone và Conor GraphConnect, hội nghị đầu tiên về cơ sở dữ liệu đồ thị.
- Năm 2013: Neo4j phiên bản 2.0 được phát hành vào tháng 12.
- Năm 2015: Đã huy động được 20 triệu đô la Series C từ Creandum với Dawn và các nhà đầu tư hiện tại. Hơn 2 triệu lượt tải xuống Neo4j
- Năm 2016: Neo4j 3.0 đã phát hành. $ 36 triệu Series D từ Greenbridge Investment
- Năm 2017: Máy tính để bàn Neo4j đã phát hành. Neo4j công bố Nền tảng đồ thị đầu tiên của ngành.
- Năm 2018: Neo4j Bloom đã phát hành. 80 triệu đô la Series E do Morgan Stanley Mở rộng Capital & One Peak Partners dẫn đầu.
- Năm 2019: Neo4j AuraDB ra mắt
- Năm 2020:
    - Neo4j đã được công nhận là người dẫn đầu trong The Forrester Wave ™ cho nền tảng dữ liệu đồ thị, Q4 2020.
    - Bản phát hành GA của AuraDB Enterprise.
    - Ra mắt Thư viện Khoa học Dữ liệu Đồ thị Neo4j
    - Đồ thị hình thành 4 COVID-19

    - Hội chợ triển lãm và hội nghị dành cho nhà phát triển trực tuyến Neo4j (NODES) đã đánh dấu sự kiện đồ thị kỹ thuật số lớn nhất của chúng tôi
    - Ra mắt Graphs 4 COVID-19 “GraphHack”, một sáng kiến mới của Chương trình Graphs4Good để giúp chống lại sự lây lan của vi rút.
- Năm 2021:
    - Đã huy động được 390 triệu đô la Series F do Eurazeo dẫn đầu với GV, Inovia Capital và các nhà đầu tư hiện tại; đây là khoản đầu tư lớn nhất trong lịch sử cơ sở dữ liệu
    - Ra mắt Neo4j AuraDB Enterprise
    - Neo4j phá vỡ rào cản quy mô với biểu đồ quan hệ nghìn tỷ
    - Ra mắt Neo4j AuraDB miễn phí
    - Năm 2022: Ra mắt Neo4j AuraDS (Khoa học dữ liệu đồ thị như một dịch vụ)

***Cấp phép và phiên bản:**

Neo4j có hai phiên bản: Cộng đồng và Doanh nghiệp. Nó được cấp phép kép: GPL v3 và giấy phép thương mại. Phiên bản Cộng đồng miễn phí nhưng chỉ giới hạn chạy trên một nút do thiếu phân cụm và không có bản sao lưu nóng.

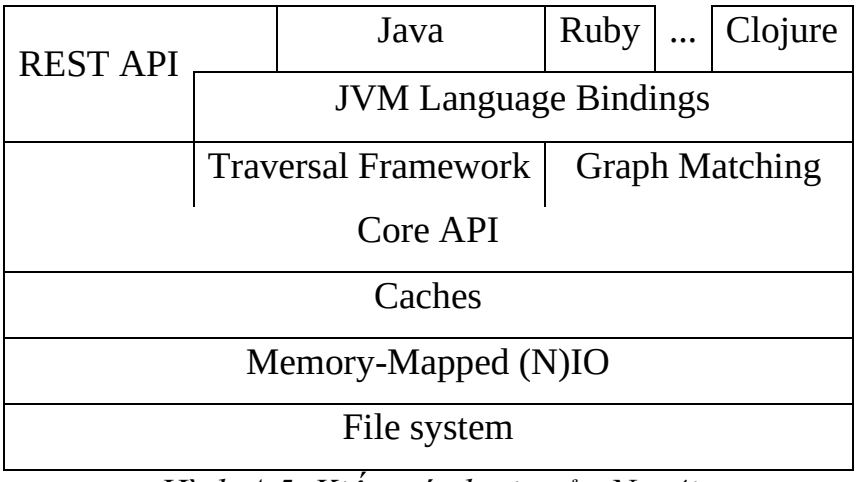
Phiên bản doanh nghiệp mở ra những hạn chế này, cho phép phân cụm, sao lưu nóng và giám sát. Phiên bản doanh nghiệp có sẵn theo giấy phép Thương mại nguồn đóng.

***Đặc điểm:**

- Neo4j cung cấp công nghệ đồ thị đã được kiểm tra hiệu suất và mở rộng quy mô.

- Kiến trúc cụm phân tán hiệu suất cao của Neo4j cho phép khách hàng chạy khối lượng công việc khoa học dữ liệu và OLTP thách thức nhất, đồng thời duy trì tính tuân thủ ACID và tính toàn vẹn của dữ liệu.
- Neo4j có thể xử lý các đồ thị với hàng tỉ các node/ các mối quan hệ/ các thuộc tính trên 1 máy tính và có thể mở rộng ra trên nhiều máy khác.
- Với Neo4j, khách hàng có thể tự do lựa chọn triển khai trên nền tảng tự lưu trữ, kết hợp hoặc đa đám mây.
- Một framework duyệt mạnh mẽ với tốc độ duyệt trên các node cực nhanh trong không gian node. Độ sâu của quá trình duyệt có thể lên đến 1000 mức và dưới tốc độ 1 giây.
- Full transactional như một cơ sở dữ liệu thực sự, với đầy đủ các đặc tính ACID (Atomicity, Consistency, Isolation, Durability).

2. Kiến trúc của Neo4j:



Hình A.5. Kiến trúc logic của Neo4j

Trong đó:

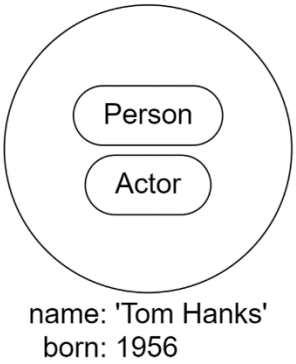
- File system: là các file trên ổ cứng, được lưu trữ cẩn thận để tính toán các offset và tìm đến bất kì một record nào trong các file một cách nhanh nhất. Ta

lưu trữ tách biệt các node, các quan hệ và các thuộc tính, và tối ưu trong những trường hợp chung nhấp để đảm bảo dữ liệu được tìm thấy trên 1 file.

- Memory-mapped (N) IO: ta sử dụng java IO cho mục đích nhanh chóng.
- Caches: cho phép làm việc nhanh chóng trên các đĩa quay, cho phép chúng ta duyệt hàng triệu phép duyệt mỗi giây trên một phần cứng của máy laptop.
- Core API: là phần nhân của Neo4j, lưu trữ các cấu trúc mức trừu tượng của đồ thị, mang tính hiệu quả cao.
- Traversal Framework: tầng truy vấn dữ liệu.
- JVM Language Bindings: các thành phần của Java API như Jruby, Jython, Scala,...

3. Node

- Node mô tả những thực thể (các đối tượng rời rạc) của một miền.
- Node có thể có 0 hoặc nhiều label để định nghĩa (phân loại) đó là loại node nào.
- Đồ thị đơn giản nhất là đồ thị chỉ chứa một node và không có relationship. Ta xem xét đồ thị chỉ chứa duy nhất một node dưới đây:



Hình A.6. Ví dụ một node

- Các node label bao gồm:
    - Person
    - Actor

- Các thuộc tính bao gồm:
    - name: Tom Hanks
    - born: 1956

4. Relationship

Một relationship mô tả cách kết nối giữa node nguồn và node đích với nhau. Ngoài ra, một node cũng có thể có một relationship tới chính nó.

Đặc điểm của relationship:

- Kết nối node nguồn và node đích.
- Có hướng (một chiều)
- Phải có type (một type) để định nghĩa (phân loại) đó là loại relationship nào.
- Có thể có các thuộc tính (cặp key-value) mô tả thêm về relationship.

Relationship sắp xếp các node theo những cấu trúc, cho phép một đồ thị trở nên giống như một danh sách, một cây, một map, hoặc một thực thể phức hợp – bất cứ thứ gì trong số đó có thể được kết hợp thành các cấu trúc phức tạp hơn, được kết nối phong phú hơn.

Hình A.7.1. Ví dụ relationship giữa 2 node

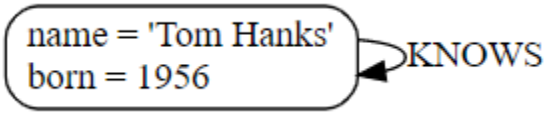
- Relationship type: ACTED_IN
- Thuộc tính:

- roles: ['Forrest']
- performance: 5

Thuộc tính roles có một mảng giá trị với item (‘Forrest’) duy nhất.

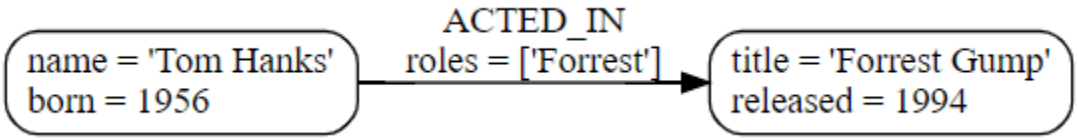
Relationship luôn luôn có hướng. Tuy nhiên, hướng có thể được bỏ qua nếu nó không cần thiết. Điều này có nghĩa là ta không nhất thiết phải thêm các relationship trùng lặp theo hướng ngược lại trừ khi cần phải mô tả data model một cách chính xác.

Một node có thể có relationship tới chính nó. Giả sử ta muốn thể hiện rằng Tom Hanks KNOWS chính bản thân mình, ta có thể vẽ relationship như sau:



Hình A.7.2. Ví dụ relationship của 1 node tới chính nó

Một relationship cần phải có đúng một relationship type. Dưới đây là ví dụ về relationship ACTED_IN, có node Tom Hanks là node nguồn và node Forrest Gump là node đích:



Hình A.7.3. Ví dụ relationship type

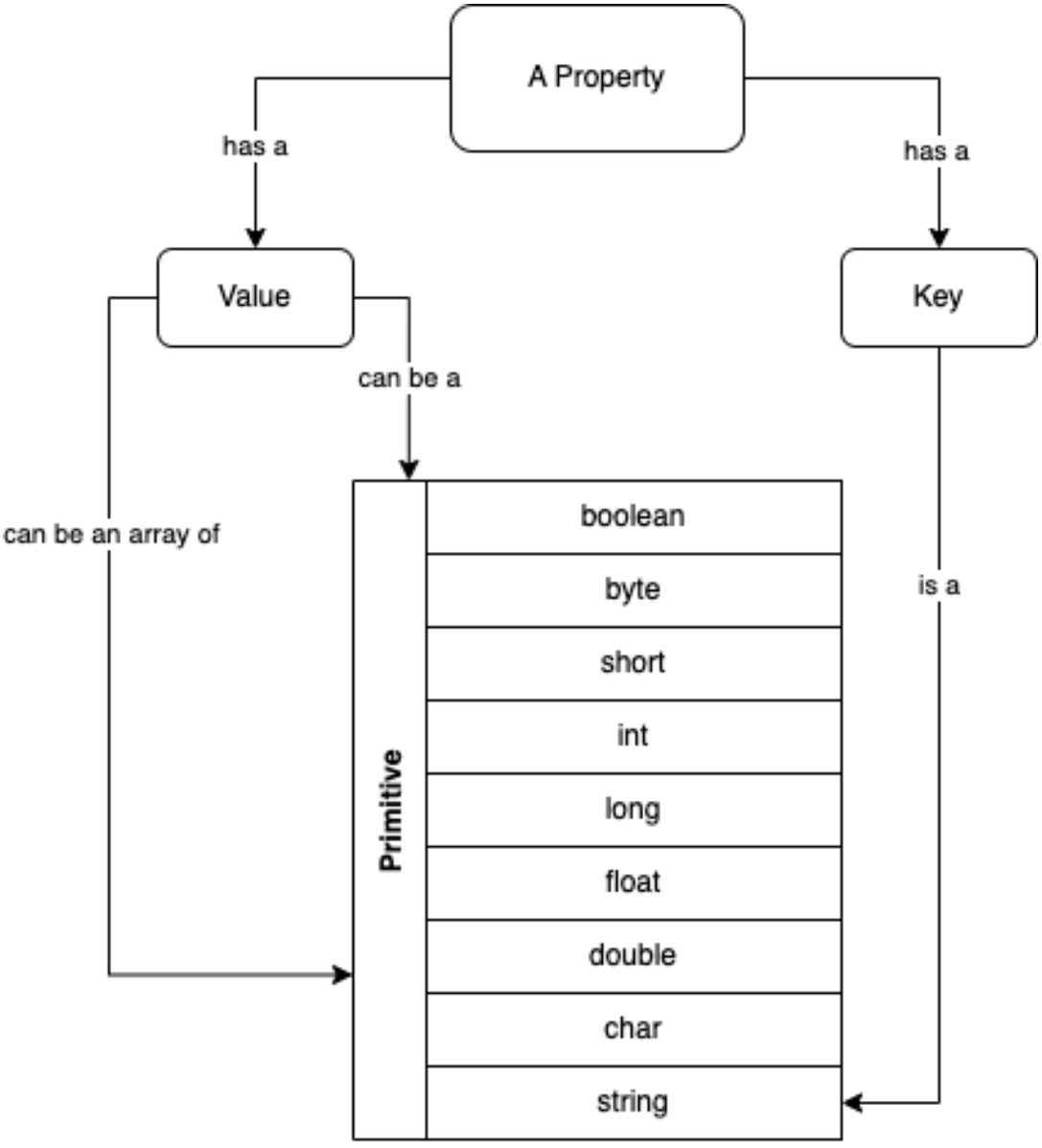
Quan sát và biết được rằng node Tom Hanks có một outgoing relationship, trong khi đó node Forrest Gump có một incoming relationship.

5. Properties

Cả node và relationships đều có các thuộc tính (properties).

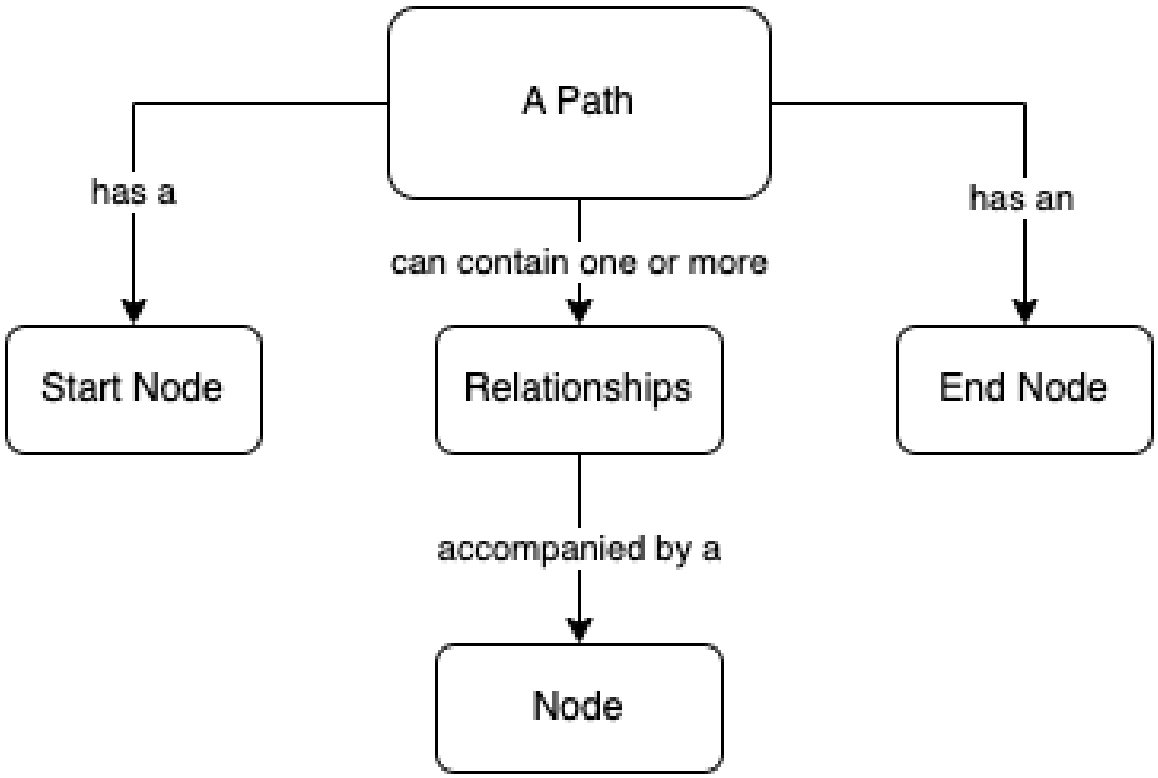
Properties là cặp key – value (khóa – giá trị) trong đó key có kiểu String (kiểu mô tả này tương tự như kiểu dữ liệu Map trong Java). Các giá trị của property có thể là 1 kiểu nguyên thủy, hoặc một mảng của nhiều kiểu nguyên thủy. Ví dụ: kiểu String, kiểu int, hoặc mảng int[].

Properties không chứa giá trị null, nếu một thuộc tính có giá trị = null, nghĩa là không tồn tại key đó trong các properties của node hoặc relationships.

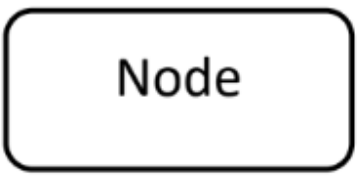


6. Đường đi Paths:

Một đường đi là một hay nhiều node được kết nối với nhau bởi các relationships, thường được biểu diễn bằng kết quả của phép duyệt.



Đường đi ngắn nhất có thể bằng 0 khi đó là đường đi từ 1 node đến chính nó, như hình sau:



7. Duyệt đồ thị

Duyệt đồ thị là cách truy vấn đồ thị, điều hướng từ đầu từ một nút đến các node có liên quan để trả lời cho các câu hỏi dạng như "Bạn của tôi thích thể loại nhạc nào?", Hoặc "Những người bạn của bạn, tôi là ai? ",…

Duyệt một thị trường đồ có nghĩa là đi thăm các node của thị đồ đó, dựa trên relationships giữa các node theo một số quy tắc. Trong một số trường hợp, chỉ có một đồ thị con được thăm. Có hai cách duyệt đồ thị cơ bản là duyệt theo chiều rộng và duyệt theo chiều sâu.

Thông thường ta duyệt đồ thị bằng cách sử dụng thuật toán Djikstra để tìm đường đi ngắn nhất từ đỉnh A đến đỉnh B, để tiết kiệm chi phí "đi lại" giữa các node.

Để thực hiện truy vấn trong một đồ thị CSDL, chúng ta thực hiện một đồ thị duyệt dựa trên một giải pháp để xác định đường đi theo thứ tự của các node.

8. Neo4j so với RDBMS và NoQuery khác

Các cửa hàng dữ liệu khác có các trường hợp sử dụng phù hợp. Nhưng bất cứ khi nào doanh nghiệp của bạn muốn tận dung các kết nối giữa các điểm dữ liệu, bạn cần khai thác sức mạnh của Neo4j.

Dưới đây là cách cơ sở dữ liệu đồ thị hàng đầu thế giới chống lại các cơ sở dữ liệu quan hệ truyền thống (RDBMS) và các kho dữ liệu NoQuery cạnh tranh khác.

* Neo4j so với cơ sở dữ liệu quan hệ (RDBMD)

Sự khác biệt đáng chú ý nhất giữa cả hai là cơ sở dữ liệu đồ thị lưu trữ các mối quan hệ giữa các dữ liệu dưới dạng dữ liệu. Cơ sở dữ liệu quan hệ suy ra tập trung vào mối quan hệ giữa các dữ liệu nhưng theo một cách khác. Trọng tâm quan hệ nằm giữa các cột của bảng dữ liệu, không phải điểm dữ liệu.

	Cơ sở dữ liệu quan hệ	**Neo4j, Cơ sở dữ liệu đồ thị gốc**
Định dạng	Bảng với hàng và cột	Các nút và các cạnh

Các mối quan hệ	Liên quan giữa các bảng, thiết lập bằng cách sử dụng khóa ngoại giữa các bảng	Được coi là dữ liệu, được biểu thị bằng các cạnh giữa các nút
Hiệu suất truy vấn	Hiệu suất xử lý dữ liệu chịu đựng số lượng và độ sâu của THAM GIA(hoặc các mối quan hệ được yêu cầu).	Xử lý đồ thị đảm bảo độ trễ bằng không và hiệu suất thời gian thực, bất kể số lượng hoặc độ sâu của mối quan hệ.
Ngôn ngữ truy vấn	Cần các truy vấn kết giữa các bảng	Chạy nhanh chóng và không cần phép kết
Trường hợp sử dụng	Trường hợp sử dụng các mối quan hệ nhiều và lớn.	Các trường hợp sử dụng trọng tâm giao dịch, bao gồm các giao dịch trực tuyến và kế toán

Neo4j so với các cơ sở dữ liệu NoQuery khác:

	Cơ sở dữ liệu NoQuery khác	**Neo4j, cơ sở dữ liệu đồ thị gốc**
Lưu trữ dữ liệu	Không hỗ trợ dữ liệu được kết nối ở cấp cơ sở dữ liệu. Hiệu suất và độ tin cậy của dữ liệu giảm theo quy mô và độ phức tạp của các kết nối.	Cấu trúc lưu trữ đồ thị gốc với tính phụ thuộc không có chỉ mục dẫn đến các giao dịch và xử lý nhanh hơn cho các mối quan hệ dữ liệu.
Mô hình hóa dữ liệu	Mô hình dữ liệu không phù hợp với kiến trúc doanh nghiệp vì các cột rộng và kho lưu trữ tài liệu	Mô hình dữ liệu linh hoạt, "thân thiện với bảng trắng" cho phép kiểm soát chi tiết

Hiệu suất truy vấn	Không có khả năng xử lý đồ thị cho các mối quan hệ dữ liệu, do đó tất cả các mối quan hệ phải được tạo ở cấp ứng dụng.	Xử lý đồ thị gốc đảm bảo độ trễ bằng không và hiệu suất thời gian thực, bất kể số lượng hoặc độ sâu của mối quan hệ.
Ngôn ngữ truy vấn	Ngôn ngữ truy vấn khác nhau, nhưng không có cấu trúc truy vấn tồn tại để thể hiện mối quan hệ dữ liệu.	Cypher : Một ngôn ngữ truy vấn biểu đồ gốc cung cấp cách thức hiệu quả và biểu cảm nhất để mô tả các truy vấn mối quan hệ.
Hỗ trợ giao dịch	Các giao dịch BASE dẫn đến tham nhũng dữ liệu vì tính khả dụng cơ bản và tính nhất quán cuối cùng là không đáng tin cậy cho các mối quan hệ dữ liệu.	Các giao dịch ACID đảm bảo dữ liệu hoàn toàn phù hợp và đáng tin cậy suốt cả ngày - hoàn hảo cho các ứng dụng doanh nghiệp toàn cầu luôn hoạt động.
Xử lý ở quy mộ	Tối ưu hóa để nhập dữ liệu nhưng không đọc dữ liệu ở quy mô. Khả năng mở rộng phụ thuộc vào kiến trúc mở rộng quy mô không bảo vệ tính toàn vẹn của dữ liệu giống như biểu đồ, vì vậy dữ liệu không đáng tin cậy.	Mô hình đồ thị nguyên bản vốn có tỷ lệ cho các truy vấn dựa trên mẫu. Kiến trúc quy mô duy trì tính toàn vẹn dữ liệu thông qua nhân rộng. Tăng cường khả năng mở rộng với các hệ thống IBM POWER8 và CAPI Flash.
Hiệu quả trung tâm dữ liệu	Hợp nhất máy chủ là có thể nhưng tốn kém cho quy mô kiến trúc. Kiến trúc quy mô là	Dữ liệu và các mối quan hệ được lưu trữ nguyên bản cùng với cải thiện hiệu suất khi độ phức tạp

	tốn kém về mua, sử dụng năng lượng và thời gian quản lý.	và quy mô tăng lên. Điều này dẫn đến việc hợp nhất máy chủ và sử dụng phần cứng cực kỳ hiệu quả.máy chủ và sử dụng phần cứng cực kỳ hiệu quả

B. NGÔN NGỮ CYPHER TRONG NEO4J

1. Khái niệm cơ bản

Cypher là ngôn ngữ truy vấn trong Neo4j. Các câu query là tập hợp các mệnh đề được liên kết với nhau.

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

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)

2. Định dạng của Cypher

Định dạng nút

- **()**: nút rỗng
- **(varname: NodeName)**: Nút có nhãn là NodeName, tên biến của nút là varname. Nút có thể không có tên biến.

Định dạng quan hệ:

[varname: RELATIONSHIP_NAME]: mối quan hệ có nhãn là RELATIONSHIP_NAME và biến quan hệ là varname.

3. Các lệnh cơ bản trong Cypher

* Match và Return:

MATCH: tìm kiếm theo pattern. OPTIONAL MATCH tương tự MATCH nhưng sẽ trả về kết quả Null nếu có missing trong pattern.

RETURN: Định nghĩa kết quả trả về.

Ví dụ: Cho label Person có 1 node với property {name: “Percy”}

- Cypher command:

  MATCH (p:Person{name: “Percy”}) RETURN p

***Create:**

CREATE: Tạo Node, Relationship hoặc một Path.

Ví dụ: Tạo thêm một node có nhãn Person với property {name: “Jack”}

- Cypher command:

  CREATE (p:Person {name: “Jack”})

  Để thiết lập mối quan hệ giữa hai node đã tạo ta sử dụng Match

- Cypher command:

  MATCH (Percy: Person {name: “Percy”})

  MATCH (Jack: Person {name: “Jack”})

  CREATE (Percy) – [rel: IS_FRIEND_WITH] -> (Jack)

*Lưu ý: Nếu không có 2 lệnh Match thì Cypher sẽ tự động tạo ra các node mới mà không kiểm tra xem nó đã tồn tại trong cơ sở dữ liệu hay chưa.

***Update:**

SET: Dùng để cập nhật Labels, Properties.

Sử dụng MATCH..SET để thêm, sửa, xóa các thuộc tính (property) trong 1 nút.

Ví dụ: Thêm thuộc tính birthday cho Person Percy

- Cypher command:

  MATCH (p:Person {name: “Percy”})

  SET p.birthday = date (“2002-01-01”)

***Delete:**

DELETE: Xóa Node, Relationship.

Ví dụ: Xóa mối quan hệ bạn bè giữa Percy và Jack dùng DELETE

- Cypher command:

MATCH (j: Person {name: “Percy”}) – [friend: IS_FRIEND_WITH] -> (m: Person {name: “Mark”})

DELETE friend

Ví dụ xóa đồng thời nút và mối quan hệ của nó dùng Detach Delete

- Cypher command:

MATCH (p: Person {name: “Jack”}) DETACH DELETE p

Xóa hoàn toàn thuộc tính và không lưu trữ nó nữa dùng REMOVE hoặc sử dụng từ khóa SET để đặt giá trị thuộc tính thành null (trong mô hình csdl Neo4j không lưu trữ giá trị null).

Ví dụ: Xóa thuộc tính birthday của Percy

MATCH (p: Person {name: “Percy”}) REMOVE p.birthday

Ví dụ: Đặt giá trị thuộc tính birthday thành null

MATCH (p: Person {name: “Percy”}) SET p.birthday = null

***Merge:**

MERGE: thực hiện chọn lọc và kiểm tra dữ liệu có tồn tại trong csdl hay không trước khi chèn vào csdl.

Ví dụ: Chèn Person Jack vào csdl bằng MERGE

Cypher command:

MERGE (jack: Person {name: “Jack”})

RETURN jack

Nút jack đã tồn tại trong csdl trước đó, nên câu lệnh trên sẽ không tạo thêm nút jack mới mà chỉ trả về nút jack đã có.

***WHERE:**

WHERE: Thêm các ràng buộc cho pattern.

Ví dụ: Tìm node Person có trường name = “Percy”

- Cypher command:

  MATCH (person: Person)

  WHERE person.name = “Percy” RETURN person

Ngoài ra, WHERE còn có thể đi cùng với AND, OR, XOR, NOT, IN.

- WHERE exists(property): kiểm tra xem thuộc tính hoặc mối quan hệ có tồn tại không.
- WHERE…IN: Kiểm tra xem giá trị thuộc tính có phải là giá trị trong danh sách đã cho.

***Xử lí chuỗi:**

- **STARTS WITH**: Tìm chuỗi bắt đầu bằng chuỗi bạn chỉ định.
- **CONTAINS**: kiểm tra xem chuỗi được chỉ định có phải 1 phần của giá trị thuộc tính không.
- **ENDS WITH**: tìm chuỗi có kết thúc bằng chuỗi bạn chỉ định.

- Cypher command:

  MATCH (varname: NodeLabel)

  WHERE varname.property STARTS WITH/ CONTAINS/ ENDS WITH yousr_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 "P".

- Cypher command:

  MATCH (p:Person)

  WHERE p.name = ~'P.*'

  RETURN p.name

***Function và Procedure:**

Ngoài các mệnh đề đã nêu ở trên, Cypher còn có các mệnh đề như WITH, UNWIND, ORDER BY, UNION, …

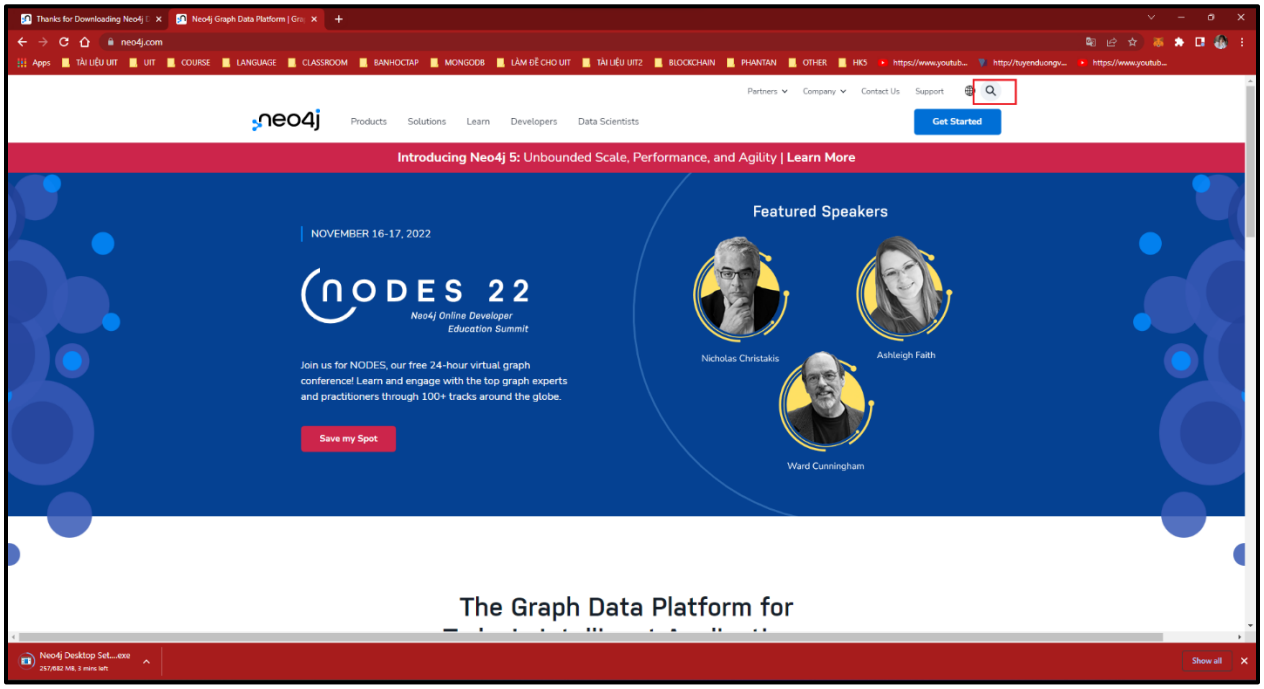
Cypher còn hỗ trợ các function như sum(), count(), max(), min(), avg(), … và một số procedure có sẵn.

C. CÀI ĐẶT TRÊN 2 MÁY TRỞ LÊN

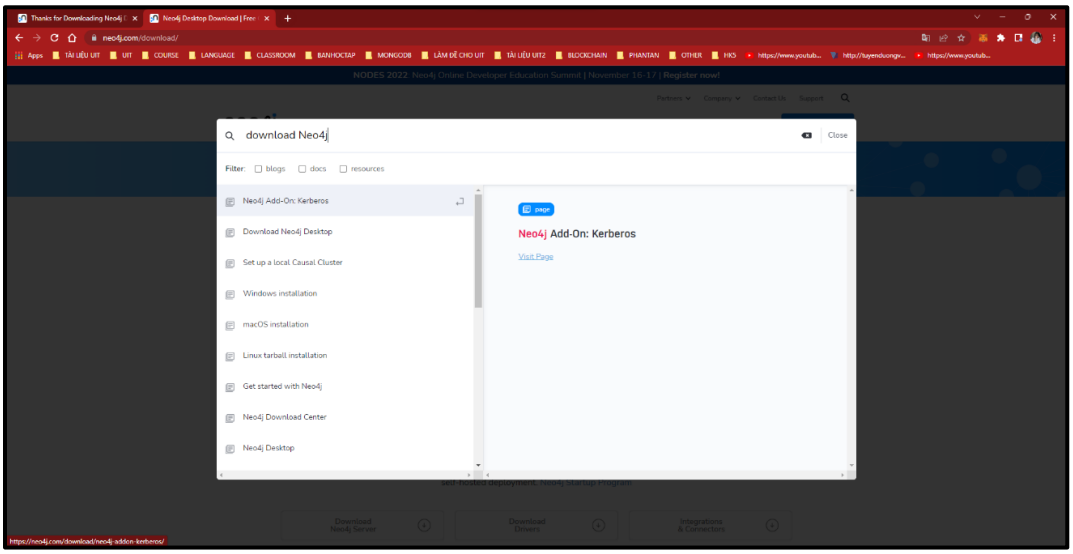
1. Cài đặt trên nhiều máy

❖ Tải Neo4j

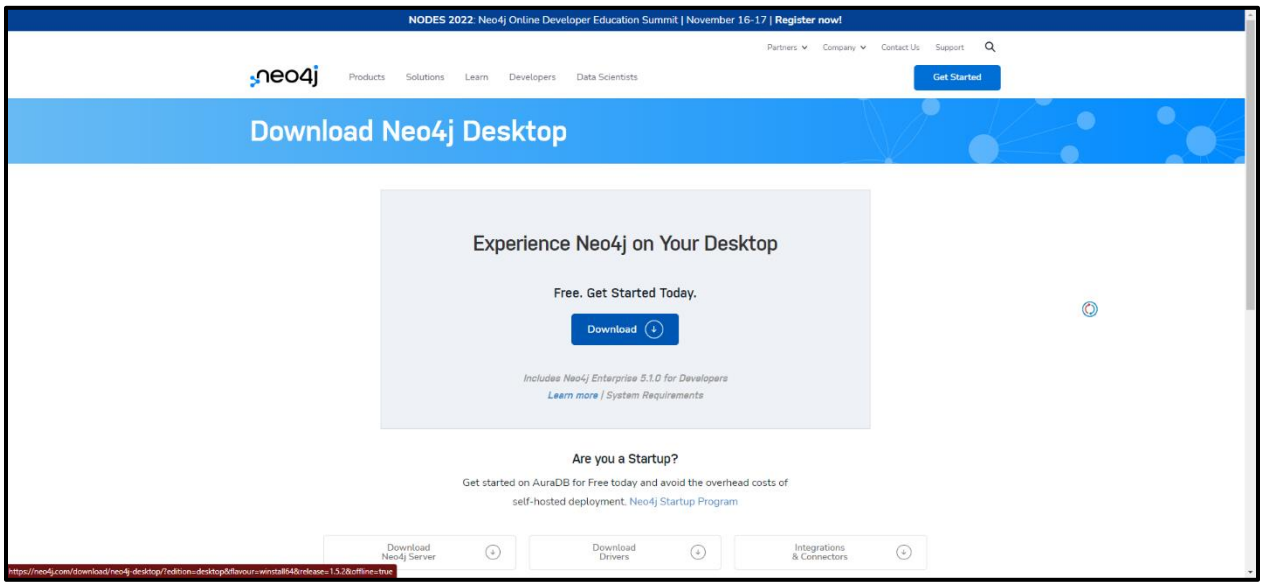
➢ Vào trang chủ Neo4j: https://neo4j.com/ -> Rồi ấn vào biểu tượng hình kính lúp như hình sau



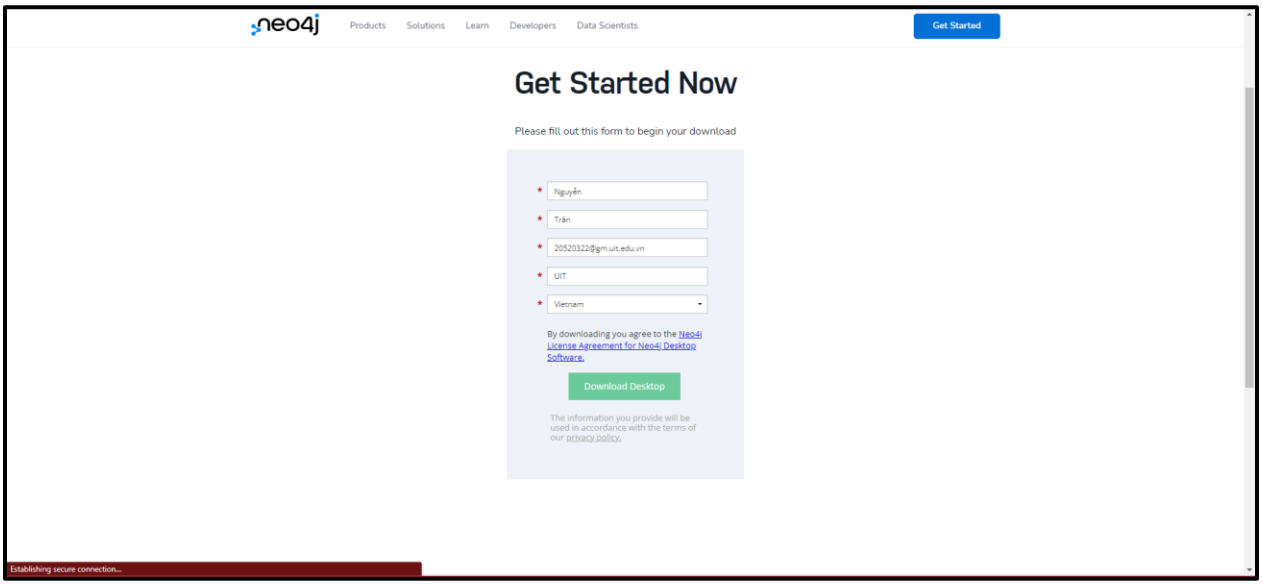
➢ Nhập download Neo4j vào ô tìm kiếm và chọn dòng chữ như trên hình

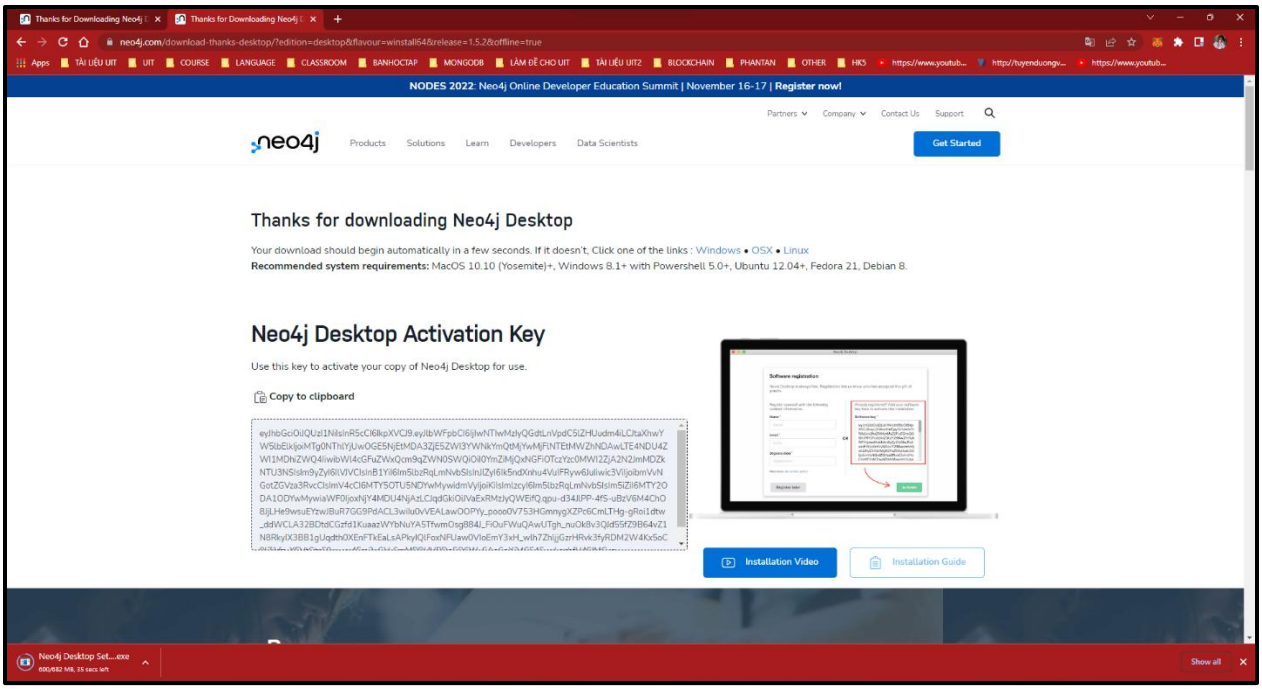


➢ Ấn "**Download**"



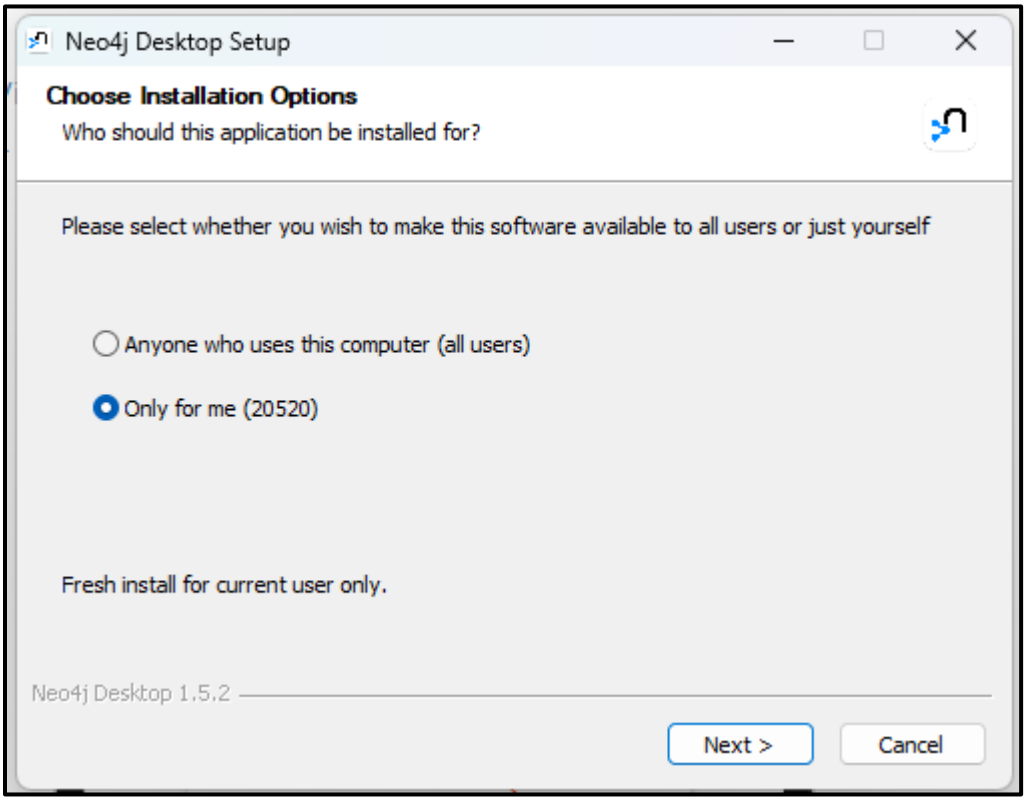
➢ Nhập đầy đủ thông tin rồi ấn vào "**Download Desktop**"



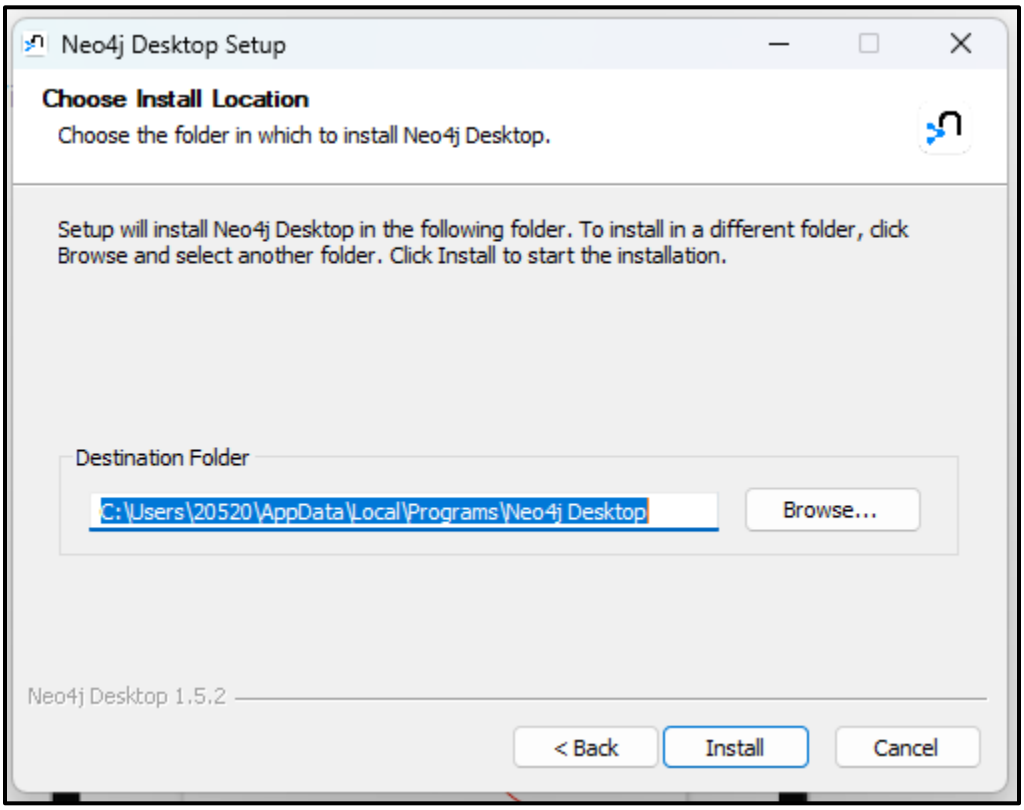

Thanks for Downloading Neo4j
neo4j.com/download-thanks-desktop/?edition=desktop&flavour=winstall64&release=1.5.2&offline=true
NODES 2022: Neo4j Online Developer Education Summit | November 16-17 | Register now!
Partners
Company
Contact Us
Support
neo4j
Products
Solutions
Learn
Developers
Data Scientists
Get Started
Thanks for downloading Neo4j Desktop
Your download should begin automatically in a few seconds. If it doesn't, Click one of the links : Windows • OSX • Linux
Recommended system requirements: MacOS 10.10 (Yosemite)+, Windows 8.1+ with Powershell 5.0+, Ubuntu 12.04+, Fedora 21, Debian 8.
Neo4j Desktop Activation Key
Use this key to activate your copy of Neo4j Desktop for use.
Copy to clipboard
Installation Video
Installation Guide
Neo4j Desktop Set....exe
Show all

❖ Cài đặt Neo4J

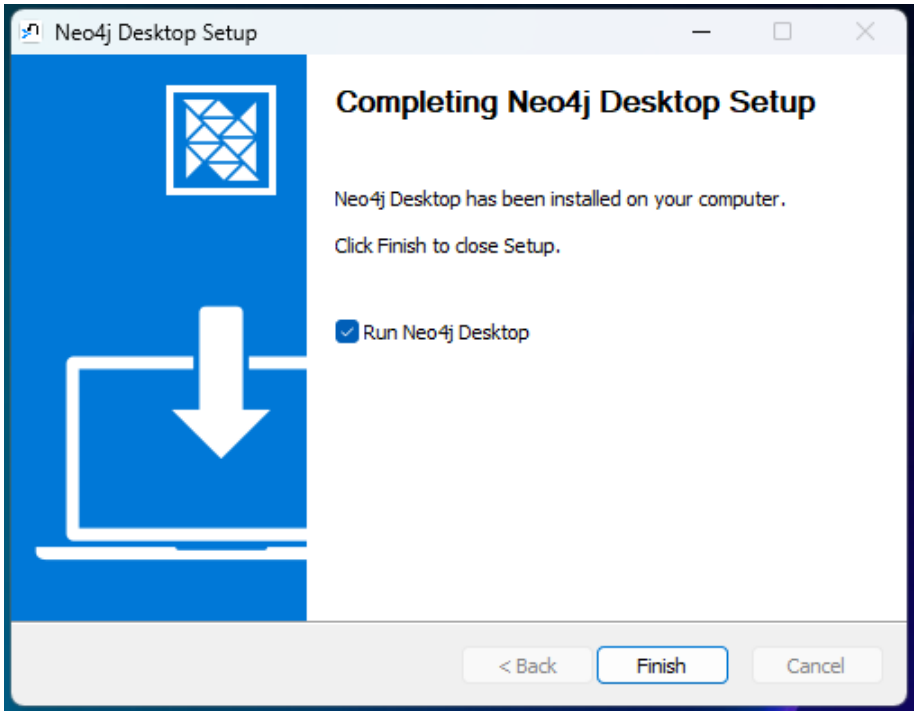
➢ Khởi động file vừa tải về, sau đó ấn "**Next**"



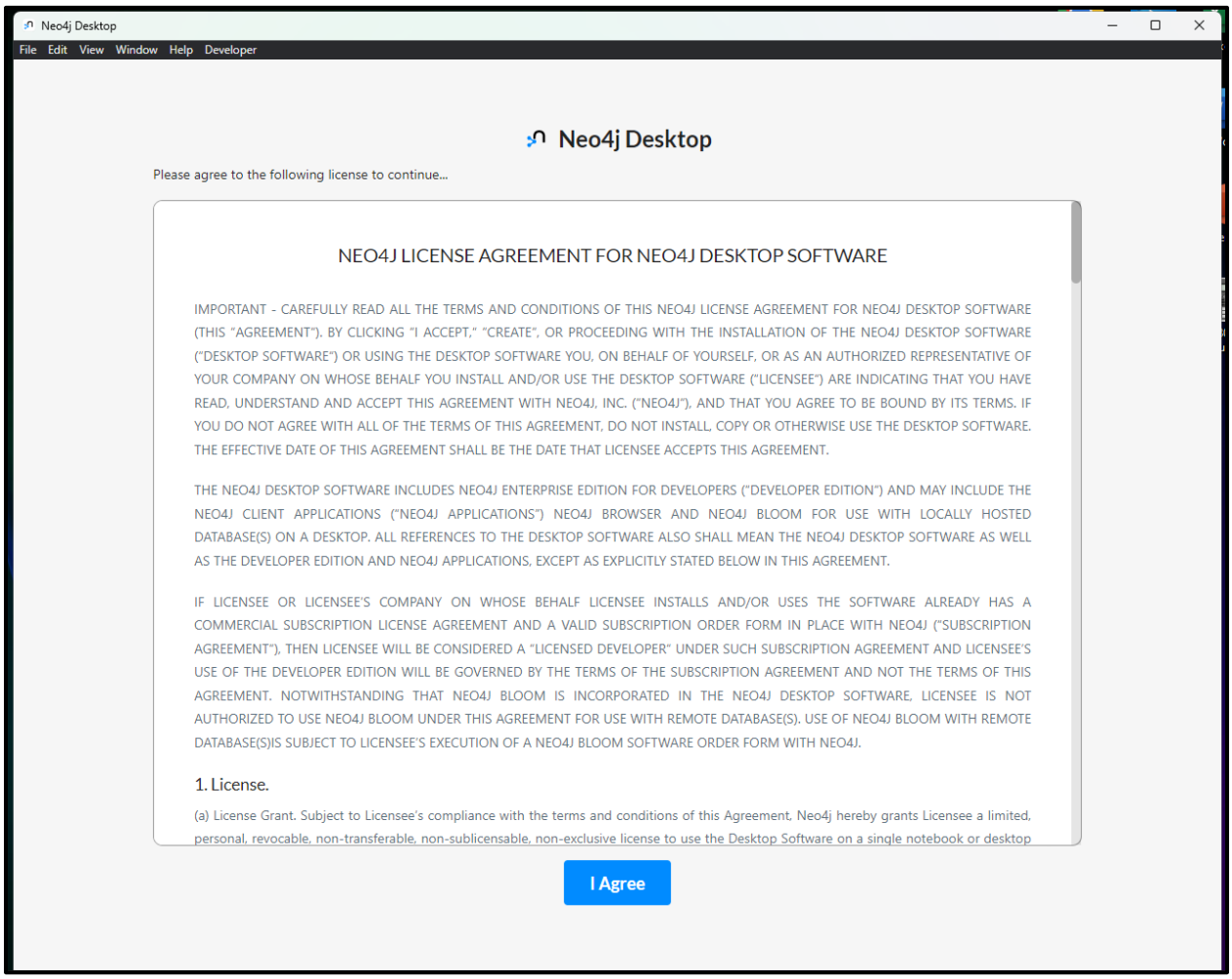
➢ Chọn đường dẫn và nhấn "**Install**".



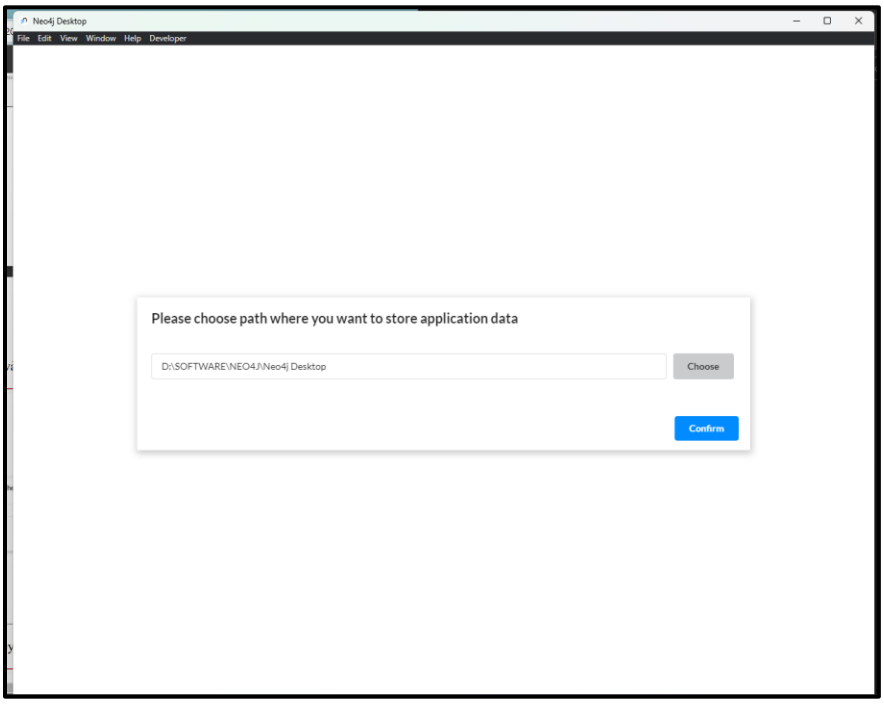
➢ Nhấn "**Finish**"



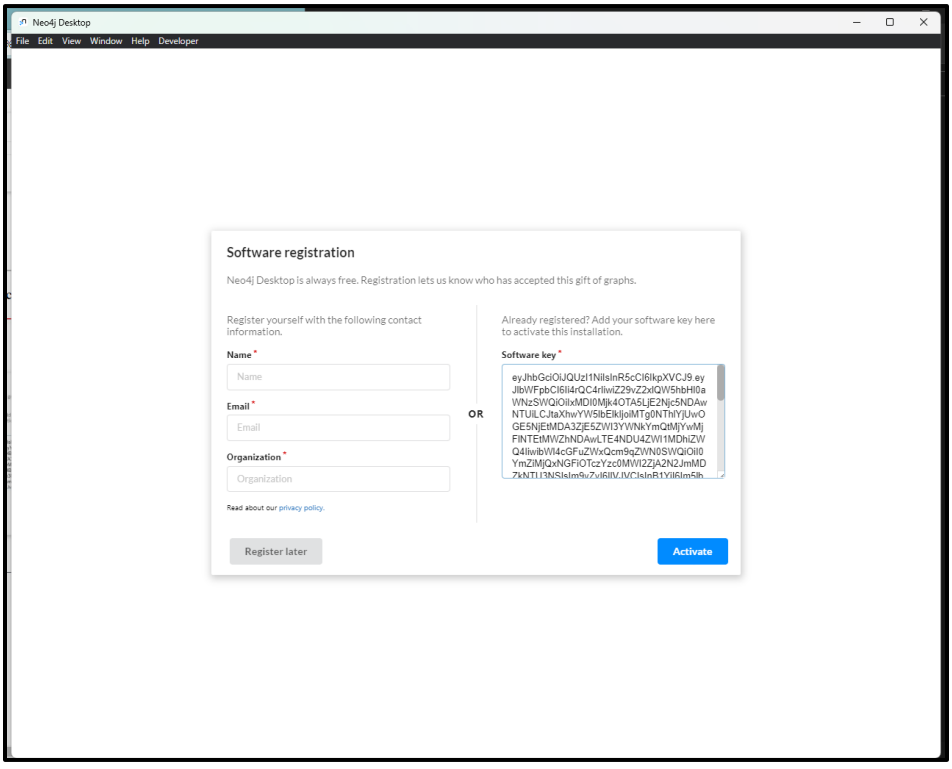
➢ Nhấn "**I Agree**"

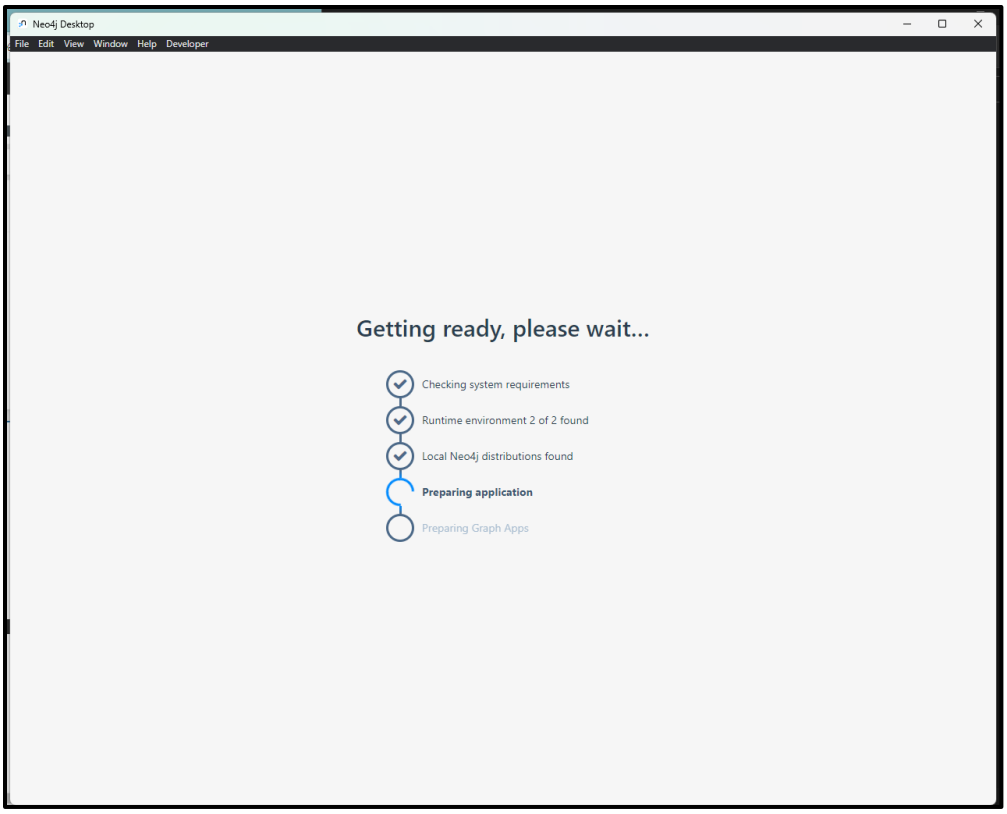

Neo4j Desktop
File Edit View Window Help Developer
Neo4j Desktop
Please agree to the following license to continue...
NEO4J LICENSE AGREEMENT FOR NEO4J DESKTOP SOFTWARE
IMPORTANT - CAREFULLY READ ALL THE TERMS AND CONDITIONS OF THIS NEO4J LICENSE AGREEMENT FOR NEO4J DESKTOP SOFTWARE (THIS "AGREEMENT"). BY CLICKING "I ACCEPT," "CREATE", OR PROCEEDING WITH THE INSTALLATION OF THE NEO4J DESKTOP SOFTWARE ("DESKTOP SOFTWARE") OR USING THE DESKTOP SOFTWARE YOU, ON BEHALF OF YOURSELF, OR AS AN AUTHORIZED REPRESENTATIVE OF YOUR COMPANY ON WHOSE BEHALF YOU INSTALL AND/OR USE THE DESKTOP SOFTWARE ("LICENSEE") ARE INDICATING THAT YOU HAVE READ, UNDERSTAND AND ACCEPT THIS AGREEMENT WITH NEO4J, INC. ("NEO4J"), AND THAT YOU AGREE TO BE BOUND BY ITS TERMS. IF YOU DO NOT AGREE WITH ALL OF THE TERMS OF THIS AGREEMENT, DO NOT INSTALL, COPY OR OTHERWISE USE THE DESKTOP SOFTWARE. THE EFFECTIVE DATE OF THIS AGREEMENT SHALL BE THE DATE THAT LICENSEE ACCEPTS THIS AGREEMENT.
THE NEO4J DESKTOP SOFTWARE INCLUDES NEO4J ENTERPRISE EDITION FOR DEVELOPERS ("DEVELOPER EDITION") AND MAY INCLUDE THE NEO4J CLIENT APPLICATIONS ("NEO4J APPLICATIONS") NEO4J BROWSER AND NEO4J BLOOM FOR USE WITH LOCALLY HOSTED DATABASE(S) ON A DESKTOP. ALL REFERENCES TO THE DESKTOP SOFTWARE ALSO SHALL MEAN THE NEO4J DESKTOP SOFTWARE AS WELL AS THE DEVELOPER EDITION AND NEO4J APPLICATIONS, EXCEPT AS EXPLICITLY STATED BELOW IN THIS AGREEMENT.
IF LICENSEE OR LICENSEE'S COMPANY ON WHOSE BEHALF LICENSEE INSTALLS AND/OR USES THE SOFTWARE ALREADY HAS A COMMERCIAL SUBSCRIPTION LICENSE AGREEMENT AND A VALID SUBSCRIPTION ORDER FORM IN PLACE WITH NEO4J ("SUBSCRIPTION AGREEMENT"), THEN LICENSEE WILL BE CONSIDERED A "LICENSED DEVELOPER" UNDER SUCH SUBSCRIPTION AGREEMENT AND LICENSEE'S USE OF THE DEVELOPER EDITION WILL BE GOVERNED BY THE TERMS OF THE SUBSCRIPTION AGREEMENT AND NOT THE TERMS OF THIS AGREEMENT. NOTWITHSTANDING THAT NEO4J BLOOM IS INCORPORATED IN THE NEO4J DESKTOP SOFTWARE, LICENSEE IS NOT AUTHORIZED TO USE NEO4J BLOOM UNDER THIS AGREEMENT FOR USE WITH REMOTE DATABASE(S). USE OF NEO4J BLOOM WITH REMOTE DATABASE(S)IS SUBJECT TO LICENSEE'S EXECUTION OF A NEO4J BLOOM SOFTWARE ORDER FORM WITH NEO4J.
1. License.
(a) License Grant. Subject to Licensee's compliance with the terms and conditions of this Agreement, Neo4j hereby grants Licensee a limited, personal, revocable, non-transferable, non-sublicensable, non-exclusive license to use the Desktop Software on a single notebook or desktop
I Agree

➢ Chọn đường dẫn và ấn "**Confirm**"

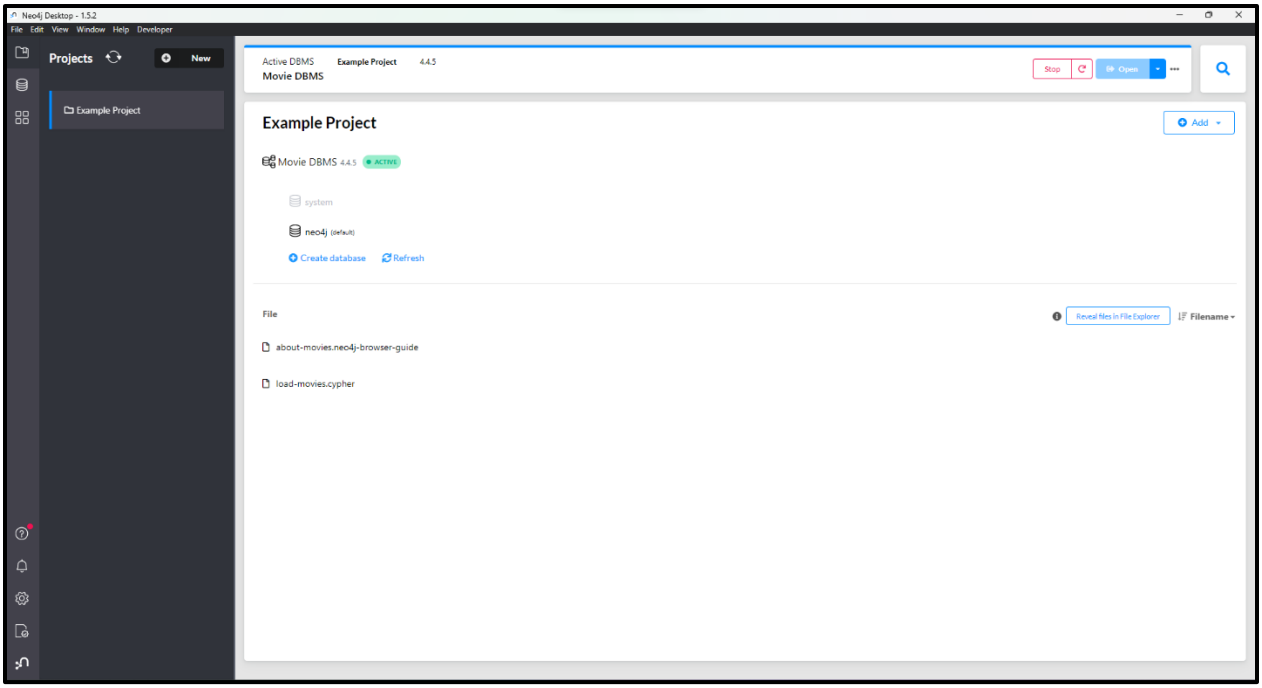


➢ Nhập Solfware key đã có ở phía trên vào ô cùng tên rồi ấn "**Activate**"



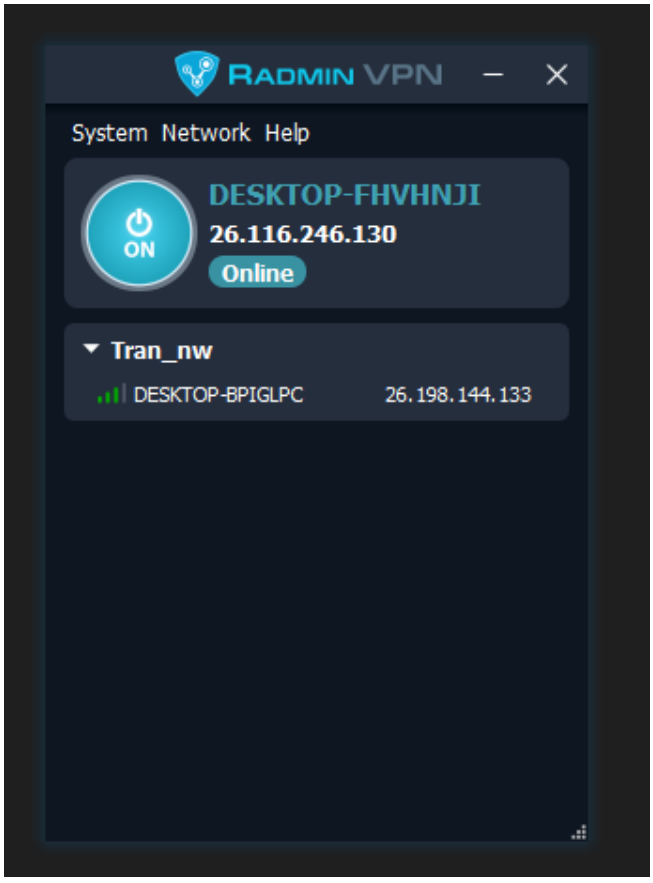


➢ Cài đặt thành công



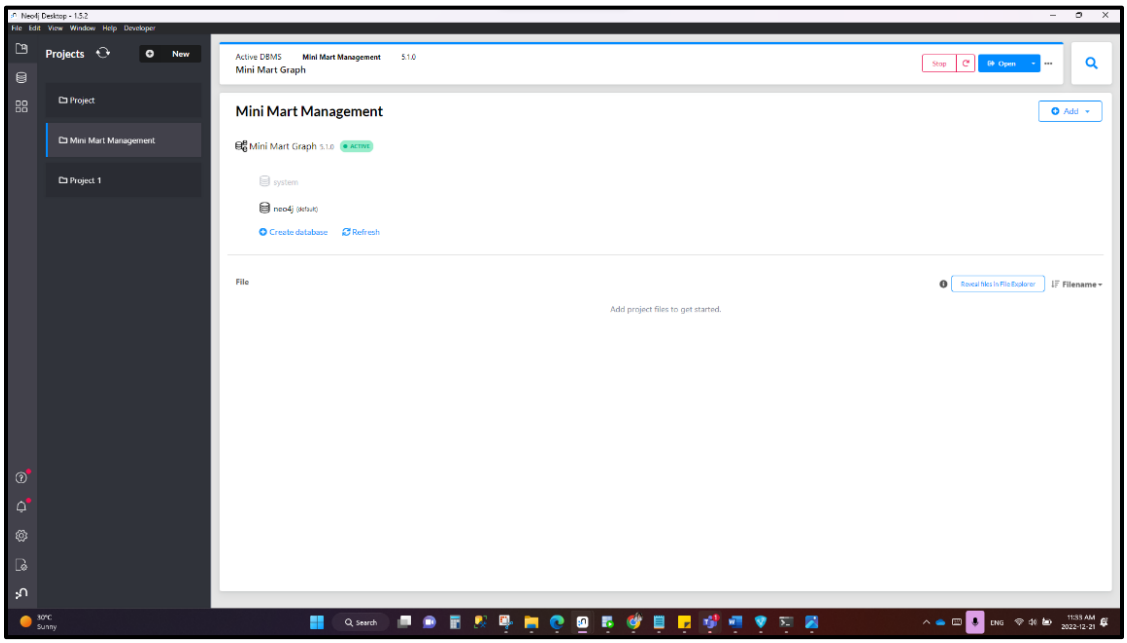
2. Config dữ liệu

❖ Kết nối VPN bằng radmin

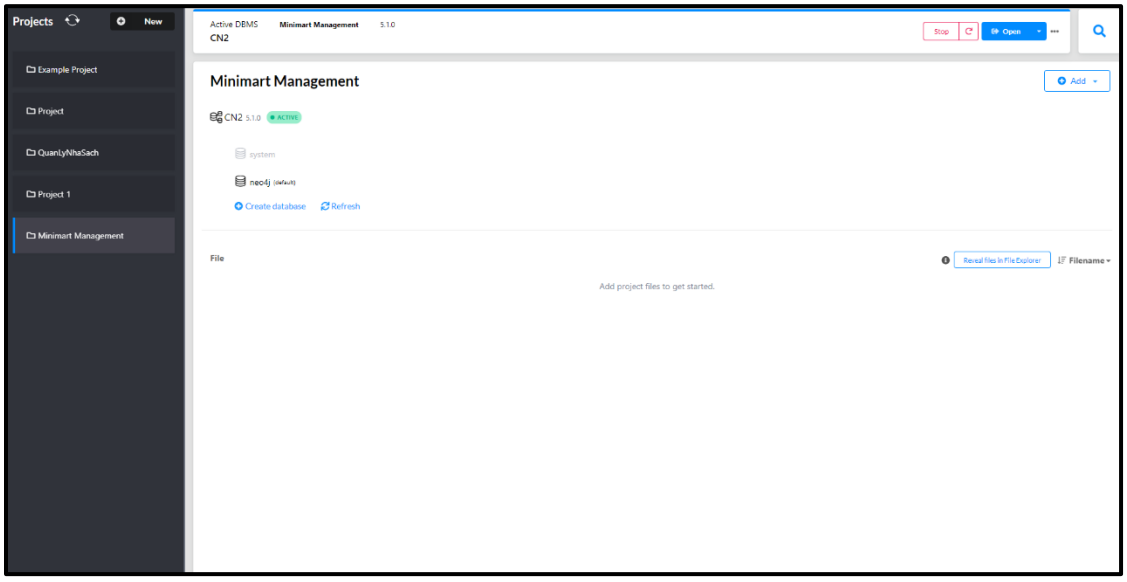


❖ Tạo project ở hai máy

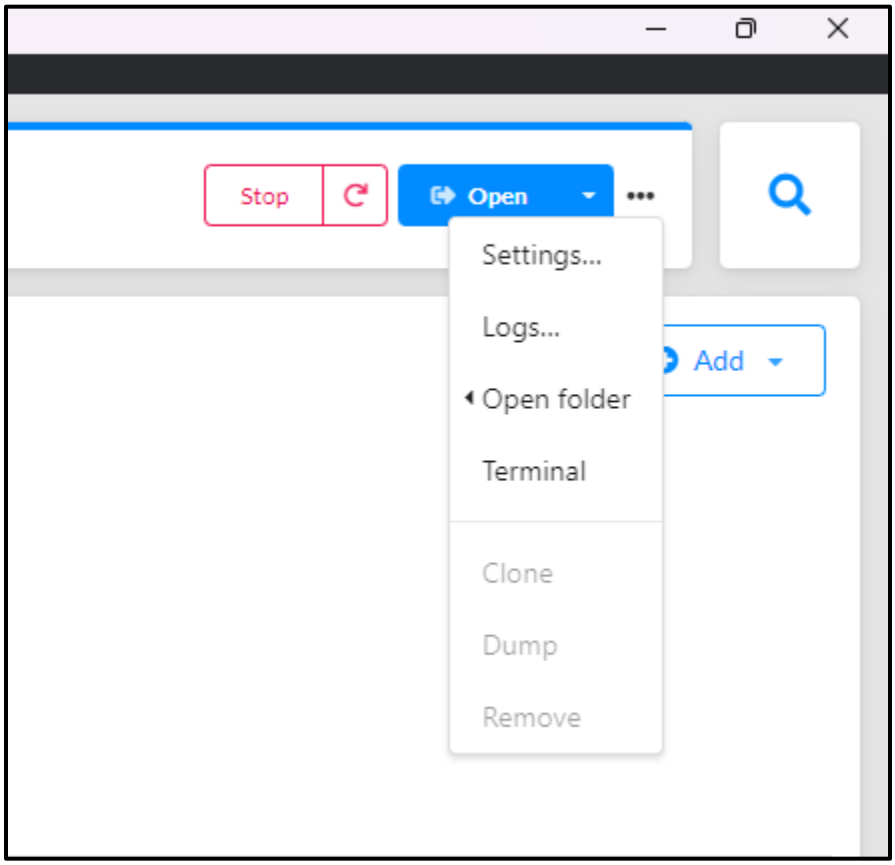
- CN1:



- CN2:



❖ Nhấn dấu 3 chấm bên cạnh tên nút open, chọn settings



Tìm dòng server.default_listen_address=0.0.0.0 rồi uncomment

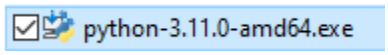
➢ Nhấn "**Apply**"

❖ Ở chi nhánh 2 làm tương tự

3. Cài python để tiến hành config data

❖ Vào trang chủ của python theo đường dẫn https://www.python.org/downloads/ sau đó rê chuột vào download chọn download python bản mới nhất

❖ Chạy file vừa tải về rồi làm theo hướng dẫn của python để cài môi trường



❖ Vào cmd gõ python, nếu ra kết quả như ảnh bên dưới tức là đã cài môi trường python thành công

```
C:\Users\tonnu>python
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

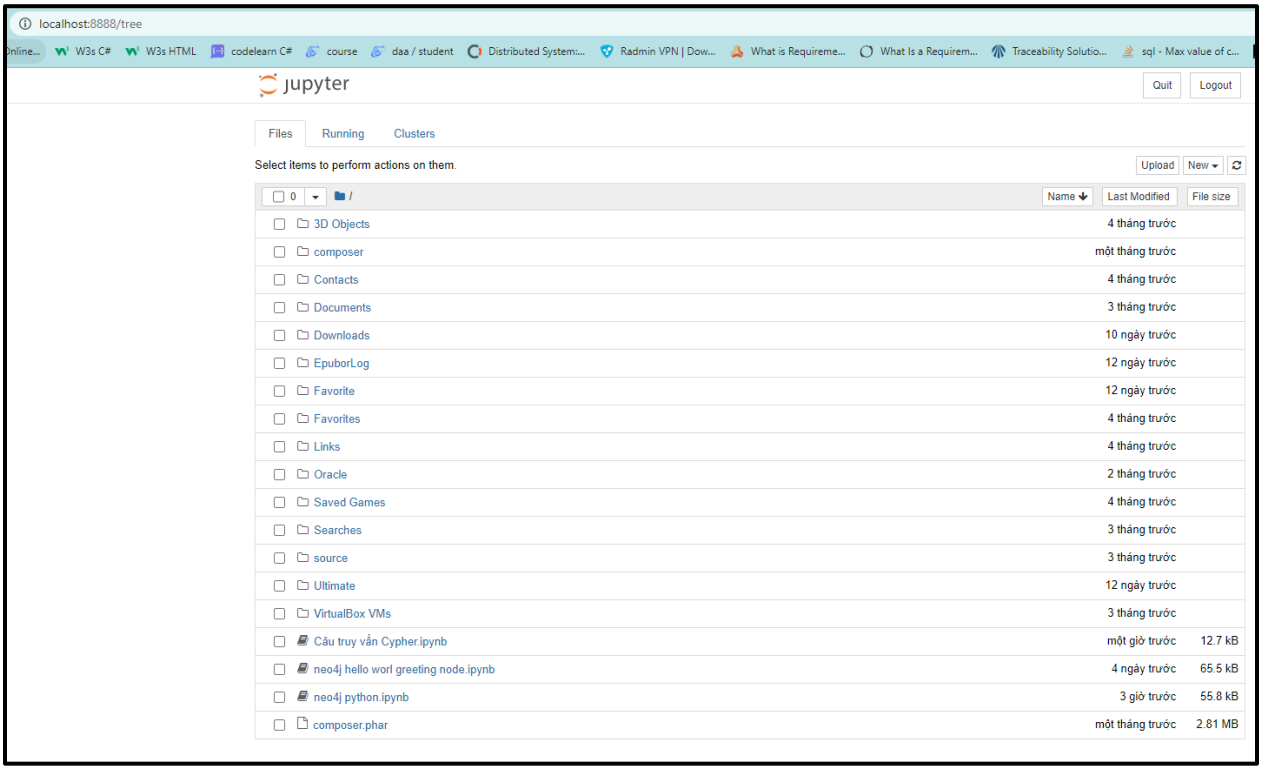
❖ Tiếp theo, vào cmd chạy lần lượt các dòng lệnh sau:

- "pip install py2neo" rồi chờ thư viện py2neo được cài thành công
- "pip install pandas" rồi chờ thư viện pandas được cài thành công
- "pip install notebook" rồi chờ jupyter noteebook được cài thành công

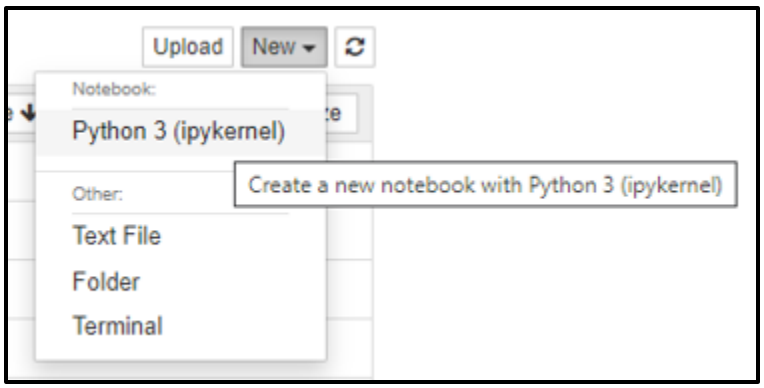
❖ Sau đó, chạy dòng lệnh "jupyter notebook" trong cmd để khởi động jupyter notebook trên trình duyệt web

```
C:\Users\tonnu>jupyter notebook
[I 11:49:04.234 NotebookApp] Serving notebooks from local directory: C:\Users\tonnu
[I 11:49:04.234 NotebookApp] Jupyter Notebook 6.5.2 is running at:
[I 11:49:04.234 NotebookApp] http://localhost:8888/?token=ae63890d1d2815256912d35b02f51391d1dd093dfd7b115c
[I 11:49:04.234 NotebookApp]  or http://127.0.0.1:8888/?token=ae63890d1d2815256912d35b02f51391d1dd093dfd7b115c
[I 11:49:04.234 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 11:49:04.303 NotebookApp]

    To access the notebook, open this file in a browser:
        file:///C:/Users/tonnu/AppData/Roaming/jupyter/runtime/nbserver-18384-open.html
    Or copy and paste one of these URLs:
        http://localhost:8888/?token=ae63890d1d2815256912d35b02f51391d1dd093dfd7b115c
     or http://127.0.0.1:8888/?token=ae63890d1d2815256912d35b02f51391d1dd093dfd7b115c
0.00s - Debugger warning: It seems that frozen modules are being used, which may
0.00s - make the debugger miss breakpoints. Please pass -Xfrozen_modules=off
0.00s - to python to disable frozen modules.
0.00s - Note: Debugging will proceed. Set PYDEVD_DISABLE_FILE_VALIDATION=1 to disable this validation.
```

❖ Trình duyệt web sẽ tự động mở ra trang jupyter notebook như sau



❖ Nhấn vào "New" chọn "Python 3" để tạo project mới

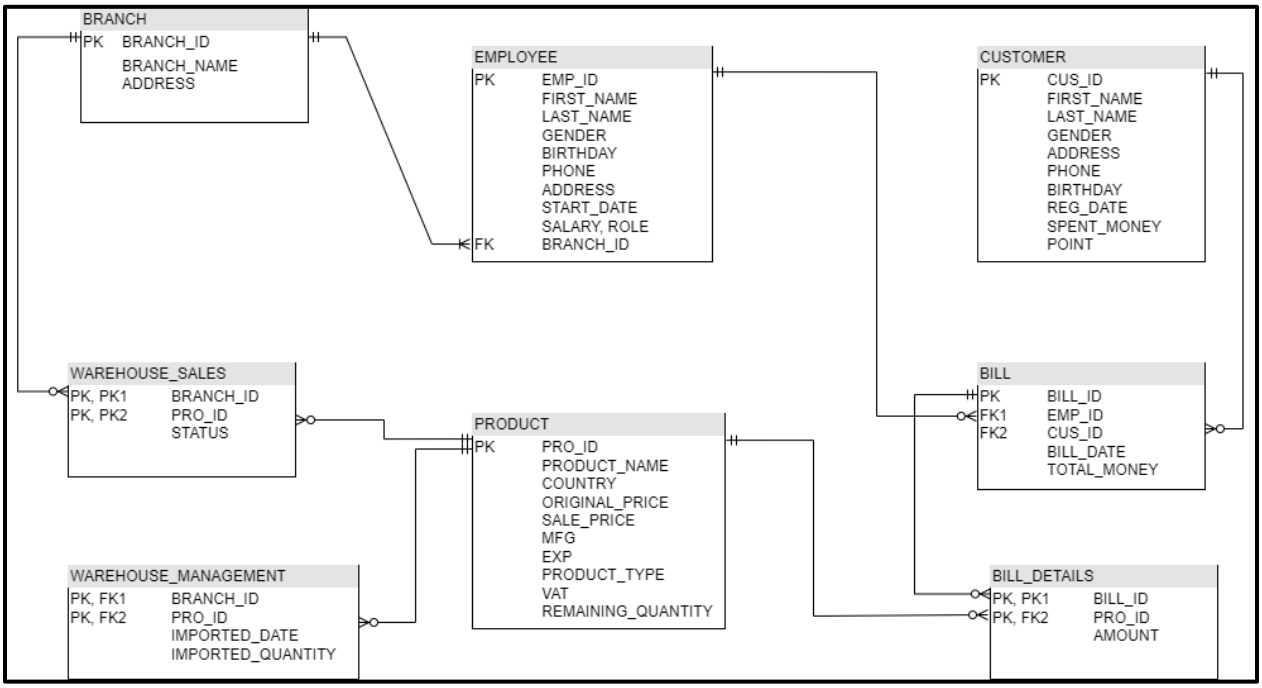


- Lưu ý, database cần phải được open ở neo4j của cả 2 máy để kết nối thành công

```
In [ ]: #import các thư viện sử dụng để phân tán
        from py2neo import Graph
        import pandas as pd
        import numpy as np

        #Kết nối tới database của chi nhánh 1 sử dụng VPN ảo của Radmin
        cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))

        #Kết nối tới database của chi nhánh 2 sử dụng VPN ảo của Radmin
        cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))

        """ở đây 'neo4j' là tên user, '123456' là mật khẩu"""

        #Các câu truy vấn hoặc dml đều sử dụng trên các biến cn1 và cn2 ở trên
```

D. THAO TÁC QUA LẠI GIỮA HAI MÁY

1. Thêm dữ liệu:



- TẠO LABEL VÀ THÊM NOTE VÀO LABEL:

	CN1	CN2
BRANCH	CREATE (B:BRANCH { BRANCH_ID: "CN1", BRANCH_NAME: "Mini mart Quan 9", ADDRESS: "5, Le Van Viet, Quan 9, TPHCM" });	CREATE (B:BRANCH { BRANCH_ID: "CN2", BRANCH_NAME: "Mini mart chi nhanh Quan 1", ADDRESS: "55, Ly Thanh Tong, Quan 1, TPHCM" });
EMPLOYEE	CREATE (E:EMPLOYEE { EMP_ID: 200001, FIRST_NAME: "Nguyen Thi My", LAST_NAME: "Tran", GENDER: "Female", BIRTHDAY: DATE("2002-05-05"), PHONE: "0921231741", ADDRESS: "KTX khu B", START_DATE: DATE("2022-02-21"), SALARY: 10000000, ROLE: "Manager", BRANCH_ID: "CN1" });	CREATE (E:EMPLOYEE { EMP_ID: 600001, FIRST_NAME: "Nguyen Thi My", LAST_NAME: "Tam", GENDER: "Female", BIRTHDAY: DATE("2002-05-05"), PHONE: "921231741", ADDRESS: "KTX khu B", START_DATE: DATE("2022-02-21"), SALARY: 10000000, ROLE: "Manager", BRANCH_ID: "CN2" }); CREATE (E:EMPLOYEE

```
CREATE (E:EMPLOYEE
{
   EMP_ID: 200002,
   FIRST_NAME: "Ton Nu Tu",
   LAST_NAME: "Quyen",
   GENDER: "Female",
   BIRTHDAY: DATE("2002-01-05"),
   PHONE: "977164279",
   ADDRESS: "KTX khu A",
   START_DATE: DATE("2022-02-21"),
   SALARY: 5000000,
   ROLE: "Cashier",
   BRANCH_ID: "CN1"

});

CREATE (E:EMPLOYEE
{
   EMP_ID: 200003,
   FIRST_NAME: "Ngo Thi",
   LAST_NAME: "Anh",
   GENDER: "Female",
   BIRTHDAY: DATE("2002-08-25"),
   PHONE: "356315873",
   ADDRESS: "731 Tran Hung Dao, Q5,
TpHCM",
   START_DATE: DATE("2022-02-21"),
   SALARY: 5000000,
   ROLE: "Cashier",
   BRANCH_ID: "CN1"
});

CREATE (E:EMPLOYEE
{
   EMP_ID: 200004,
   FIRST_NAME: "Nguyen Anh",
   LAST_NAME: "Viet",
   GENDER: "Male",
   BIRTHDAY: DATE("2002-07-05"),
   PHONE: "585784600",
   ADDRESS: "45 Nguyen Canh Chan, Q1,
TPHCM",
   START_DATE: DATE("2022-02-21"),
   SALARY: 5000000,
   ROLE: "Storekeeper",
   BRANCH_ID: "CN1"
});

CREATE (E:EMPLOYEE
{
   EMP_ID: 200005,
   FIRST_NAME: "Nguyen Thi Minh",
   LAST_NAME: "Triet",
   GENDER: "Female",
   BIRTHDAY: DATE("2002-09-05"),
```

```
{
   EMP_ID: 600002,
   FIRST_NAME: "Ton Nu Tu",
   LAST_NAME: "Tran",
   GENDER: "Female",
   BIRTHDAY: DATE("2002-01-05"),
   PHONE: "977164279",
   ADDRESS: "KTX khu A",
   START_DATE: DATE("2022-02-21"),
   SALARY: 5000000,
   ROLE: "Cashier",
   BRANCH_ID: "CN2"
});

CREATE (E:EMPLOYEE
{
   EMP_ID: 600003,
   FIRST_NAME: "Ngo Thi",
   LAST_NAME: "Bap",
   GENDER: "Female",
   BIRTHDAY: DATE("2002-08-25"),
   PHONE: "356315873",
   ADDRESS: "731 Tran Hung Dao, Q5,
TpHCM",
   START_DATE: DATE("2022-02-21"),
   SALARY: 5000000,
   ROLE: "Cashier",
   BRANCH_ID: "CN2"
});

CREATE (E:EMPLOYEE
{
   EMP_ID: 600004,
   FIRST_NAME: "Nguyen Anh",
   LAST_NAME: "Kiet",
   GENDER: "Male",
   BIRTHDAY: DATE("2002-07-05"),
   PHONE: "585784600",
   ADDRESS: "45 Nguyen Canh Chan, Q1,
TPHCM",
   START_DATE: DATE("2022-02-21"),
   SALARY: 5000000,
   ROLE: "Storekeeper",
   BRANCH_ID: "CN2"
});

CREATE (E:EMPLOYEE
{
   EMP_ID: 600005,
   FIRST_NAME: "Nguyen Thi Minh",
   LAST_NAME: "Huong",
   GENDER: "Female",
   BIRTHDAY: DATE("2002-02-04"),
   PHONE: "932342312",
   ADDRESS: "50/34 Le Dai Hanh, Q10,
TPHCM",
```

	PHONE: "932342312", ADDRESS: "50/34 Le Dai Hanh, Q10, TPHCM", START_DATE: DATE("2022-02-21"), SALARY: 5000000, ROLE: "Salesperson", BRANCH_ID: "CN1" }); CREATE (E:EMPLOYEE { EMP_ID: 200006, FIRST_NAME: "Tran Anh", LAST_NAME: "Huy", GENDER: "Male", BIRTHDAY: DATE("2002-07-15"), PHONE: "916430073", ADDRESS: "KTX khu B", START_DATE: DATE("2022-02-21"), SALARY: 7000000, ROLE: "Salesperson", BRANCH_ID: "CN1" }); CREATE (E:EMPLOYEE { EMP_ID: 200007, FIRST_NAME: "Phan Hong", LAST_NAME: "Anh", GENDER: "Female", BIRTHDAY: DATE("2001-08-20"), PHONE: "944562311", ADDRESS: "16 Le Van Viet, Q9, TPHCM", START_DATE: DATE("2022-02-21"), SALARY: 7000000, ROLE: "Customer Service Assistant", BRANCH_ID: "CN1" }); CREATE (E:EMPLOYEE { EMP_ID: 200008, FIRST_NAME: "Thai Tang", LAST_NAME: "Duc", GENDER: "Male", BIRTHDAY: DATE("2002-10-01"), PHONE: "931231756", ADDRESS: "KTX khu A", START_DATE: DATE("2022-02-21"), SALARY: 5000000, ROLE: "Quality Checker", BRANCH_ID: "CN1" }); CREATE (E:EMPLOYEE	START_DATE: DATE("2022-02-21"), SALARY: 5000000, ROLE: "Salesperson", BRANCH_ID: "CN2" }); CREATE (E:EMPLOYEE { EMP_ID: 600006, FIRST_NAME: "Tran Anh", LAST_NAME: "Tuan", GENDER: "Male", BIRTHDAY: DATE("2002-08-05"), PHONE: "916430073", ADDRESS: "KTX khu B", START_DATE: DATE("2022-02-21"), SALARY: 7000000, ROLE: "Salesperson", BRANCH_ID: "CN2" }); CREATE (E:EMPLOYEE { EMP_ID: 600007, FIRST_NAME: "Phan Hong", LAST_NAME: "Duc", GENDER: "Female", BIRTHDAY: DATE("2001-08-20"), PHONE: "944562311", ADDRESS: "16 Le Van Viet, Q9, TPHCM", START_DATE: DATE("2022-02-21"), SALARY: 7000000, ROLE: "Customer Service Assistant", BRANCH_ID: "CN2" }); CREATE (E:EMPLOYEE { EMP_ID: 600008, FIRST_NAME: "Thai Tang", LAST_NAME: "Minh", GENDER: "Male", BIRTHDAY: DATE("2002-10-16"), PHONE: "931231756", ADDRESS: "KTX khu A", START_DATE: DATE("2022-02-21"), SALARY: 5000000, ROLE: "Quality Checker", BRANCH_ID: "CN2" }); CREATE (E:EMPLOYEE { EMP_ID: 600009, FIRST_NAME: "Le Nguyen Thuy",

	{ EMP_ID: 200009, FIRST_NAME: "Le Nguyen Thuy", LAST_NAME: "Vi", GENDER: "Female", BIRTHDAY: DATE("2001-06-06"), PHONE: "948965678", ADDRESS: "KTX khu B", START_DATE: DATE("2022-02-21"), SALARY: 5000000, ROLE: "Sanitation Worker", BRANCH_ID: "CN1" }); CREATE (E:EMPLOYEE { EMP_ID: 200010, FIRST_NAME: "Nguyen Thi Kim", LAST_NAME: "Hieu", GENDER: "Female", BIRTHDAY: DATE("2002-08-28"), PHONE: "931231756", ADDRESS: "KTX khu B", START_DATE: DATE("2022-02-21"), SALARY: 5000000, ROLE: "Security Guard", BRANCH_ID: "CN1" });	LAST_NAME: "Lieu", GENDER: "Female", BIRTHDAY: DATE("2001-06-06"), PHONE: "948965678", ADDRESS: "KTX khu B", START_DATE: DATE("2022-02-21"), SALARY: 5000000, ROLE: "Sanitation Worker", BRANCH_ID: "CN2" }); CREATE (E:EMPLOYEE { EMP_ID: 600010, FIRST_NAME: "Nguyen Thi Kim", LAST_NAME: "Ngan", GENDER: "Female", BIRTHDAY: DATE("2002-08-28"), PHONE: "931231756", ADDRESS: "KTX khu B", START_DATE: DATE("2022-02-21"), SALARY: 5000000, ROLE: "Security Guard", BRANCH_ID: "CN2" });
CUSTOMER	CREATE (C:CUSTOMER { CUS_ID: 300001, FIRST_NAME: "Tran Minh", LAST_NAME: "Tien", GENDER: "Male", ADDRESS: "117/2 Nguyen Trai, Q5, TpHCM", PHONE: "883644231", BIRTHDAY: DATE("2002-04-15"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 279072, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300002, FIRST_NAME: "Nguyen Bang", LAST_NAME: "Huu", GENDER: "Male", ADDRESS: "731 Tran Hung Dao, Q5, TpHCM", PHONE: "975244479", BIRTHDAY: DATE("2001-07-02"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 26163, POINT: 0	CREATE (C:CUSTOMER { CUS_ID: 300001, FIRST_NAME: "Tran Minh", LAST_NAME: "Tien", GENDER: "Male", ADDRESS: "117/2 Nguyen Trai, Q5, TpHCM", PHONE: "883644231", BIRTHDAY: DATE("2002-04-15"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 279072, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300002, FIRST_NAME: "Nguyen Bang", LAST_NAME: "Huu", GENDER: "Male", ADDRESS: "731 Tran Hung Dao, Q5, TpHCM", PHONE: "975244479", BIRTHDAY: DATE("2001-07-02"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 26163, POINT: 0

	});	});
	CREATE (C:CUSTOMER { CUS_ID: 300003, FIRST_NAME: "Nguyen Huu", LAST_NAME: "Tho", GENDER: "Male", ADDRESS: "23/5 Nguyen Trai, Q5, TpHCM", PHONE: "361234578", BIRTHDAY: DATE("2001-06-18"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 98838, POINT: 0 });	CREATE (C:CUSTOMER { CUS_ID: 300003, FIRST_NAME: "Nguyen Huu", LAST_NAME: "Tho", GENDER: "Male", ADDRESS: "23/5 Nguyen Trai, Q5, TpHCM", PHONE: "361234578", BIRTHDAY: DATE("2001-06-18"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 98838, POINT: 0 });
	CREATE (C:CUSTOMER { CUS_ID: 300004, FIRST_NAME: "Nguyen Phuong", LAST_NAME: "Thanh", GENDER: "Female", ADDRESS: "27/2 Nguyen Trai, Q5, TpHCM", PHONE: "365238774", BIRTHDAY: DATE("1999-06-24"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 96900, POINT: 0 });	CREATE (C:CUSTOMER { CUS_ID: 300004, FIRST_NAME: "Nguyen Phuong", LAST_NAME: "Thanh", GENDER: "Female", ADDRESS: "27/2 Nguyen Trai, Q5, TpHCM", PHONE: "365238774", BIRTHDAY: DATE("1999-06-24"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 96900, POINT: 0 });
	CREATE (C:CUSTOMER { CUS_ID: 300005, FIRST_NAME: "Le Thi Tuong", LAST_NAME: "Vi", GENDER: "Female", ADDRESS: "45 Nguyen Canh Chan, Q1, TPHCM", PHONE: "938776266", BIRTHDAY: DATE("2002-02-22"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 222870, POINT: 1 });	CREATE (C:CUSTOMER { CUS_ID: 300005, FIRST_NAME: "Le Thi Tuong", LAST_NAME: "Vi", GENDER: "Female", ADDRESS: "45 Nguyen Canh Chan, Q1, TPHCM", PHONE: "938776266", BIRTHDAY: DATE("2002-02-22"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 222870, POINT: 1 });
	CREATE (C:CUSTOMER { CUS_ID: 300006, FIRST_NAME: "Nguyen Thanh", LAST_NAME: "Duy", GENDER: "Male", ADDRESS: "50/34 Le Dai Hanh, Q10, TPHCM", PHONE: "938826866",	CREATE (C:CUSTOMER { CUS_ID: 300006, FIRST_NAME: "Nguyen Thanh", LAST_NAME: "Duy", GENDER: "Male", ADDRESS: "50/34 Le Dai Hanh, Q10, TPHCM", PHONE: "938826866",

	BIRTHDAY: DATE("2001-01-18"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 38760, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300007, FIRST_NAME: "Vo Minh", LAST_NAME: "Trong", GENDER: "Male", ADDRESS: "837 Le Hong Phong,Q5,TPHCM", PHONE: "937825255", BIRTHDAY: DATE("2001-10-14"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 14535, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300008, FIRST_NAME: "Dinh Van", LAST_NAME: "Quyet", GENDER: "Male", ADDRESS: "50/34 Le Dai Hanh, Q10, TPHCM", PHONE: "837822285", BIRTHDAY: DATE("2002-04-16"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 7752, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300009, FIRST_NAME: "Ho Nguyen Thu", LAST_NAME: "An", GENDER: "Female", ADDRESS: "34/34B Nguyen Trai, Q5, TPHCM", PHONE: "237825224", BIRTHDAY: DATE("2001-11-09"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 96900, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300010, FIRST_NAME: "Nguyen Anh", LAST_NAME: "Phi",	BIRTHDAY: DATE("2001-01-18"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 38760, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300007, FIRST_NAME: "Vo Minh", LAST_NAME: "Trong", GENDER: "Male", ADDRESS: "837 Le Hong Phong,Q5,TPHCM", PHONE: "937825255", BIRTHDAY: DATE("2001-10-14"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 14535, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300008, FIRST_NAME: "Dinh Van", LAST_NAME: "Quyet", GENDER: "Male", ADDRESS: "50/34 Le Dai Hanh, Q10, TPHCM", PHONE: "837822285", BIRTHDAY: DATE("2002-04-16"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 7752, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300009, FIRST_NAME: "Ho Nguyen Thu", LAST_NAME: "An", GENDER: "Female", ADDRESS: "34/34B Nguyen Trai, Q5, TPHCM", PHONE: "237825224", BIRTHDAY: DATE("2001-11-09"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 96900, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300010, FIRST_NAME: "Nguyen Anh", LAST_NAME: "Phi",

	GENDER: "Male", ADDRESS: "227 Nguyen Van Cu, Q5, TPHCM", PHONE: "937885255", BIRTHDAY: DATE("2002-12-06"), REG_DATE: DATE("2022-09-04"), SPENT_MONEY: 145350, POINT: 1 }); CREATE (C:CUSTOMER { CUS_ID: 300011, FIRST_NAME: "Tran Minh", LAST_NAME: "Minh", GENDER: "Male", ADDRESS: "117/2 Nguyen Trai, Q5, TpHCM", PHONE: "883644231", BIRTHDAY: DATE("2002-04-15"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 279072, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300012, FIRST_NAME: "Nguyen Bang", LAST_NAME: "Ha", GENDER: "Male", ADDRESS: "731 Tran Hung Dao, Q5, TpHCM", PHONE: "975244479", BIRTHDAY: DATE("2002-04-11"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 26163, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300013, FIRST_NAME: "Nguyen Huu", LAST_NAME: "Tham", GENDER: "Male", ADDRESS: "23/5 Nguyen Trai, Q5, TpHCM", PHONE: "361234578", BIRTHDAY: DATE("2001-06-18"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 98838, POINT: 0 }); CREATE (C:CUSTOMER	GENDER: "Male", ADDRESS: "227 Nguyen Van Cu, Q5, TPHCM", PHONE: "937885255", BIRTHDAY: DATE("2002-12-06"), REG_DATE: DATE("2022-09-04"), SPENT_MONEY: 145350, POINT: 1 }); CREATE (C:CUSTOMER { CUS_ID: 300011, FIRST_NAME: "Tran Minh", LAST_NAME: "Minh", GENDER: "Male", ADDRESS: "117/2 Nguyen Trai, Q5, TpHCM", PHONE: "883644231", BIRTHDAY: DATE("2002-04-15"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 279072, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300012, FIRST_NAME: "Nguyen Bang", LAST_NAME: "Ha", GENDER: "Male", ADDRESS: "731 Tran Hung Dao, Q5, TpHCM", PHONE: "975244479", BIRTHDAY: DATE("2002-04-11"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 26163, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300013, FIRST_NAME: "Nguyen Huu", LAST_NAME: "Tham", GENDER: "Male", ADDRESS: "23/5 Nguyen Trai, Q5, TpHCM", PHONE: "361234578", BIRTHDAY: DATE("2001-06-18"), REG_DATE: DATE("2022-09-01"), SPENT_MONEY: 98838, POINT: 0 }); CREATE (C:CUSTOMER

	{ CUS_ID: 300014, FIRST_NAME: "Nguyen Phuong", LAST_NAME: "Thao", GENDER: "Female", ADDRESS: "27/2 Nguyen Trai, Q5, TpHCM", PHONE: "365238774", BIRTHDAY: DATE("1999-06-24"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 96900, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300015, FIRST_NAME: "Le Thi Tuong", LAST_NAME: "An", GENDER: "Female", ADDRESS: "45 Nguyen Canh Chan, Q1, TPHCM", PHONE: "938776266", BIRTHDAY: DATE("2002-06-10"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 222870, POINT: 1 }); CREATE (C:CUSTOMER { CUS_ID: 300016, FIRST_NAME: "Nguyen Thanh", LAST_NAME: "Khiet", GENDER: "Male", ADDRESS: "50/34 Le Dai Hanh, Q10, TPHCM", PHONE: "938826866", BIRTHDAY: DATE("2001-01-18"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 38760, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300017, FIRST_NAME: "Vo Minh", LAST_NAME: "Tam", GENDER: "Male", ADDRESS: "837 Le Hong Phong,Q5,TPHCM", PHONE: "937825255", BIRTHDAY: DATE("2001-10-14"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 14535,	{ CUS_ID: 300014, FIRST_NAME: "Nguyen Phuong", LAST_NAME: "Thao", GENDER: "Female", ADDRESS: "27/2 Nguyen Trai, Q5, TpHCM", PHONE: "365238774", BIRTHDAY: DATE("1999-06-24"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 96900, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300015, FIRST_NAME: "Le Thi Tuong", LAST_NAME: "An", GENDER: "Female", ADDRESS: "45 Nguyen Canh Chan, Q1, TPHCM", PHONE: "938776266", BIRTHDAY: DATE("2002-06-10"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 222870, POINT: 1 }); CREATE (C:CUSTOMER { CUS_ID: 300016, FIRST_NAME: "Nguyen Thanh", LAST_NAME: "Khiet", GENDER: "Male", ADDRESS: "50/34 Le Dai Hanh, Q10, TPHCM", PHONE: "938826866", BIRTHDAY: DATE("2001-01-18"), REG_DATE: DATE("2022-09-02"), SPENT_MONEY: 38760, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300017, FIRST_NAME: "Vo Minh", LAST_NAME: "Tam", GENDER: "Male", ADDRESS: "837 Le Hong Phong,Q5,TPHCM", PHONE: "937825255", BIRTHDAY: DATE("2001-10-14"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 14535,

	POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300018, FIRST_NAME: "Dinh Van", LAST_NAME: "Toan", GENDER: "Male", ADDRESS: "50/34 Le Dai Hanh, Q10, TPHCM", PHONE: "837822285", BIRTHDAY: DATE("2002-04-16"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 7752, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300019, FIRST_NAME: "Ho Nguyen Thu", LAST_NAME: "Minh", GENDER: "Female", ADDRESS: "34/34B Nguyen Trai, Q5, TPHCM", PHONE: "237825224", BIRTHDAY: DATE("2003-11-09"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 96900, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300020, FIRST_NAME: "Nguyen Anh", LAST_NAME: "Tu", GENDER: "Male", ADDRESS: "227 Nguyen Van Cu, Q5, TPHCM", PHONE: "937885255", BIRTHDAY: DATE("2002-12-06"), REG_DATE: DATE("2022-09-04"), SPENT_MONEY: 145350, POINT: 1 });	POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300018, FIRST_NAME: "Dinh Van", LAST_NAME: "Toan", GENDER: "Male", ADDRESS: "50/34 Le Dai Hanh, Q10, TPHCM", PHONE: "837822285", BIRTHDAY: DATE("2002-04-16"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 7752, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300019, FIRST_NAME: "Ho Nguyen Thu", LAST_NAME: "Minh", GENDER: "Female", ADDRESS: "34/34B Nguyen Trai, Q5, TPHCM", PHONE: "237825224", BIRTHDAY: DATE("2003-11-09"), REG_DATE: DATE("2022-09-03"), SPENT_MONEY: 96900, POINT: 0 }); CREATE (C:CUSTOMER { CUS_ID: 300020, FIRST_NAME: "Nguyen Anh", LAST_NAME: "Tu", GENDER: "Male", ADDRESS: "227 Nguyen Van Cu, Q5, TPHCM", PHONE: "937885255", BIRTHDAY: DATE("2002-12-06"), REG_DATE: DATE("2022-09-04"), SPENT_MONEY: 145350, POINT: 1 });
PRODUCT	CREATE (P:PRODUCT { PRO_ID: 400001, PRODUCT_NAME: "Book", COUNTRY: "China", ORIGINAL_PRICE: 5000, SALE_PRICE: 8000, MFG: DATE("2022-01-01"), EXP: null,	CREATE (P:PRODUCT { PRO_ID: 400001, PRODUCT_NAME: "Book", COUNTRY: "China", ORIGINAL_PRICE: 5000, SALE_PRICE: 8000, MFG: DATE("2022-01-01"), EXP: null,

	PRODUCT_TYPE: "Requisite", VAT: 2, REMAINING_QUANTITY: 25 }); CREATE (P:PRODUCT { PRO_ID: 400002, PRODUCT_NAME: "Pen", COUNTRY: "USA", ORIGINAL_PRICE: 3000, SALE_PRICE: 5000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Requisite", VAT: 2, REMAINING_QUANTITY: 27 }); CREATE (P:PRODUCT { PRO_ID: 400003, PRODUCT_NAME: "Soap", COUNTRY: "France", ORIGINAL_PRICE: 10000, SALE_PRICE: 15000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 29 }); CREATE (P:PRODUCT { PRO_ID: 400004, PRODUCT_NAME: "Cleanser", COUNTRY: "ThaiLand", ORIGINAL_PRICE: 50000, SALE_PRICE: 100000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400005, PRODUCT_NAME: "Coca", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 9000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"),	PRODUCT_TYPE: "Requisite", VAT: 2, REMAINING_QUANTITY: 25 }); CREATE (P:PRODUCT { PRO_ID: 400002, PRODUCT_NAME: "Pen", COUNTRY: "USA", ORIGINAL_PRICE: 3000, SALE_PRICE: 5000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Requisite", VAT: 2, REMAINING_QUANTITY: 27 }); CREATE (P:PRODUCT { PRO_ID: 400003, PRODUCT_NAME: "Soap", COUNTRY: "France", ORIGINAL_PRICE: 10000, SALE_PRICE: 15000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 29 }); CREATE (P:PRODUCT { PRO_ID: 400004, PRODUCT_NAME: "Cleanser", COUNTRY: "ThaiLand", ORIGINAL_PRICE: 50000, SALE_PRICE: 100000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400005, PRODUCT_NAME: "Coca", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 9000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"),

	PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400006, PRODUCT_NAME: "Pepsi", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 9000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 29 }); CREATE (P:PRODUCT { PRO_ID: 400007, PRODUCT_NAME: "Apple", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 10000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Organic food", VAT: 2, REMAINING_QUANTITY: 26 }); CREATE (P:PRODUCT { PRO_ID: 400008, PRODUCT_NAME: "Banana", COUNTRY: "VietNam", ORIGINAL_PRICE: 5000, SALE_PRICE: 15000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Organic food", VAT: 2, REMAINING_QUANTITY: 22 }); CREATE (P:PRODUCT { PRO_ID: 400009, PRODUCT_NAME: "Tissue", COUNTRY: "China", ORIGINAL_PRICE: 2000, SALE_PRICE: 5000, MFG: DATE("2022-01-01"), EXP: null,	PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400006, PRODUCT_NAME: "Pepsi", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 9000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 29 }); CREATE (P:PRODUCT { PRO_ID: 400007, PRODUCT_NAME: "Apple", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 10000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Organic food", VAT: 2, REMAINING_QUANTITY: 26 }); CREATE (P:PRODUCT { PRO_ID: 400008, PRODUCT_NAME: "Banana", COUNTRY: "VietNam", ORIGINAL_PRICE: 5000, SALE_PRICE: 15000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Organic food", VAT: 2, REMAINING_QUANTITY: 22 }); CREATE (P:PRODUCT { PRO_ID: 400009, PRODUCT_NAME: "Tissue", COUNTRY: "China", ORIGINAL_PRICE: 2000, SALE_PRICE: 5000, MFG: DATE("2022-01-01"), EXP: null,

	PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 30 }); CREATE (P:PRODUCT { PRO_ID: 400010, PRODUCT_NAME: "Milk", COUNTRY: "Netherlands", ORIGINAL_PRICE: 5000, SALE_PRICE: 8000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400011, PRODUCT_NAME: "NoteBook", COUNTRY: "China", ORIGINAL_PRICE: 5000, SALE_PRICE: 8000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Requisite", VAT: 2, REMAINING_QUANTITY: 25 }); CREATE (P:PRODUCT { PRO_ID: 400012, PRODUCT_NAME: "Pencil", COUNTRY: "USA", ORIGINAL_PRICE: 3000, SALE_PRICE: 5000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Requisite", VAT: 2, REMAINING_QUANTITY: 27 }); CREATE (P:PRODUCT { PRO_ID: 400013, PRODUCT_NAME: "Vim", COUNTRY: "France", ORIGINAL_PRICE: 10000, SALE_PRICE: 15000, MFG: DATE("2022-01-01"), EXP: null,	PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 30 }); CREATE (P:PRODUCT { PRO_ID: 400010, PRODUCT_NAME: "Milk", COUNTRY: "Netherlands", ORIGINAL_PRICE: 5000, SALE_PRICE: 8000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400011, PRODUCT_NAME: "NoteBook", COUNTRY: "China", ORIGINAL_PRICE: 5000, SALE_PRICE: 8000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Requisite", VAT: 2, REMAINING_QUANTITY: 25 }); CREATE (P:PRODUCT { PRO_ID: 400012, PRODUCT_NAME: "Pencil", COUNTRY: "USA", ORIGINAL_PRICE: 3000, SALE_PRICE: 5000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Requisite", VAT: 2, REMAINING_QUANTITY: 27 }); CREATE (P:PRODUCT { PRO_ID: 400013, PRODUCT_NAME: "Vim", COUNTRY: "France", ORIGINAL_PRICE: 10000, SALE_PRICE: 15000, MFG: DATE("2022-01-01"), EXP: null,

	PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 29 }); CREATE (P:PRODUCT { PRO_ID: 400014, PRODUCT_NAME: "Scouring pad", COUNTRY: "ThaiLand", ORIGINAL_PRICE: 50000, SALE_PRICE: 100000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400015, PRODUCT_NAME: "Strong bow", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 9000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400016, PRODUCT_NAME: "C2", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 9000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 29 }); CREATE (P:PRODUCT { PRO_ID: 400017, PRODUCT_NAME: "Orange", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 10000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"),	PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 29 }); CREATE (P:PRODUCT { PRO_ID: 400014, PRODUCT_NAME: "Scouring pad", COUNTRY: "ThaiLand", ORIGINAL_PRICE: 50000, SALE_PRICE: 100000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400015, PRODUCT_NAME: "Strong bow", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 9000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 28 }); CREATE (P:PRODUCT { PRO_ID: 400016, PRODUCT_NAME: "C2", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 9000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 29 }); CREATE (P:PRODUCT { PRO_ID: 400017, PRODUCT_NAME: "Orange", COUNTRY: "USA", ORIGINAL_PRICE: 5000, SALE_PRICE: 10000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"),

	PRODUCT_TYPE: "Organic food", VAT: 2, REMAINING_QUANTITY: 26 }); CREATE (P:PRODUCT { PRO_ID: 400018, PRODUCT_NAME: "Mango", COUNTRY: "VietNam", ORIGINAL_PRICE: 5000, SALE_PRICE: 15000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Organic food", VAT: 2, REMAINING_QUANTITY: 22 }); CREATE (P:PRODUCT { PRO_ID: 400019, PRODUCT_NAME: "Mop", COUNTRY: "China", ORIGINAL_PRICE: 2000, SALE_PRICE: 5000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 30 }); CREATE (P:PRODUCT { PRO_ID: 400020, PRODUCT_NAME: "Milk tea", COUNTRY: "Netherlands", ORIGINAL_PRICE: 5000, SALE_PRICE: 8000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 28 });	PRODUCT_TYPE: "Organic food", VAT: 2, REMAINING_QUANTITY: 26 }); CREATE (P:PRODUCT { PRO_ID: 400018, PRODUCT_NAME: "Mango", COUNTRY: "VietNam", ORIGINAL_PRICE: 5000, SALE_PRICE: 15000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Organic food", VAT: 2, REMAINING_QUANTITY: 22 }); CREATE (P:PRODUCT { PRO_ID: 400019, PRODUCT_NAME: "Mop", COUNTRY: "China", ORIGINAL_PRICE: 2000, SALE_PRICE: 5000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Toiletries", VAT: 2, REMAINING_QUANTITY: 30 }); CREATE (P:PRODUCT { PRO_ID: 400020, PRODUCT_NAME: "Milk tea", COUNTRY: "Netherlands", ORIGINAL_PRICE: 5000, SALE_PRICE: 8000, MFG: DATE("2022-01-01"), EXP: DATE("2023-01-01"), PRODUCT_TYPE: "Drinks", VAT: 2, REMAINING_QUANTITY: 28 });
WAREHOUSE MANAGEMENT	CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400001, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 });	CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400001, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 });

	CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400002, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400003, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400004, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400005, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400006, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400007,	CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400002, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400003, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400004, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400005, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400011, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400012,

	IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400008, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400009, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400010, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400011, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400012, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 });	IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400013, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400014, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400015, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400016, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400017, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 });

	CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400013, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400014, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN1", PRO_ID: 400015, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 });	CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400018, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400019, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 }); CREATE (WM:WAREHOUSE_MANAGEMENT { BRANCH_ID: "CN2", PRO_ID: 400020, IMPORTED_DATE: DATE("2022-02-01"), IMPORTED_QUANTITY: 30 });
WAREHOUSE_SALES	CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400001, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400002, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400003, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400004, STATUS: "Con hang" });	CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400001, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400002, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400003, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400004, STATUS: "Con hang" });

	CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400005, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400006, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400007, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400008, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400009, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400010, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400011, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400012, STATUS: "Con hang" });	CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400005, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400011, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400012, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400013, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400014, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400015, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400016, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400017, STATUS: "Con hang" });

	CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400013, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400014, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400015, STATUS: "Con hang" });	CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400018, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400019, STATUS: "Con hang" }); CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400020, STATUS: "Con hang" });
BILL	CREATE (B:BILL { BILL_ID: 500001, EMP_ID: 200002, CUS_ID: 300001, BILL_DATE: DATE("2022-01-01"), TOTAL_MONEY: 53295 }); CREATE (B:BILL { BILL_ID: 500002, EMP_ID: 200003, CUS_ID: 300011, BILL_DATE: DATE("2022-02-10"), TOTAL_MONEY: 75582 }); CREATE (B:BILL { BILL_ID: 500003, EMP_ID: 200002, CUS_ID: 300003, BILL_DATE: DATE("2022-02-20"), TOTAL_MONEY: 77520 }); CREATE (B:BILL { BILL_ID: 500004, EMP_ID: 200003, CUS_ID: 300003, BILL_DATE: DATE("2022-03-01"), TOTAL_MONEY: 26163 });	CREATE (B:BILL { BILL_ID: 700001, EMP_ID: 600002, CUS_ID: 300011, BILL_DATE: DATE("2022-01-01"), TOTAL_MONEY: 53295 }); CREATE (B:BILL { BILL_ID: 700002, EMP_ID: 600003, CUS_ID: 300011, BILL_DATE: DATE("2022-02-10"), TOTAL_MONEY: 75582 }); CREATE (B:BILL { BILL_ID: 700003, EMP_ID: 600002, CUS_ID: 300006, BILL_DATE: DATE("2022-02-20"), TOTAL_MONEY: 77520 }); CREATE (B:BILL { BILL_ID: 700004, EMP_ID: 600003, CUS_ID: 300013, BILL_DATE: DATE("2022-03-01"), TOTAL_MONEY: 26163

```
CREATE (B:BILL
{
   BILL_ID: 500005,
   EMP_ID: 200002,
   CUS_ID: 300015,
   BILL_DATE: DATE("2022-03-09"),
   TOTAL_MONEY: 72675
});

CREATE (B:BILL
{
   BILL_ID: 500006,
   EMP_ID: 200003,
   CUS_ID: 300015,
   BILL_DATE: DATE("2022-03-18"),
   TOTAL_MONEY: 96900
});

CREATE (B:BILL
{
   BILL_ID: 500007,
   EMP_ID: 200002,
   CUS_ID: 300006,
   BILL_DATE: DATE("2022-03-25"),
   TOTAL_MONEY: 87210
});

CREATE (B:BILL
{
   BILL_ID: 500008,
   EMP_ID: 200003,
   CUS_ID: 300007,
   BILL_DATE: DATE("2022-04-05"),
   TOTAL_MONEY: 11628
});

CREATE (B:BILL
{
   BILL_ID: 500009,
   EMP_ID: 200002,
   CUS_ID: 300009,
   BILL_DATE: DATE("2022-04-17"),
   TOTAL_MONEY: 145350
});

CREATE (B:BILL
{
   BILL_ID: 500010,
   EMP_ID: 200003,
   CUS_ID: 300019,
   BILL_DATE: DATE("2022-04-23"),
   TOTAL_MONEY: 77520
});
```

```
});

CREATE (B:BILL
{
   BILL_ID: 700005,
   EMP_ID: 600002,
   CUS_ID: 300016,
   BILL_DATE: DATE("2022-03-09"),
   TOTAL_MONEY: 72675
});

CREATE (B:BILL
{
   BILL_ID: 700006,
   EMP_ID: 600003,
   CUS_ID: 300016,
   BILL_DATE: DATE("2022-03-18"),
   TOTAL_MONEY: 96900
});

CREATE (B:BILL
{
   BILL_ID: 700007,
   EMP_ID: 600002,
   CUS_ID: 300009,
   BILL_DATE: DATE("2022-03-25"),
   TOTAL_MONEY: 87210
});

CREATE (B:BILL
{
   BILL_ID: 700008,
   EMP_ID: 600003,
   CUS_ID: 300017,
   BILL_DATE: DATE("2022-04-05"),
   TOTAL_MONEY: 11628
});

CREATE (B:BILL
{
   BILL_ID: 700009,
   EMP_ID: 600002,
   CUS_ID: 300019,
   BILL_DATE: DATE("2022-04-17"),
   TOTAL_MONEY: 145350
});

CREATE (B:BILL
{
   BILL_ID: 700010,
   EMP_ID: 600003,
   CUS_ID: 300003,
   BILL_DATE: DATE("2022-04-23"),
   TOTAL_MONEY: 77520
});
```

BILL_DETAILS	CREATE (BD:BILL_DETAILS { BILL_ID: 500001, PRO_ID: 400001, AMOUNT: 5 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500001, PRO_ID: 400012, AMOUNT: 3 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500002, PRO_ID: 400014, AMOUNT: 4 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500002, PRO_ID: 400005, AMOUNT: 2 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500002, PRO_ID: 400003, AMOUNT: 1 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500003, PRO_ID: 400015, AMOUNT: 1 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500004, PRO_ID: 400005, AMOUNT: 2 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500004, PRO_ID: 400007, AMOUNT: 1 });	CREATE (BD:BILL_DETAILS { BILL_ID: 700001, PRO_ID: 400001, AMOUNT: 5 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700001, PRO_ID: 400012, AMOUNT: 3 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700002, PRO_ID: 400012, AMOUNT: 4 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700002, PRO_ID: 400015, AMOUNT: 2 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700002, PRO_ID: 400013, AMOUNT: 1 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700003, PRO_ID: 400017, AMOUNT: 10 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700004, PRO_ID: 400015, AMOUNT: 2 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700004, PRO_ID: 400007, AMOUNT: 1 });

	CREATE (BD:BILL_DETAILS { BILL_ID: 500005, PRO_ID: 400003, AMOUNT: 3 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500006, PRO_ID: 400014, AMOUNT: 1 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500007, PRO_ID: 400007, AMOUNT: 6 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500008, PRO_ID: 400013, AMOUNT: 3 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500009, PRO_ID: 400009, AMOUNT: 1 }); CREATE (BD:BILL_DETAILS { BILL_ID: 500010, PRO_ID: 400010, AMOUNT: 1 });	CREATE (BD:BILL_DETAILS { BILL_ID: 700005, PRO_ID: 400013, AMOUNT: 3 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700006, PRO_ID: 400005, AMOUNT: 1 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700007, PRO_ID: 400017, AMOUNT: 6 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700008, PRO_ID: 400004, AMOUNT: 3 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700009, PRO_ID: 400019, AMOUNT: 1 }); CREATE (BD:BILL_DETAILS { BILL_ID: 700010, PRO_ID: 400020, AMOUNT: 1 });

- TẠO KHÓA CHÍNH CHO CÁC LABLE

CREATE CONSTRAINT PK_BRANCH FOR (B: BRANCH) REQUIRE B.BRANCH_ID IS UNIQUE;

CREATE CONSTRAINT PK_CUSTOMER FOR (C: CUSTOMER) REQUIRE C.CUS_ID IS UNIQUE;

CREATE CONSTRAINT PK_BILLH FOR (B: BILL) REQUIRE B.BILL_ID IS UNIQUE;

CREATE CONSTRAINT PK_BILL_DETAILS FOR (BD: BILL_DETAILS) REQUIRE (BD.BILL_ID, BD.PRO_ID) IS UNIQUE;

CREATE CONSTRAINT PK_EMPLOYEEH FOR (E: EMPLOYEE) REQUIRE E.EMP_ID IS UNIQUE;

CREATE CONSTRAINT PK_PRODUCTH FOR (P: PRODUCT) REQUIRE P.PRO_ID IS UNIQUE;

CREATE CONSTRAINT PK_WAREHOUSE_MANAGEMENT FOR (WM: WAREHOUSE_MANAGEMENT) REQUIRE (WM.BRANCH_ID, WM.PRO_ID) IS UNIQUE;

CREATE CONSTRAINT PK_WAREHOUSE_SALES FOR (WS: WAREHOUSE_SALES) REQUIRE (WS.BRANCH_ID, WS.PRO_ID) IS UNIQUE;

- TẠO RELATIONSHIP GIỮA CÁC LABLE

MATCH(E:EMPLOYEE),(B:BRANCH) WHERE E.BRANCH_ID = B.BRANCH_ID CREATE (E) -[R:FK_EMP_BRANCH] -> (B);

MATCH(WH:WAREHOUSE_MANAGEMENT),(B:BRANCH) WHERE WH.BRANCH_ID = B.BRANCH_ID CREATE (WH) -[R:FK_WM_BRANCH] -> (B);

MATCH(WH:WAREHOUSE_MANAGEMENT),(P:PRODUCT) WHERE WH.PRO_ID = P.PRO_ID CREATE (WH) -[R:FK_WM_PRO] -> (P);

MATCH(WS:WAREHOUSE_SALES),(B:BRANCH) WHERE WS.BRANCH_ID = B.BRANCH_ID CREATE (WS) -[R:FK_WS_BRANCH] -> (B);

MATCH(WS:WAREHOUSE_SALES),(P:PRODUCT) WHERE WS.PRO_ID = P.PRO_ID CREATE (WS) -[R:FK_WS_PRO] -> (P);

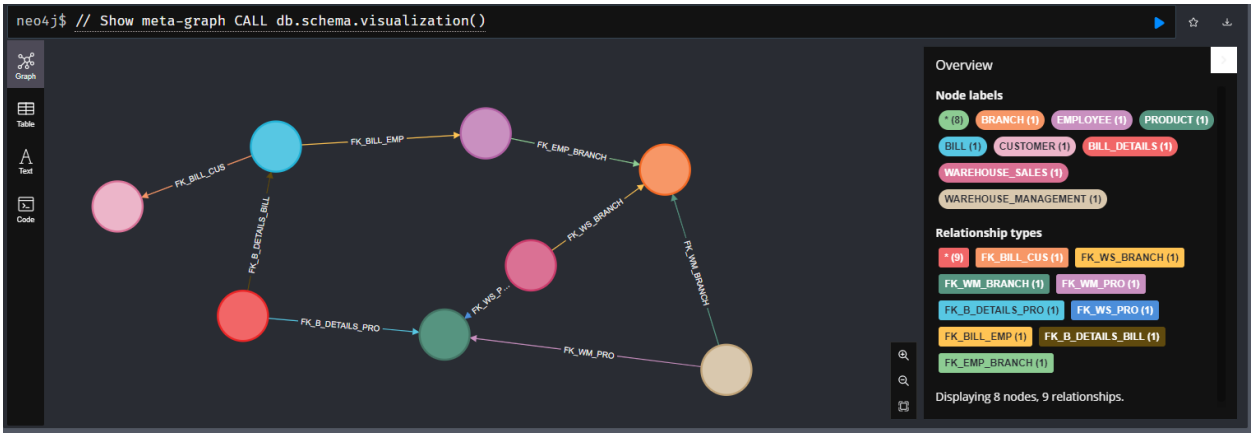
MATCH(B:BILL),(E:EMPLOYEE) WHERE B.EMP_ID = E.EMP_ID CREATE (B) -[R:FK_BILL_EMP] -> (E);

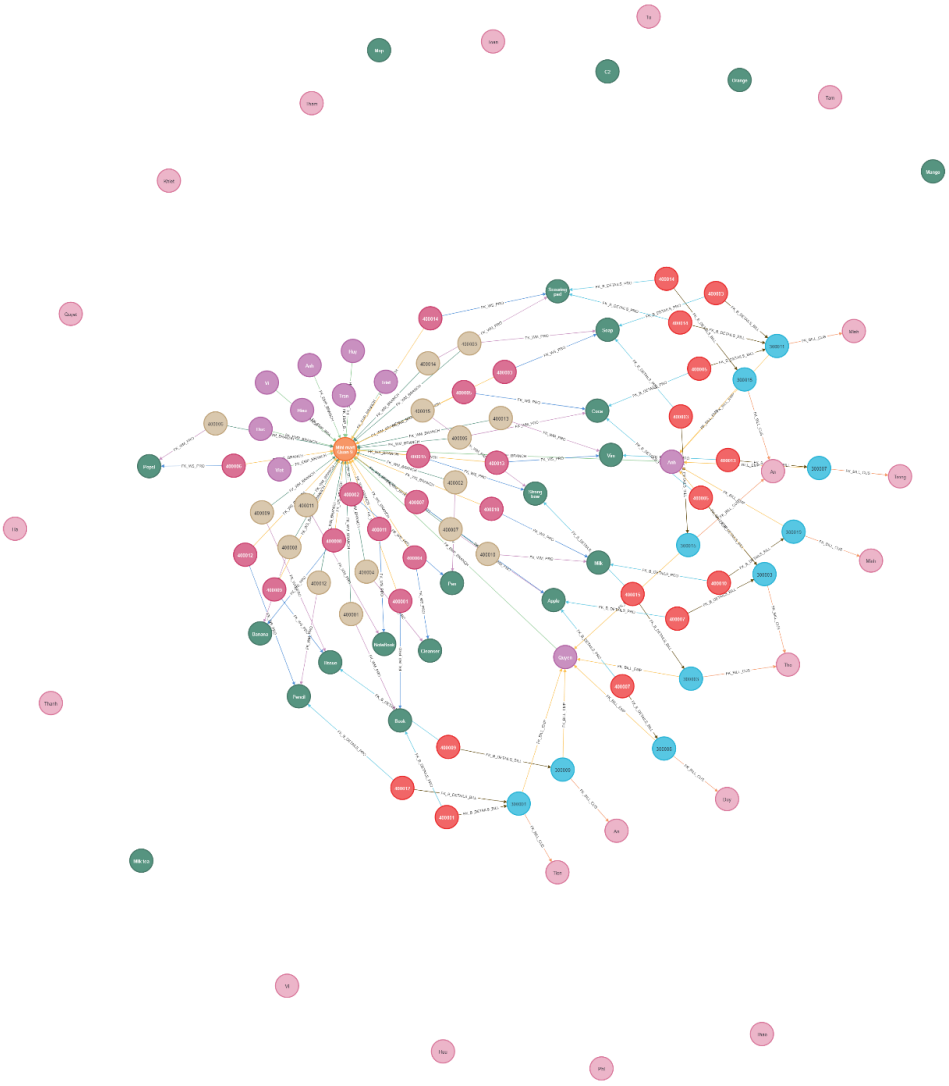
MATCH(B:BILL),(C:CUSTOMER) WHERE B.CUS_ID = C.CUS_ID CREATE (B) -[R:FK_BILL_CUS] -> (C);

MATCH(BD:BILL_DETAILS),(B:BILL) WHERE BD.BILL_ID = B.BILL_ID CREATE (BD) -[R:FK_B_DETAILS_BILL] -> (B);

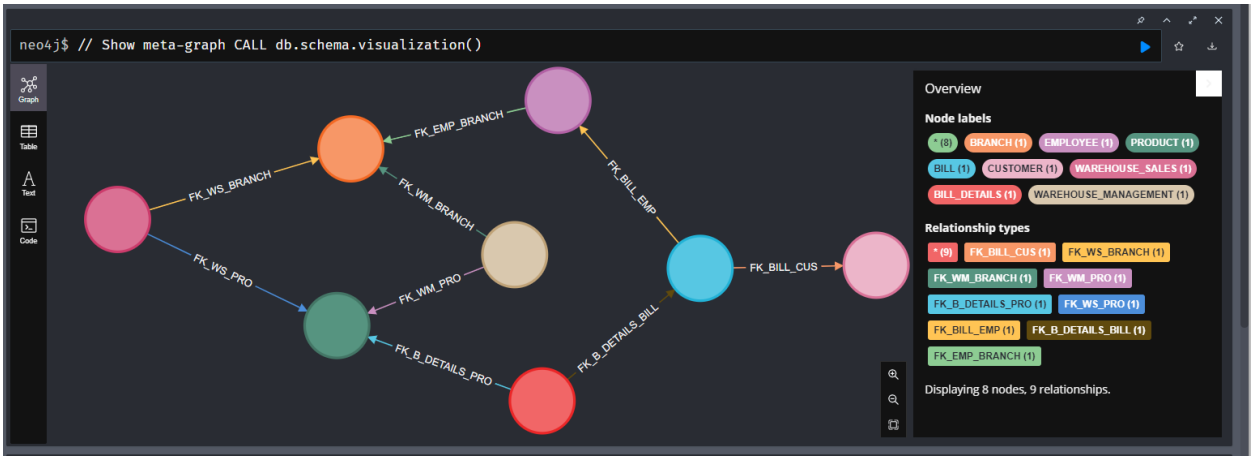
MATCH(BD:BILL_DETAILS),(P:PRODUCT) WHERE BD.PRO_ID = P.PRO_ID CREATE (BD) -[R:FK_B_DETAILS_PRO] -> (P);

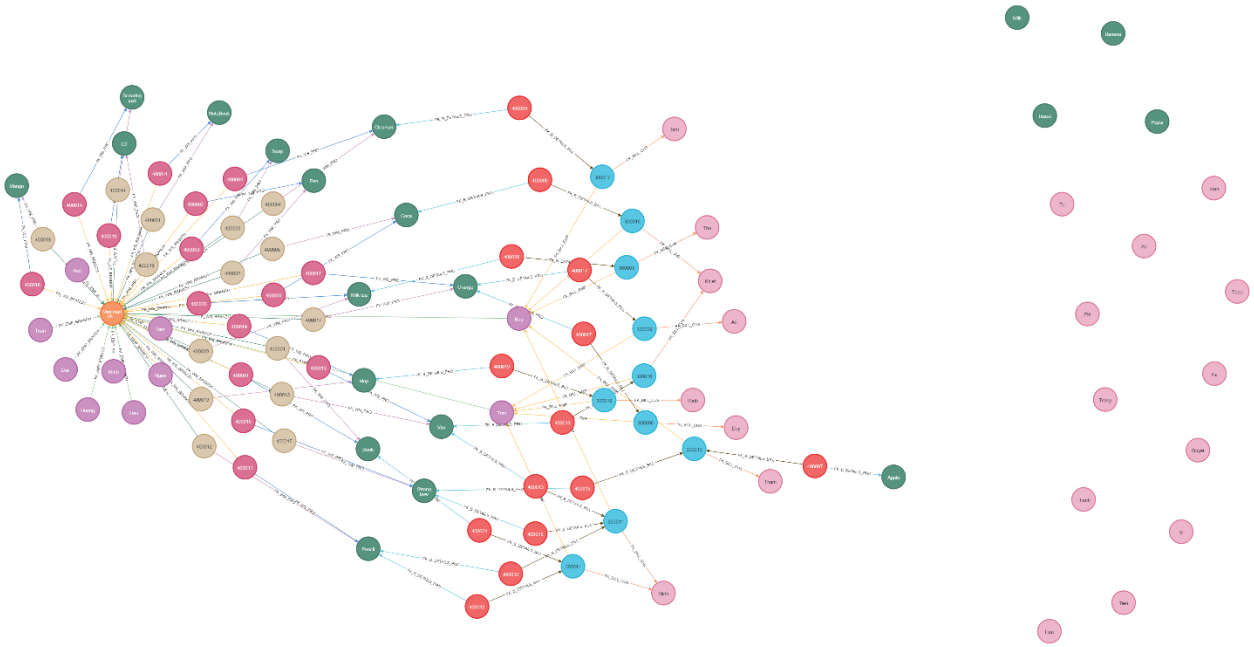
- **Kết quả thêm dữ liệu ở CN1:**





- **Kết quả thêm dữ liệu ở CN2:**





2. Query

Sử dụng python để thực hiện câu truy vấn cypher và cho ra kết quả

Câu 1: In ra mã hóa đơn, trị giá các hóa đơn và họ tên nhân viên đã thanh toán những hóa đơn này do khách hàng có tên là “Nguyen Huu Tho” mua

```
In [22]: #In ra mã hóa đơn, trị giá các hóa đơn và họ tên nhân viên đã thanh toán những hóa đơn này
         #do khách hàng có tên là "Nguyen Huu Tho"  mua
         from py2neo import Graph
         import pandas as pd
         import numpy as np

         cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
         cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
         data1 = cn1.run("MATCH   (B) -[:FK_BILL_CUS]->(C), (B)-[R:FK_BILL_EMP]->(E) \
                         WHERE  C.FIRST_NAME +' ' +C.LAST_NAME = 'Nguyen Huu Tho' \
                         RETURN B.BILL_ID, B.TOTAL_MONEY, E.FIRST_NAME +' ' +E.LAST_NAME AS EMP_FULLNAME").data()
         data2 = cn2.run("MATCH   (B) -[:FK_BILL_CUS]->(C), (B)-[R:FK_BILL_EMP]->(E) \
                         WHERE  C.FIRST_NAME +' ' +C.LAST_NAME = 'Nguyen Huu Tho' \
                         RETURN B.BILL_ID, B.TOTAL_MONEY, E.FIRST_NAME +' ' +E.LAST_NAME AS EMP_FULLNAME").data()

         d1 = pd.DataFrame(data1)
         d2 = pd.DataFrame(data2)
         df = [d1,d2]
         result = pd.concat(df, ignore_index=True)
         print(result)
```

```
   B.BILL_ID  B.TOTAL_MONEY     EMP_FULLNAME
0     500003          77520  Ton Nu Tu Quyen
1     500004          26163      Ngo Thi Anh
2     700010          77520      Ngo Thi Bap
```

Câu 2: In ra danh sách các sản phẩm (PRO_ID, PRODUCT_NAME) không bán được của nước “USA”

```
In [24]: #In ra danh sách các sản phẩm (PRO_ID, PRODUCT_NAME) không bán được của nước "USA"
         from py2neo import Graph
         import pandas as pd
         import numpy as np

         cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
         cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
         data1 = cn1.run("MATCH (P:PRODUCT) \
                         WHERE NOT (()-[:FK_B_DETAILS_PRO]->(P)) AND P.COUNTRY = 'USA' \
                         RETURN P.PRO_ID, P.PRODUCT_NAME").data()
         data2 = cn2.run("MATCH (P:PRODUCT) \
                         WHERE NOT (()-[:FK_B_DETAILS_PRO]->(P)) AND P.COUNTRY = 'USA' \
                         RETURN P.PRO_ID, P.PRODUCT_NAME").data()

         d1 = pd.DataFrame(data1)
         d2 = pd.DataFrame(data2)
         df = [d1,d2]
         result = pd.concat(df, ignore_index=True)
         print(result.drop_duplicates())
```

```
   P.PRO_ID P.PRODUCT_NAME
0    400002            Pen
1    400006          Pepsi
2    400016             C2
3    400017         Orange
```

Câu 3: Tìm sản phẩm được mua nhiều nhất

In [90]:
```
#TÌM SẢN PHẨM ĐƯỢC MUA NHIỀU NHẤT
from py2neo import Graph
import pandas as pd
import numpy as np

cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
data1 = cn1.run("MATCH (BD)-[:FK_B_DETAILS_PRO]->(P) \
                WITH  P, SUM(BD.AMOUNT) AS SLMUA \
                WITH MAX(SLMUA) AS MAX \
                MATCH (BD)-[:FK_B_DETAILS_PRO]->(P) \
                WITH  P, MAX, SUM(BD.AMOUNT) AS SL \
                WHERE SL=MAX \
                RETURN P.PRO_ID, P.PRODUCT_NAME, SL").data()
data2 = cn2.run("MATCH (BD)-[:FK_B_DETAILS_PRO]->(P) \
                WITH  P, SUM(BD.AMOUNT) AS SLMUA \
                WITH MAX(SLMUA) AS MAX \
                MATCH (BD)-[:FK_B_DETAILS_PRO]->(P) \
                WITH  P, MAX, SUM(BD.AMOUNT) AS SL \
                WHERE SL=MAX \
                RETURN P.PRO_ID, P.PRODUCT_NAME, SL").data()

d1 = pd.DataFrame(data1)
d2 = pd.DataFrame(data2)
df = pd.concat([d1,d2], ignore_index=True)
dsort = df.sort_values('SL', ascending = False)
result = dsort[dsort["SL"] == dsort['SL'].max()]
print(result)
```

```
   P.PRO_ID P.PRODUCT_NAME  SL
1    400017         Orange  16
```

Câu 4: Top 5 sản phẩm được bán nhiều nhất

In [96]:
```
#Top 5 SẢN PHẨM ĐƯỢC BÁN NHIỀU NHẤT
from py2neo import Graph
import pandas as pd
import numpy as np

cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
data1 = cn1.run("MATCH (BD)-[:FK_B_DETAILS_PRO]->(P) \
                RETURN P.PRO_ID,P.PRODUCT_NAME, SUM(BD.AMOUNT) AS SL \
                ORDER BY SL DESC").data()
data2 = cn2.run("MATCH (BD)-[:FK_B_DETAILS_PRO]->(P) \
                RETURN P.PRO_ID,P.PRODUCT_NAME, SUM(BD.AMOUNT) AS SL \
                ORDER BY SL DESC").data()

d1 = pd.DataFrame(data1)
d2 = pd.DataFrame(data2)
df = [d1,d2]
fulldata = pd.concat(df, ignore_index=True)

result = fulldata.sort_values('SL', ascending = False).head(5)

print(result)
```

```
    P.PRO_ID P.PRODUCT_NAME  SL
10    400017         Orange  16
0     400015     Strong bow  10
11    400012         Pencil   7
1     400007          Apple   7
12    400001           Book   5
```

Câu 5: Tìm tất cả khách hàng đã mua có ít nhất 3 lần và được ít nhất 2 lần nhân viên khác nhau thanh toán

In [109]:
```
#TÌM TẤT CẢ KHÁCH HÀNG ĐÃ MUA có ít nhất 3 lần và được ít nhất 2 nhân viên khác nhau thanh toán
from py2neo import Graph
import pandas as pd
import numpy as np

cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
data1 = cn1.run("MATCH (B:BILL)-[:FK_BILL_CUS]->(C:CUSTOMER) \
WITH C, count(B.BILL_ID) AS NUM_BILL,COUNT(B.EMP_ID) AS NUM_EMP \
WITH C, NUM_BILL, NUM_EMP \
RETURN C.CUS_ID, C.LAST_NAME, NUM_BILL, NUM_EMP \
ORDER BY NUM_EMP DESC, NUM_BILL DESC").data()
data2 = cn2.run("MATCH (B:BILL)-[:FK_BILL_CUS]->(C:CUSTOMER) \
WITH C, count(B.BILL_ID) AS NUM_BILL,COUNT(B.EMP_ID) AS NUM_EMP \
WITH C, NUM_BILL, NUM_EMP \
RETURN C.CUS_ID, C.LAST_NAME, NUM_BILL, NUM_EMP \
ORDER BY NUM_EMP DESC, NUM_BILL DESC").data()

d1 = pd.DataFrame(data1)
d2 = pd.DataFrame(data2)
df = [d1,d2]

fulldata = pd.concat(df, ignore_index=True).groupby(['C.CUS_ID', 'C.LAST_NAME']).sum()
result = fulldata[(fulldata["NUM_BILL"] >=3) & (fulldata["NUM_EMP"] >=2)]
print(result)
```

```
                      NUM_BILL  NUM_EMP
C.CUS_ID C.LAST_NAME
300003   Tho                 3        3
300011   Minh                3        3
```

Câu 6: Tìm các sản phẩm bán được ở cả 2 chi nhánh

In [130]:
```
#Tìm sản phẩm bán được ở cả 2 chi nhánh
from py2neo import Graph
import pandas as pd
import numpy as np

cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
data1 = cn1.run("MATCH (BD:BILL_DETAILS)-[:FK_B_DETAILS_PRO]->(P:PRODUCT) \
                 RETURN P.PRO_ID, P.PRODUCT_NAME").data()
data2 = cn2.run("MATCH (BD:BILL_DETAILS)-[:FK_B_DETAILS_PRO]->(P:PRODUCT) \
                 RETURN P.PRO_ID, P.PRODUCT_NAME").data()

d1 = pd.DataFrame(data1)
d2 = pd.DataFrame(data2)
df = [d1,d2]
result = pd.merge(d1, d2 , on=["P.PRO_ID", "P.PRODUCT_NAME"], how="inner")
print(result.drop_duplicates())
```

```
   P.PRO_ID P.PRODUCT_NAME
0    400001           Book
1    400012         Pencil
3    400005           Coca
5    400015     Strong bow
7    400007          Apple
9    400013            Vim
```

Câu 7: Nhập vào mã nhân viên, cho biết nhân viên đó làm việc tại chi nhánh nào (tính trong suốt)

In [23]:
```
#Nhập vào mã nhân viên, cho biết nhân viên đó làm việc tại chi nhánh nào
from py2neo import Graph
import pandas as pd
import numpy as np

cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))

cypher_text = "MATCH(E: EMPLOYEE) WHERE E.EMP_ID = maNV RETURN E"
print("Nhập mã nhân viên cần tìm:")
eID = input()
cypher_text = cypher_text.replace("maNV",str(eID))

employee1 = cn1.run(cypher_text).data()
employee2 = cn2.run(cypher_text).data()

if(len(employee1) > 0):
    print("Nhân viên có mã " + str(eID) + " làm việc tại CN1")
else:
    if(len(employee2) > 0):
        print("Nhân viên có mã " + str(eID) + " làm việc tại CN2")
    else:
        print("Không tìm thấy nhân viên có mã " + str(eID))
```

```
Nhập mã nhân viên cần tìm:
200001
Nhân viên có mã 200001 làm việc tại CN1
```

3. Thêm, sửa, xóa qua lại giữa hai máy

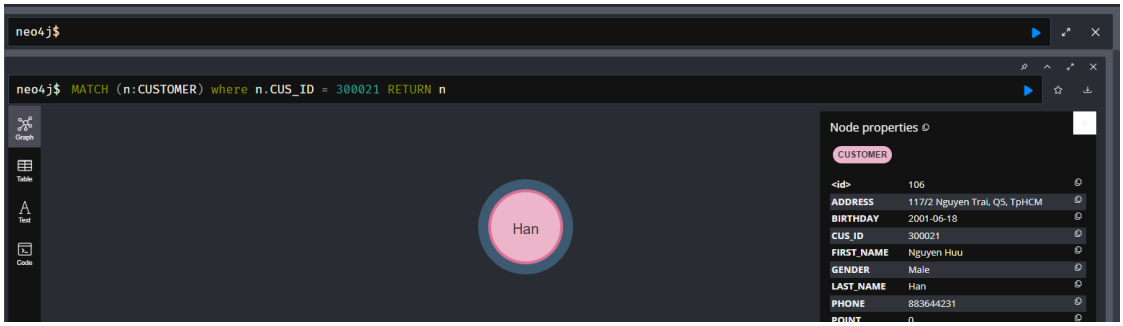
❖ Tại CN2:

- **Thêm CUSTOMER vào CN1**
    - CUSTOMER ở CN1 trước khi thêm 1 CUSTOMER bởi CN2

- Thêm customer vào CN1

```
In [132]: #Tại máy CN2, thêm customer vào CN1
          from py2neo import Graph
          import pandas as pd
          import numpy as np

          cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
          cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
          cn1.run("CREATE(C:CUSTOMER{ \
                  CUS_ID:300021, \
                  FIRST_NAME: 'Nguyen Huu', \
                  LAST_NAME: 'Han', \
                  PHONE: '883644231', \
                  ADDRESS: '117/2 Nguyen Trai, Q5, TpHCM', \
                  REG_DATE: '2022-09-01', \
                  GENDER: 'Male', \
                  POINT: 0, \
                  BIRTHDAY: '2001-06-18', \
                  SPENT_MONEY: 38760})").stats()

Out[132]: {'labels_added': 1, 'nodes_created': 1, 'properties_set': 10}
```

- Sau khi thêm CUSTOMER



- **Sửa CUSTOMER ở CN1**
    - Tại máy CN2, sửa phone của CUSTOMER CÓ mã 300021 thành 0921231741 tại CN1

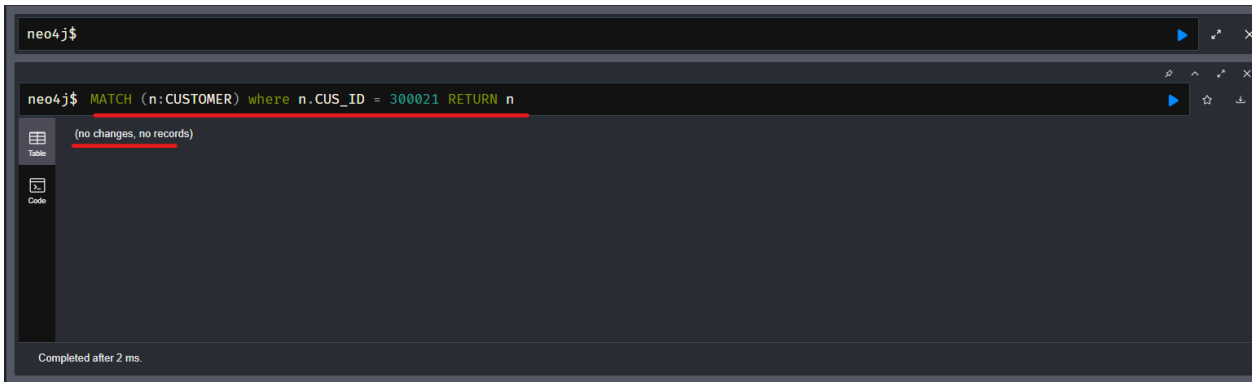
```
In [136]: #Tại máy CN2, sửa phone của CUSTOMER CÓ mã 300021 thành 0921231741 tại CN1
          from py2neo import Graph
          import pandas as pd
          import numpy as np

          cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
          cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
          cn1.run("MATCH (C:CUSTOMER {CUS_ID: 300021}) SET C.PHONE = '0921231741'").stats()

Out[136]: {'properties_set': 1}
```

    - Sau khi sửa CUSTOMER

- **Xóa CUSTOMER ở CN1**
    - Tại máy CN2, XÓA CUSTOMER CÓ MÃ 300021 tại CN1

```
In [133]: #Tại máy CN2, XÓA CUSTOMER CÓ MÃ 300021 tại CN1
          from py2neo import Graph
          import pandas as pd
          import numpy as np

          cn1 = Graph(uri="bolt://26.116.246.130:7687", auth=("neo4j","123456"))
          cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
          cn1.run("MATCH (C:CUSTOMER {CUS_ID: 300021}) DELETE C").stats()

Out[133]: {'nodes_deleted': 1}
```

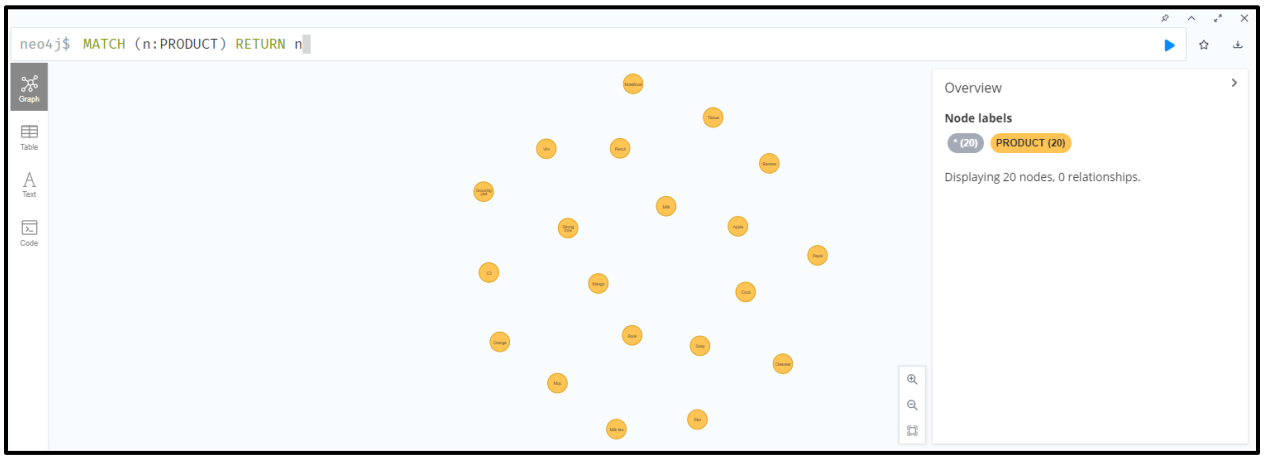
- Sau khi xóa CUSTOMER



❖ Tại CN1:

- Thêm PRODUCT vào CN2

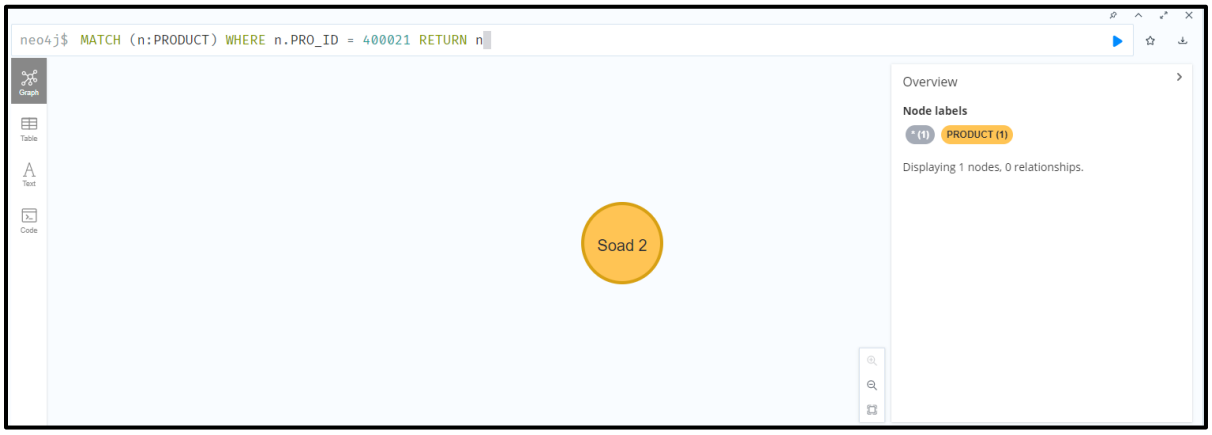
- PRODUCT ở CN2 trước khi thêm 1 PRODUCT bởi CN1



- Tại máy CN1, thêm PRODUCT vào CN2

```
In [6]: #Tại máy CN1, thêm product vào CN2
        from py2neo import Graph
        import pandas as pd
        import numpy as np
        cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
        cn2.run("CREATE(P:PRODUCT{ \
            PRO_ID: 400021, \
            PRODUCT_NAME: 'Soad 2',\
            SALE_PRICE: 8000, \
            PRODUCT_TYPE: 'Requisite', \
            ORIGINAL_PRICE: 5000, \
            COUNTRY: 'China', \
            VAT: 2, \
            REMAINING_QUANTITY: 30, \
            MFG: '2022-01-01', \
            EXP: '2023-01-01'})").stats()
Out[6]: {'labels_added': 1, 'nodes_created': 1, 'properties_set': 10}
```

- Sau khi thêm PRODUCT



- Sửa PRODUCT

- Tại máy CN1 sửa nước sản xuất của PRODUCT có mã 400021 ở CN2 thành VN

```
In [7]: #Tại máy CN1 sửa nước sản xuất của product có mã 400021 ở CN2 thành VN
        from py2neo import Graph
        import pandas as pd
        import numpy as np
        cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
        cn2.run(
            "MATCH (P:PRODUCT {PRO_ID: 400021}) \
            SET P.COUNTRY = 'VN'").stats()
Out[7]: {'properties_set': 1}
```

- Trước khi sửa PRODUCT

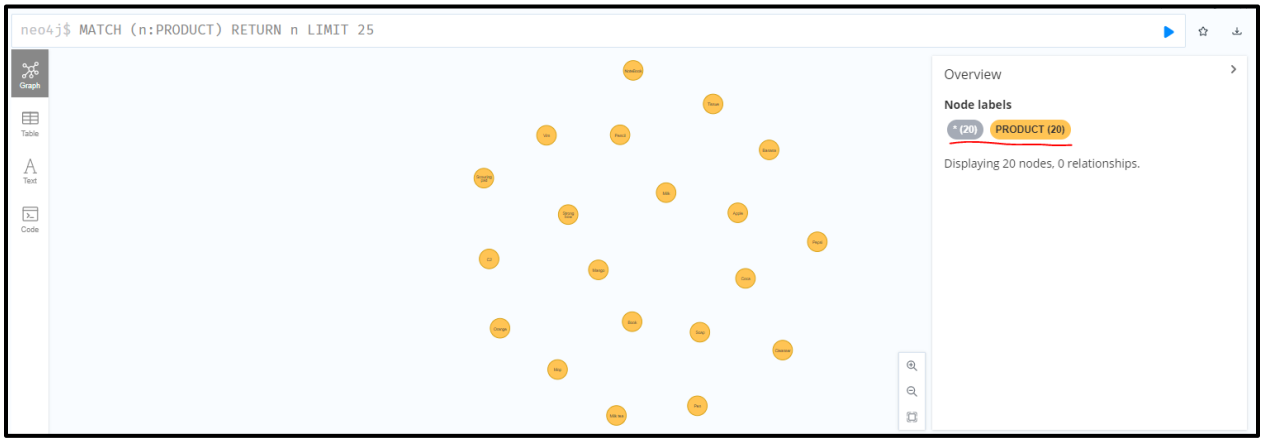
- Sau khi sửa PRODUCT

- Xóa PRODUCT

- Tại máy CN1 xóa product có mã 400021 ở CN2

```
In [8]: #Tại máy CN1 xóa product có mã 400021 ở CN2
        from py2neo import Graph
        import pandas as pd
        import numpy as np
        cn2 = Graph(uri="bolt://26.198.144.133:7687", auth=("neo4j","123456"))
        cn2.run("MATCH (P:PRODUCT {PRO_ID: 400021}) DELETE P").stats()
Out[8]: {'nodes_deleted': 1}
```

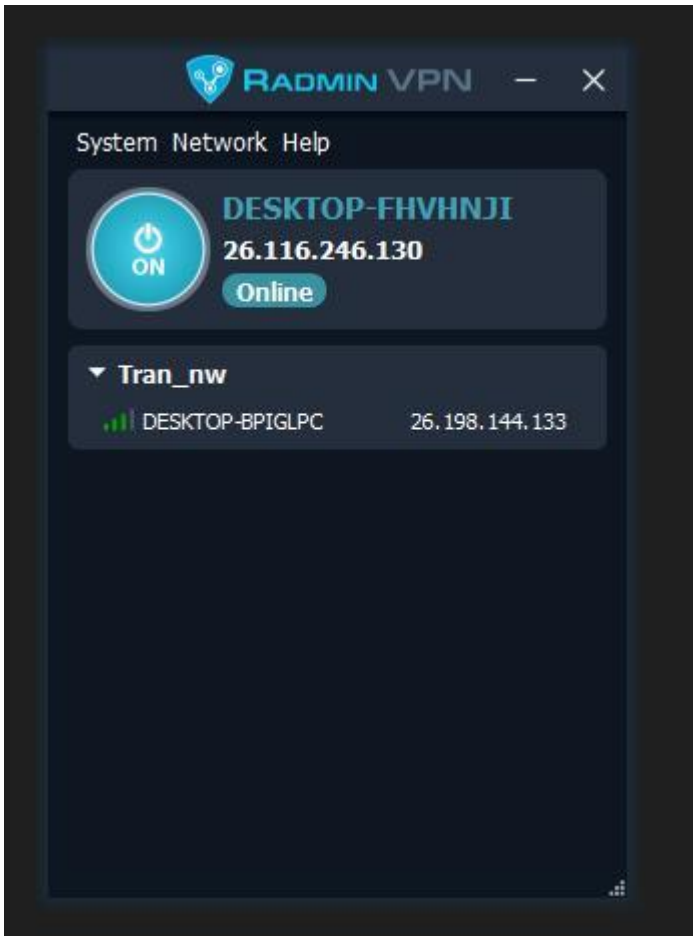
- Sau khi xóa PRODUCT



4. Cơ chế nhân bản trong phân tán NoSQL Neo4J

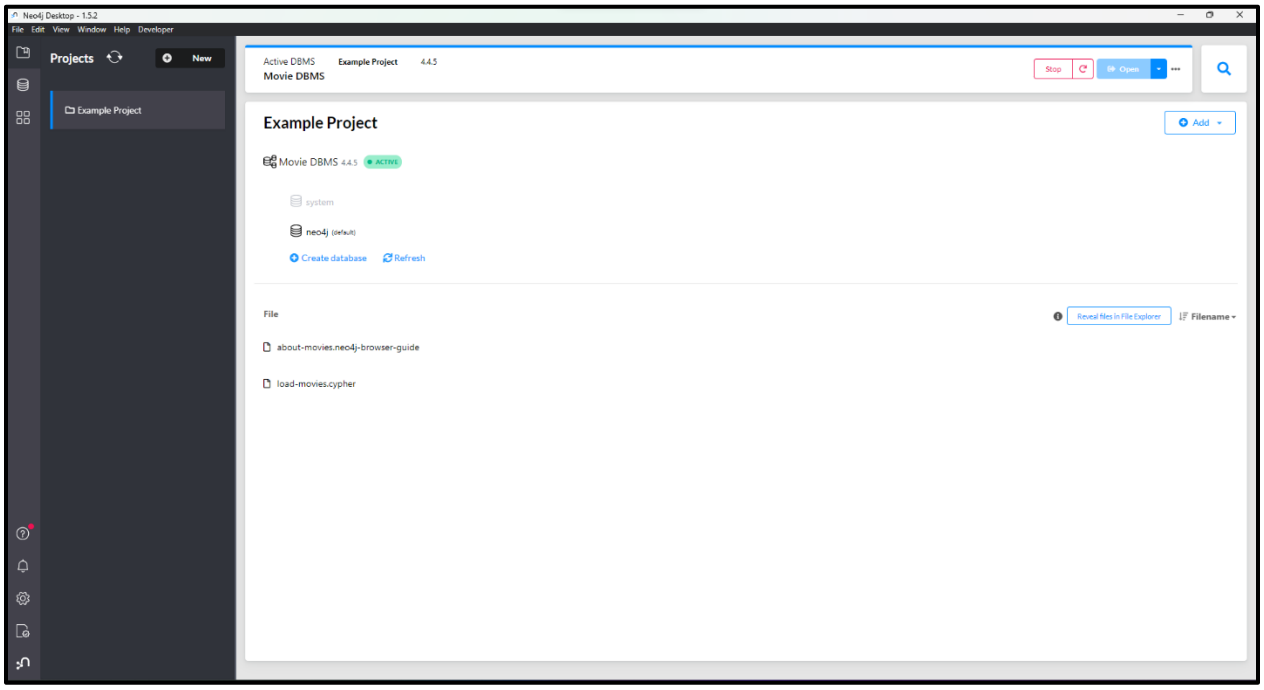
❖ Cài đặt cơ chế nhân bản trên 2 máy

- Cài đặt radmin

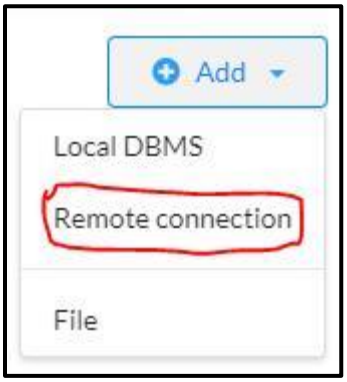


- IP CN1: 26.116.246.130
- IP CN2: 26.198.144.133

- Tạo local database ở máy chủ và thêm user cho máy trạm



- Tạo remote connection ở máy khách: Bấm “Add” -> “Remote conection”



- Nhập tên Remote connection và connect url theo mẫu: bolt://<IP máy chủ>:7687. Rồi ấn “Next”.

- Nhập user đã tạo ở máy chủ rồi ấn “Save”.

- Máy chủ kết nối vào local database, máy khách kết nối vào remote connection

E. PHÂN CÔNG CÔNG VIỆC

Công việc	Trân	Quyên	Đức	Huy
Mục A	A.1	A.2, A.3, A.4	A.5, A.6, A.7	A.8
Mục B			X	X
Mục C	X	X		
Mục D	X	X		
Làm slide			X	
Làm file báo cáo	X			X
Thuyết trình			X	X
Demo	X	X		

TÀI LIỆU THAM KHẢO

[1] "Neo4j documentation," [Online]. https://neo4j.com/docs/.

[2] "Dang Thi Ngoc Anh," tim-hieu-ve-ngon-ngu-truy-van-cypher, [Online]. https://viblo.asia/p/tim-hieu-ve-ngon-ngu-truy-van-cypher-gDVK2BmAKLj.

[3] "Lam-quen-voi-Neo4j," [Online]. https://ai-blog.bappartners.com/neo4j-for-data-science/.

[4] "python-documentation," [Online]. https://www.python.org/doc/.

[5] "py2neo, " [Online]. https://py2neo.org/2021.1/index.html#cypher

[6] "pandas" [Online]. https://pandas.pydata.org/docs/user_guide/10min.html