CƠ CHẾ PHÂN TÁN TRONG HỆ QUẢN TRỊ NOSQL- NEO4J

Nguyễn Thị Mỹ Trân
Khoa Hệ thống Thông Tin
Đại học Công Nghệ Thông Tin
Thành phố Hồ Chí Minh, Việt Nam
20520322@gm.uit.edu.vn

Tôn Nữ Tú Quyên
Khoa Hệ thống Thông Tin
Đại học Công Nghệ Thông Tin
Thành phố Hồ Chí Minh, Việt Nam
20520296@gm.uit.edu.vn

Thái Tăng Đức
Khoa Hệ thống Thông Tin
Đại học Công Nghệ Thông Tin
Thành phố Hồ Chí Minh, Việt Nam
20521203@gm.uit.edu.vn

Trần Anh Huy
Khoa Hệ thống Thông Tin
Đại học Công Nghệ Thông Tin
Thành phố Hồ Chí Minh, Việt Nam
20520296@gm.uit.edu.vn

Tóm tắt: Neo4j là một hệ quản trị cơ sở dữ liệu NoSQLNeo4j được triển khai bằng Java và có thể truy cập được từ phần mềm được viết bằng các ngôn ngữ khác bằng cách sử dụng ngôn ngữ truy vấn Cypher thông qua điểm cuối HTTP giao dịch hoặc thông qua giao thức " Bolt " nhị phân "4j" trong Neo4j ám chỉ đến việc nó được xây dựng bằng Java.

Từ khóa: Neo4j, phân tán cơ sở dữ liệu Neo4j, config Neo4j, python, py2neo, pandas, ngôn ngữ cypher

I. GIỚI THIỆU VỀ HỆ QUẢN TRỊ CSDL NOSQL NEO4J

A. 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

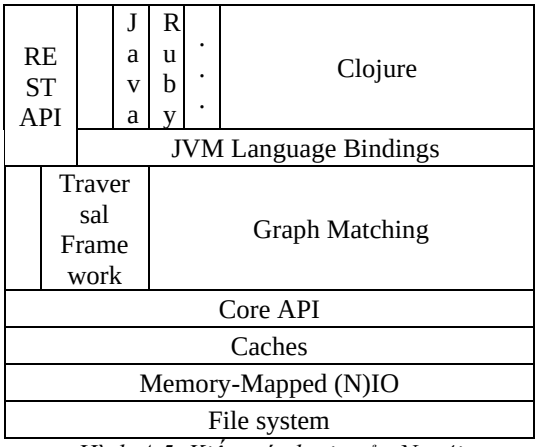
*Lịch sử
- Năm 2000: Những người sáng lập Neo4j là Emil, Johan và Peter đã gặp phải các vấn đề về hiệu suất với RDBMS và bắt đầu xây dựng Neo4j prototype đầu tiên
- Năm 2002: Đã phát triển phiên bản Neo4j đầu tiên
- Năm 2003: Triển khai Neo4j sản xuất 24 × 7 đầu tiên
- Năm 2007: Thành lập một công ty 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)

B. Cấp phép và phiên bản:

* 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).

C. Kiến trúc của Neo4j

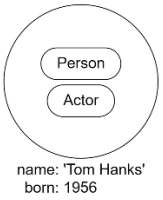


Hình A.5. Kiến trúc logic của Neo4j

- 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,…

D. 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:

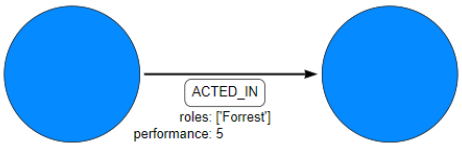


Hình A.6. Ví dụ một node

- Các node label bao gồm:
  - Person
  - Actor
- Các thuộc tính bao gồm:
  - name: Tom Hanks
  - born: 1956

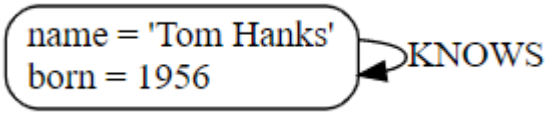
E. Relationship

- 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.



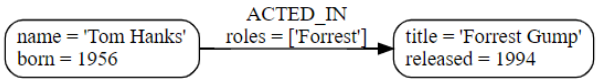
Hình A.7.1. Ví dụ relationship giữa 2 node

  - 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

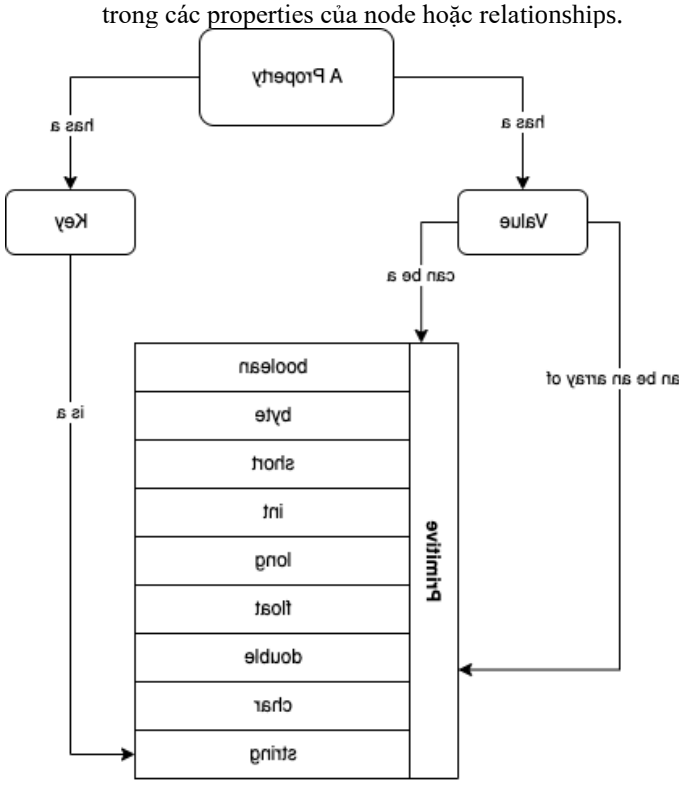
- Quan sát và biết được rằng node Tom Hanks có một outgoing relationship, trong khi đó node Forrest Gump có một incoming relationship.

F. 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.



II. NGÔN NGỮ CYPHER TRONG NEO4J

A. 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.

Neo4j lưu trữ dữ liệu trên các nút, xây dựng các cấu trúc dữ liệu khác nhau dựa trên relationships.

- Nút(node): Là một trong hai đơn vị cơ bản tạo nên đồ thị.
- Các mối quan hệ(relationships): Một mối quan hệ kết hai nút
- Thuộc tính (property)
- Nhãn (label): Là tên một cấu trúc đồ thị để nhóm các node vào một tập hợp (bộ)
- Duyệt đồ thị (traversal): Là cách truy vấn đồ thị, điều hướng bắt đầu từ một node đến các node liên quan.

B. Định dạng của cypher

Định dạng nút (nodes)

- (): 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 (relationship)

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

C. Các lệnh cơ bản trong Cypher

- 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.
- CREATE: Tạo Node, Relationship hoặc một Path.
- UPDATE: Thêm, sửa, xóa các thuộc tính (property) trong 1 nút.
- SET: Dùng để cập nhật Labels, Properties.
- DELETE: Xóa Node, Relationship.
- MERGE: đảm bảo pattern luôn tồn tại trong đồ thị, nếu không tồn tại, MERGE sẽ giúp tạo pattern đó.
- WHERE: Thêm các ràng buộc cho pattern.
- WITH: thực hiện các thao tác với các output trước khi sang các mệnh đề khác.
- LIMIT, ORDER BY : tương tự như SQL.
- 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.

III. CÀI ĐẶT TRÊN 2 MÁY TRỞ LÊN

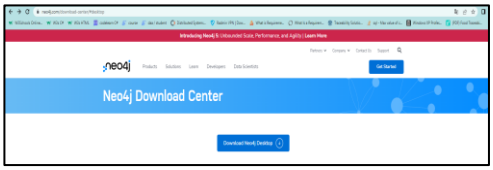
Hiện tại, Neo4j phiên bản Desktop vẫn chưa hỗ trợ việc phân tán ngay trong ứng dụng. Vậy nên, để có thể thực hiện phân tán và truy vấn dữ liệu qua lại giữa hai máy, ta cần phải cài đặt Neo4j cùng các ứng dụng và môi trường liên quan.

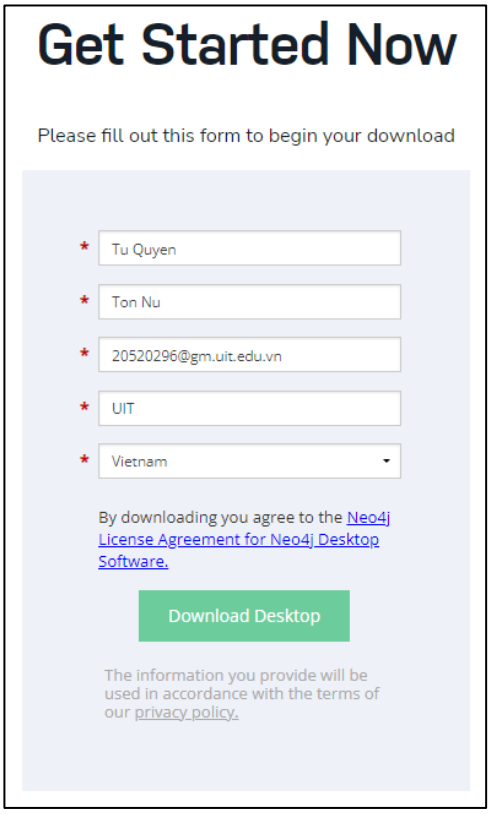
Ở đây, nhóm chúng em chọn Python làm môi trường và Jupyter Notebook làm ứng dụng thực hiện quá trình phân tán và các câu lệnh truy vấn qua lại giữa 2 máy dựa trên nền tảng ngôn ngữ truy vấn Cypher của Neo4j.

A. Cài đặt trên nhiều máy

1. Cài đặt Neo4j Desktop
   - Vào trang web https://neo4j.com/download-center/#desktop. Nhấn nút "Download Neo4j

Desktop” và tiến hành đăng ký tài khoản rồi download theo hướng dẫn của Neo4j





- Video hướng dẫn của Neo4j: https://youtu.be/hIvNexwVYNw

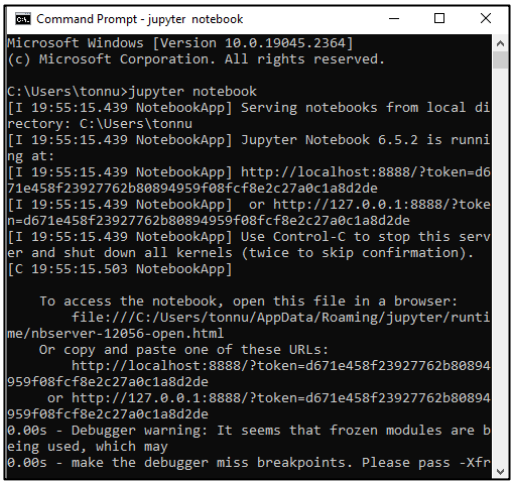
2. Cài đặt môi trường Python và các thư viện liên quan
   - Vào trang chủ của python theo đường dẫn https://www.python.org/downloads/
   - Sau đó rê chuột vào Downloads chọn download Python bản mới nhất.

   - 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 tức là đã cài đặt môi trường thành công.

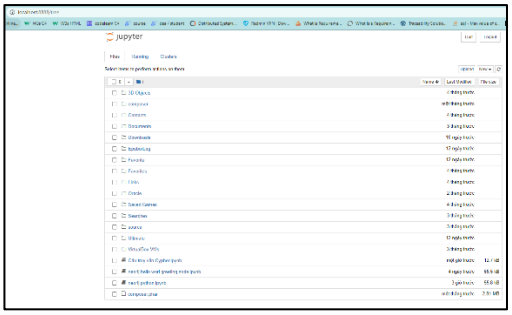
```
C:\Users\tonnu>python
Python 3.11.0 (main, Oct 24 2022, 18:26:48) [MSC
v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license"
for more information.
>>>
```

   - 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.

3. Cài đặt Jupyter Notebook
   - Vào cmd, chạy dòng lệnh “pip install notebook” rồi chờ ứng dụng Jupyter Notebook được cài thành công.
   - Sau khi đã cài thành công, tiếp tục chạy dòng lệnh “jupyter notebook” trong cmd để khởi động ứng dụng Jupyter Notebook.

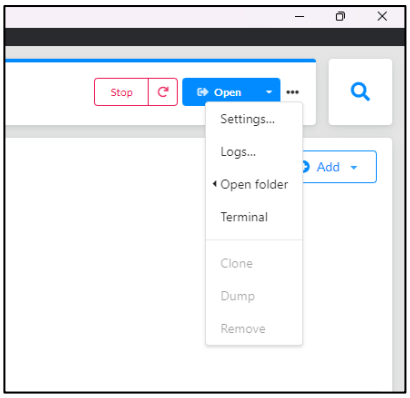


   - Trình duyệt web sẽ tự động mở ra trang Jupyter Notebook như sau:



B. *Config dữ liệu*

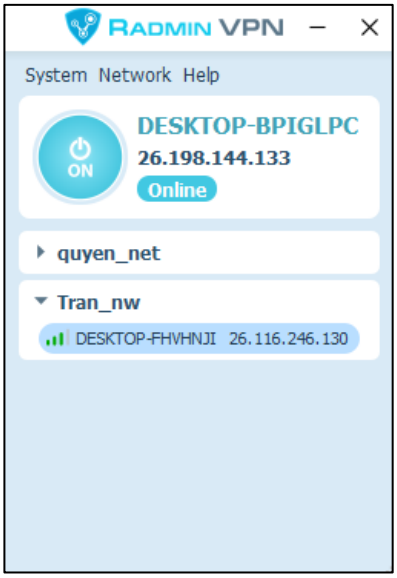
1. Config Neo4j:
   - Khởi động Neo4j và Start database sử dụng để phân tán
   - Nhấn dấu 3 chấm bên cạnh tên nút “Open”, chọn “Settings…”



- Tìm dòng “server.default_listen_address=0.0.0.0” rồi uncomment. Sau đó nhấn “Apply”.

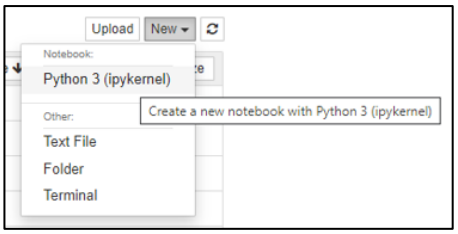
2. Kết nối VPN

- Sử dụng Radmin VPN để thực hiện kết nối VPN thông qua mạng LAN ảo.



3. Tạo project trên Jupyter Notebook

- Nhấn vào “New” chọn “Python 3” để tạo project mới.

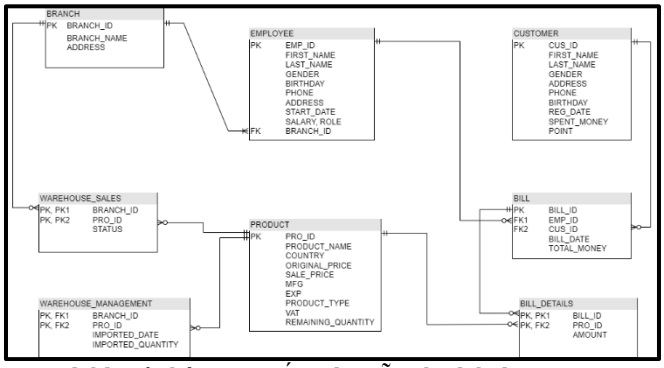


- Sau đó viết các dòng lệnh sau để hoàn thành khâu chuẩn bị cho việc phân tán. Lưu ý, database cần phải được open ở Neo4j của cả 2 máy để kết nối thành công.

```
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
```

IV. THAO TÁC QUA LẠI GIỮA 2 MÁY

A. Thêm dữ liệu



* Tạo lable và thêm một số node mẫu cho 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",	CREATE (E:EMPLOYEE { EMP_ID: 600001, FIRST_NAME: "Nguyen Thi My", LAST_NAME: "Tam", 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" });	BIRTHDAY: DATE("2002-05-05"), PHONE: "921231741", ADDRESS: "KTX khu B", START_DATE: DATE("2022-02-21"), SALARY: 10000000, ROLE: "Manager", 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: 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 });
PRODUCT	CREATE (P:PRODUCT { PRO_ID: 400001, PRODUCT_NAME: "Book", COUNTRY: "China",	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 });	ORIGINAL_PRICE: 5000, SALE_PRICE: 8000, MFG: DATE("2022-01-01"), EXP: null, PRODUCT_TYPE: "Requisite", VAT: 2, REMAINING_QUANTITY: 25 });
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 });
WAREHOUSE_SALES	CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN1", PRO_ID: 400001, STATUS: "Con hang" });	CREATE (WS:WAREHOUSE_SALES { BRANCH_ID: "CN2", PRO_ID: 400001, 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: 700001, EMP_ID: 600002, CUS_ID: 300011, BILL_DATE: DATE("2022-01-01"), TOTAL_MONEY: 53295 });

BILL_DETAILS	CREATE (BD:BILL_DETAILS { BILL_ID: 500001, PRO_ID: 400001, AMOUNT: 5 });	CREATE (BD:BILL_DETAILS { BILL_ID: 700001, PRO_ID: 400001, AMOUNT: 5 });

* 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);

B. Query

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

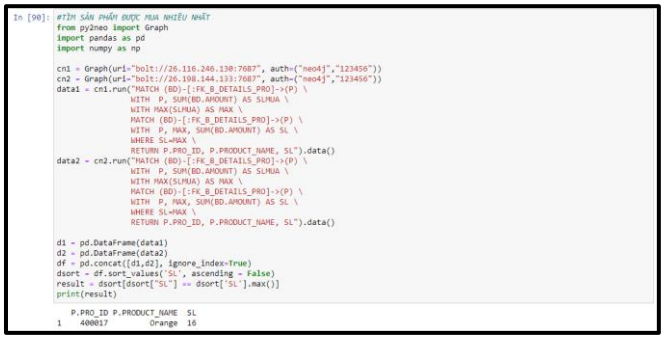
Câu 1: In ra mã hóa đơn, trị giá các hóa đơn và họ tên nhân viên đã thanh toán những hóa đơn này do khách hàng có tên là “Nguyen Huu Tho” mua.

```
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 [98]: #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  10
```

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
```

C. Cơ chế nhân bản trong phân tán NoSQL NEO4J

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

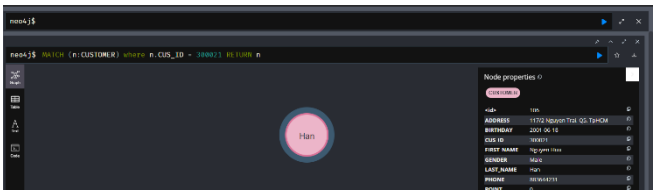
* Tại CN2:

- 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

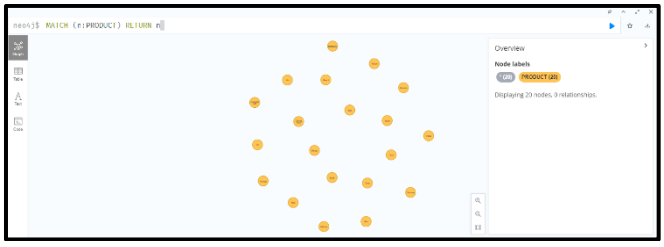
- Xóa CUSTOMER ở CN1
  - Tại máy CN2, XÓA CUSTOMER CÓ MÃ 300021 tại CN1

```
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:

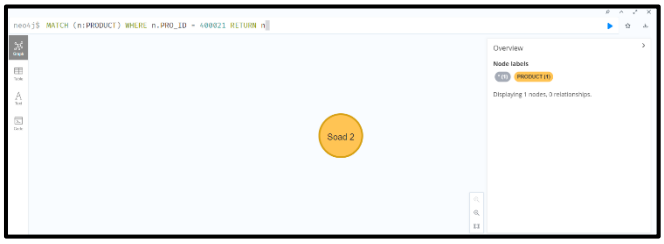
- Thêm PRODUCT vào CN2
  - PRODUCT ở CN2 trước khi thêm 1 PRODUCT bởi CN1



- 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

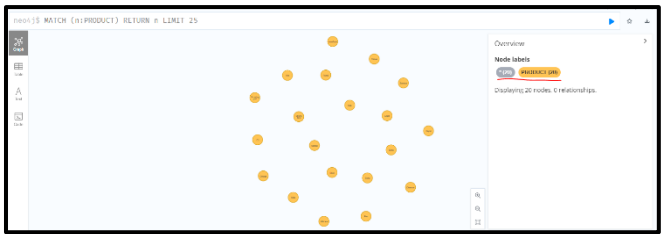
- Sau khi sửa PRODUCT

- 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



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.

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