

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP THỰC PHẨM TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

 **----------🙠☯🙢---------**

**BÁO CÁO ĐỒ ÁN MÔN**

**CƠ SỞ DỮ LIỆU NOSQL**

**ĐỀ TÀI**

**TÌM HIỂU CÁC CHỨC NĂNG CỦA CSDL NEO4J, KẾT NỐI CSDL VỚI ỨNG DỤNG QUẢN LÝ CỬA HÀNG XE MÁY.**

**GVHD:** Nguyễn Thị Định

**SVTH:**

Trần Phát 2001180300

Mai Quốc Đạt 2001180432

Đinh Quốc Khánh Nguyên 2001180226

**TP. HỒ CHÍ MINH – 4/2021**

# BẢNG ĐÁNH GIÁ KẾT QUẢ THỰC HIỆN CÔNG VIỆC NHÓM

|  |  |  |  |
| --- | --- | --- | --- |
| Danh sách công việc | Trần Phát | Đinh Quốc Khánh Nguyên | Mai Quốc Đạt |
| Tạo CSDL tác nghiệp | X | X | X |
| Tạo Form |  | X |  |
| Làm báo cáo work | X |  | X |
| Làm báo cáo PP | X |  | X |
| Đánh giá phần trăm làm việc | 30% | 40% | 30% |

# LỜI CẢM ƠN

Để hoàn thành đồ án này, trước hết chúng em xin cảm ơn quý thầy, cô trong Khoa Công Nghệ Thông Tin Trường Đại Học Công Nghiệp Thực Phẩm TP. Hồ Chí Minh đã truyền đạt kiến thức và kinh nghiệm quý báo cho nhóm chúng em trong suốt quá trình học tập và rèn luyện.

Trong quá trình thực hiện đề tài nhóm đã gặp không ít khó khăn. Nhưng với sự giúp đỡ của quý thầy cô, bạn bè, nhóm đã hoàn thành đề tài nghiên cứu của mình và có được những kinh nghiệm, được trao dồi thêm kiến thức hữu ích cho bản thân.

Đặc biệt nhóm chúng em chân thành cảm ơn cô Nguyễn Thị Định, cảm ơn cô đã tận tình chỉ bảo, giúp đỡ chúng em, trực tiếp hướng dẫn chúng em trong suốt thời gian thực hiện đề tài.

Dù có cố gắng nhưng không thể tránh khỏi những sai sót. Rất mong sự thông cảm và đóng góp ý kiến của quý thầy cô để đồ án được hoàn thiện hơn.

Cuối cùng, xin kính chúc quý thầy cô sức khoẻ, luôn thành công trong công việc và cuộc sống.

# LỜI MỞ ĐẦU

Cơ sở dữ liệu đồ thị là một vấn đề mới mà chúng ta cần phải nghiên cứu hiện nay. Đỉnh cao của ứng dụng cơ sở dữ liệu đồ thị là dùng để lưu trữ dữ liệu của một hay nhiều mạng xã hội một cách dễ dàng và hiệu quả. Nếu như cơ sở dữ liệu truyền thống biểu diễn bởi các bảng và mối quan hệ giữa chúng thông qua khóa ngoại, thì cơ sở dữ liệu đồ thị biểu diễn trực quan hơn bởi các node, thể hiện các mối quan hệ khác nhau giữa các node, các cạnh, dựa trên các quan hệ đó để tính toán các lợi ích kinh tế khác. Có nhiều cơ sở dữ liệu đồ thị khác nhau, tuy nhiên, Neo4j được xem cơ sở dữ liệu hiệu quả nhất, bằng việc lưu trữ nhiều cơ sở dữ liệu đồ thị, tốc độ truy vấn nhanh và khả năng lưu trữ tốt.

Nội dung bài tiểu luận tập trung vào trình bày cơ sở dữ liệu đồ thị neo4j và cách tiếp cận neo4j trong một ứng dụng quản lý cửa hàng xe máy. Em xin chân thành cảm ơn cô Nguyễn Thị Định đã truyền đạt những kiến thức quý báu về bộ môn Cơ sở dữ liệu và NoSql, và đặc biệt là những kiến thức về cơ sở dữ liệu đồ thị, làm tiền đề cho những ý tưởng của em về việc xây dựng một ứng dụng quản lý cửa hàng xe máy.

Bài viết còn nhiều sai xót do sự hạn chế về thời gian cũng như kinh nghiệm thực tế, mong cô thông cảm và bỏ qua.

# DANH MỤC HÌNH ẢNH

*Hình 1.1. Ví dụ về đồ thị*

*Hình 1.2.2: Tổ chức của một cơ sở dữ liệu*

*Hình 1.2.4. Mô hình truy vấn cơ sở dữ liệu đồ thị*

*Hình 1.2.5. Quan hệ giữa chỉ mục và các thành phần trong cơ sở dữ liệu đồ thị*

*Hình 1.2.5.1. Tổng quan về cơ sở dữ liệu đồ thi*

*Hình 1.2.5.2. Ví dụ về cơ sở dữ liệu đồ thị*

*Hình 2.4. Kiến trúc logic của Neo4j*

*Hình 2.5.1. Node, relationships và properties*

*Hình 2.5.2. Ví dụ về một đồ thị đơn giản với 1 node và 1 properties*

*Hình 2.6. Relationships và các quan hệ liên quan*

*Hình 2.6.1. Một relationships giữa start node và end node*

*Hình 2.6.2. Quan hệ vào và quan hệ ra của 1 node*

*Hình 2.6.3. Quan hệ trên 1 node*

*Hình 2.6.4. Ví dụ một mạng xã hội*

*Hình 2.7. Biểu diễn 1 Properties*

*Hình 2.8.1. Path*

*Hình 2.8.2. Đường đi đến chính node đó bằng 0*

*Hình 2.8.3 Đường đi từ node 1 đến node 2 có quan hệ relatonship 1*

*Hình 5.1. Database*

*Hình 5.2. Cơ sở dữ liệu đồ thị*

*Hình 5.3. Form đăng nhập*

*Hình 5.4. Form hệ thống*

*Hình 5.5. Form danh sách các loại xe*

*Hình 5.6. Tìm kiếm xe*

*Hình 5.7. Form hóa đơn nhập hàng*

*Hình 5.8. Thêm hóa đơn nhập hàng*

*Hình 5.9. Sửa hóa đơn nhập hàng*

*Hình 5.10. Xóa hóa đơn nhập hàng*

*Hình 5.11. form danh sách các nhà cung cấp*

*Hình 5.12. Form hóa đơn bán hàng*

*Hình 5.13. Thêm hóa đơn bán hàng*

*Hình 5.14. Sửa hóa đơn bán hàng*

*Hình 5.15. Xóa hóa đơn bán hàng*

**MỤC LỤC**

[BẢNG ĐÁNH GIÁ KẾT QUẢ THỰC HIỆN CÔNG VIỆC NHÓM](#_Toc70085965)

[LỜI CẢM ƠN](#_Toc70085966)

[LỜI MỞ ĐẦU](#_Toc70085967)

[DANH MỤC HÌNH ẢNH](#_Toc70085968)

[CHƯƠNG 1: TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU ĐỒ THỊ 1](#_Toc70085969)

[1.1. LÝ THUYẾT ĐỒ THỊ 1](#_Toc70085970)

[1.2. Cơ sở dữ liệu đồ thị 1](#_Toc70085971)

[1.2.1. Khái niệm 1](#_Toc70085972)

[1.2.2. Một đồ thị chứa các node và các quan hệ 2](#_Toc70085973)

[1.2.3 Các mối quan hệ tổ chức nên đồ thị 3](#_Toc70085974)

[1.2.4 Truy vấn cơ sở dữ liệu đồ thị bằng cách duyệt đồ thị 3](#_Toc70085975)

[1.2.5. Indexing 4](#_Toc70085976)

[1.2.6. Tính chất 7](#_Toc70085977)

[CHƯƠNG 2: GIỚI THIỆU VỀ NEO4J 8](#_Toc70085978)

[2.1. LỊCH SỬ 8](#_Toc70085979)

[2.2. CẤP PHÉP VÀ PHIÊN BẢN 8](#_Toc70085980)

[2.4. KIẾN TRÚC CỦA NEO4J 9](#_Toc70085981)

[2.5. NODES 10](#_Toc70085982)

[2.6. RELATIONSHIPS 11](#_Toc70085983)

[2.7. PROPERTIES 13](#_Toc70085984)

[2.8. ĐƯỜNG ĐI PATHS 14](#_Toc70085985)

[2.9. DUYỆT ĐỒ THỊ 15](#_Toc70085986)

[2.10. ƯU ĐIỂM NEO4J 15](#_Toc70085987)

[2.11. NHƯỢC ĐIỂM NEO4J 15](#_Toc70085988)

[2.12 SO SÁNH NEO4J VỚI CƠ SỞ DỮ LIỆU QUAN HỆ (RDBMD) 15](#_Toc70085989)

[2.13. NEO4J SO VỚI CÁC CƠ SỞ DỮ LIỆU NOQUERY KHÁC 17](#_Toc70085990)

[CHƯƠNG 3. CÀI ĐẶT NEO4J 19](#_Toc70085991)

[3.1. CÀI ĐẶT NEO4J 19](#_Toc70085992)

[3.2. KHÁM PHÁ BỘ DỮ LIỆU 25](#_Toc70085993)

[3.2.1. Ví dụ tạo các node tạo khóa cho nó 25](#_Toc70085994)

[3.2.2. Ví dụ tạo các relationship cho các đối tượng 26](#_Toc70085995)

[3.2.3. Ví dụ về truy vấn trong Neo4J 27](#_Toc70085996)

[3.2.4. Ví dụ về truy vấn where 29](#_Toc70085997)

[3.3. THƯ VIỆN APOC NEO4J 31](#_Toc70085998)

[3.3.1. Cài đặt thư viện APOC 31](#_Toc70085999)

[3.3.2. Các Apoc được sử dụng trong đề tài 33](#_Toc70086000)

[CHƯƠNG 4: GIỚI THIỆU ĐỀ TÀI 36](#_Toc70086001)

[4.1. LÝ DO CHỌN ĐỀ TÀI 36](#_Toc70086002)

[4.2. MỤC ĐÍCH VÀ NHIỆM VỤ NGHIÊN CỨU 36](#_Toc70086003)

[4.3. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU 36](#_Toc70086004)

[4.4. Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN CỦA ĐỀ TÀI 37](#_Toc70086005)

[4.5. PHƯƠNG PHÁP NGHIÊN CỨU 37](#_Toc70086006)

[CHƯƠNG 5: THIẾT KẾ GIAO DIỆN 38](#_Toc70086007)

[TÀI LIỆU THAM KHẢO 47](#_Toc70086008)

# CHƯƠNG 1: TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU ĐỒ THỊ

## 1.1. LÝ THUYẾT ĐỒ THỊ

Đồ thị là một tập các đối tượng được gọi là các đỉnh nối với nhau bởi các cạnh( hoặc cung). Đồ thị thường được biểu diễn dưới dạng một tập các điểm nối với nhau bằng các đoạn

Shape

Description automatically generated

*Hình 1.1. Ví dụ về đồ thị*

## 1.2. Cơ sở dữ liệu đồ thị

### 1.2.1. Khái niệm

Trong máy tính, cơ sở dữ liệu đồ thị (GDB) là cơ sở dữ liệu sử dụng các cấu trúc biểu đồ cho các truy vấn ngữ nghĩa với các nút, cạnh và các thuộc tính để biểu diễn và lưu trữ dữ liệu. Một khái niệm quan trọng của hệ thống là biểu đồ ( hoặc các cạnh hoặc mối quan hệ ), liên quan trực tiếp đến các mục dữ liệu trong kho lưu trữ một tập hợp các nút dữ liệu và các cạnh biểu thị mối quan hẹ giữa các nút. Các mối quan hệ cho phép dữ liệu trong cửa hàng được liên kết trực tiếp với nhau và trong nhiều trường hợp được truy xuất với một thao thác. Cơ sở dữ liệu đồ thị giữ mối quan hệ giữa các dữ liệu là một ưu tiên. Truy vấn các mối quan hệ trong cơ sở dữ liệu đồ thị là nhanh vì chúng được lưu trữ vĩnh viễn trong chính cơ sở dữ liệu. Mối quan hệ có thể được trực quan hóa bằng cách sử dụng cơ sở dữ liệu đồ thị, làm cho nó hữu ích cho dữ liệu được kết nối với nhau.

Nguồn gốc: Cơ sở dữ liệu đồ thị mô tả dữ liệu khi nó được xem theo khái niệm. Điều này được thực hiện bằng cách chuyển dữ liệu vào các nút và các mối quan hệ của nó thành các cạnh.

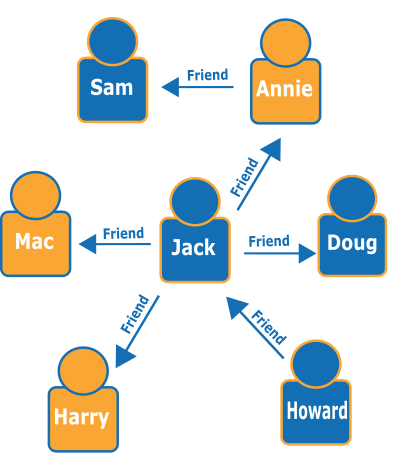
Cơ sở dữ liệu đồ thị sử dụng các cấu trúc đồ thị với các node, các cạnh và các thuộc tính để biểu diễn và lưu trữ dữ liệu.

### 1.2.2. Một đồ thị chứa các node và các quan hệ

“Đồ thị - lưu trữ dữ liệu trong - Các node - có – Các mối quan hệ”

Đồ thị đơn giản nhất có thể là đồ thị chỉ có 1 node duy nhất, lưu các thuộc tính biểu diễn giá trị của nó.

Một node có thể bắt đầu chỉ với 1 thuộc tính Property và sau đó phát triển thành hàng triệu properties. Tại một số điểm, nó phân tán dữ liệu thành nhiều node khác, tổ chức thành các mối quan hệ rõ ràng.



*Hình 1.2.2. Tổ chức của một cơ sở dữ liệu*

### 1.2.3 Các mối quan hệ tổ chức nên đồ thị

“Các node – được tổ chức bỏi - > các mối quan hệ- có các - > thuộc tính”

Các mối quan hệ này tổ chức các node thành các trúc tùy tiện, cho phép đồ thị tương tự như 1 List, Tree, Map hay các compound Entity, mà có thể kết nối thành những cấu trúc phức tạp hơn, đa dạng về kết nối với nhau.

### 1.2.4 Truy vấn cơ sở dữ liệu đồ thị bằng cách duyệt đồ thị

“Một phép duyệt – điều hướng -> một đồ thị, nó xác định -> đường đi - theo thứ tự- các node”

Chúng ta truy vấn 1 cơ sở dữ liệu đồ thi sử dụng cách duyệt đồ thị, xuất phát từ node bắt đầu đến các node kế cận theo 1 giải thuật, được mô hình dưới đây:

Traveral

Graph

Paths

Algorithm

Relationships

Nodes

Navigates

Expresses

Organize

Records data in

Identifies

Order

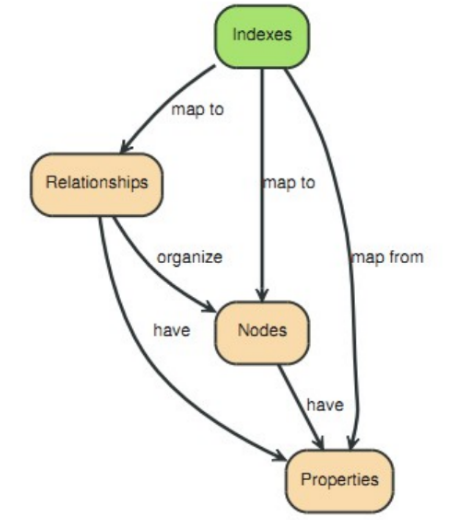
Records data in

*Hình 1.2.4. Mô hình truy vấn cơ sở dữ liệu đồ thị*

### 1.2.5. Indexing

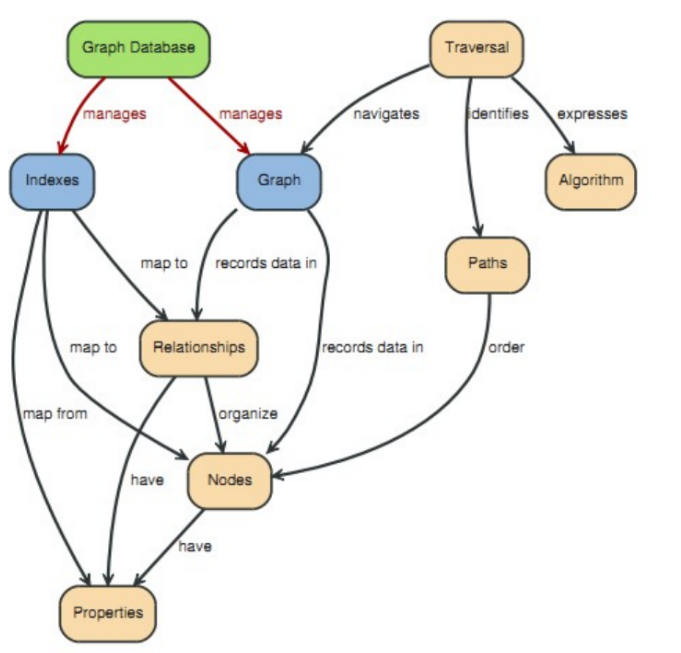
“Một chỉ mục – được nối từ -> các thuộc tính – đến -> các node hoặc các mối quân hệ”

Mục đích của việc đánh chỉ mục (indexing) là tìm kiếm được nhanh chóng và chính xác trong cơ sở dữ liệu đồ thị.



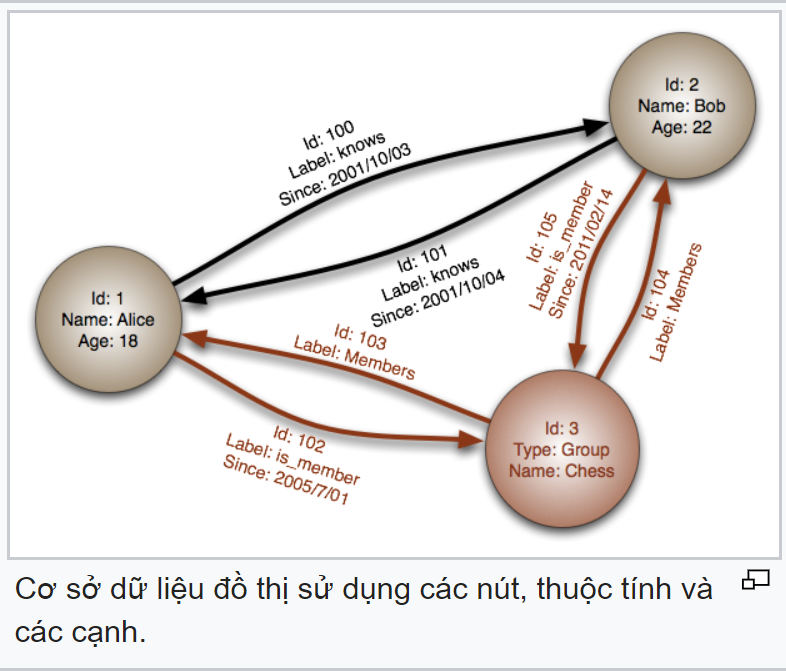
*Hình 1.2.5. Quan hệ giữa chỉ mục và các thành phần trong cơ sở dữ liệu đồ thị*

Như vậy, ta có cái nhìn tổng quan về cơ sở dữ liệu đồ thị:



*Hình 1.2.5.1. Tổng quan về cơ sở dữ liệu đồ thi*

Ta thấy, một cơ sở dữ liệu đồ thị sẽ quản lý các chỉ mục và đồ thị gồm các node và các cạnh. Các node trong đồ thị lưu trữ thông tin của chúng thông qua các node và các mối quan hệ giữa các node. Các chỉ mục sẽ ánh xạ đến từng node , từng mối quan hệ giữa các node, và đến các thuộc tính của các node và các mối quan hệ đó. Để thực hiện truy vấn trong 1 cơ sở dữ liệu đồ thị, ta thực hiện một phép duyệt đồ thị dựa trên một giải thuật để xác định đường đi theo thứ tự của các node.

****

*Hình 1.2.5.2. Ví dụ về cơ sở dữ liệu đồ thị*

Trong ví dụ này, đồ thị của cơ sở dữ liệu gồm 3 node N = {Alice, Bob, Group , với tập thuộc tính gồm PV = {id, name, age, type}. Các cạnh cũng có tập các thuộc tính sau: PE = {id, label, since}. Đây là một đồ thị có hướng với những thuộc tính của các cạnh nối 2 đỉnh là khác nhau. Trong cơ sở dữ liệu này, ta có thể thực hiện truy vấn các thông tin, như: tìm mối quan hệ giữa người có tên là Alice với người có tên là Bob, tìm các cá nhân có quan hệ là “is\_member” của group có tên là “Chess”….

### 1.2.6. Tính chất

Cơ sở dữ liệu đồ thị là một công cụ mạnh mẽ cho các truy vấn giống như đồ thị. Ví dụ, tính toán đường đi ngắn nhất giữa hai nút trong biểu đồ. Các truy vấn giống như biểu đồ khác có thể được thực hiện trên cơ sở dữ liệu biểu đồ theo cách tự nhiên (ví dụ: tính toán đường kính của biểu đồ hoặc phát hiện cộng đồng).

Đồ thị rất linh hoạt, có nghĩa là nó cho phép người dùng chèn dữ liệu mới vào biểu đồ hiện có mà không làm mất chức năng ứng dụng. Không cần người thiết kế cơ sở dữ liệu để lên kế hoạch chi tiết cho các trường hợp sử dụng trong tương lai của cơ sở dữ liệu.

# CHƯƠNG 2: GIỚI THIỆU VỀ NEO4J

Neo4j là một hệ thống quản lý cơ sở dữ liệu đồ thị được phát triển bởi Neo4j, Inc. Được các nhà phát triển của nó mô tả là cơ sở dữ liệu giao dịch ACID - compliant với lưu trữ và xử lý đồ thi gốc, Neo4j là cơ sở dữ liệu đồ thị phổ biến nhất theo xếp hạng DB-Engines, và cơ sở dữ liệu phổ biến 22 nói chung.

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.

## 2.1. LỊCH SỬ

Phiên bản 1.0 được phát hành vào tháng 2 năm 2010.

Neo4j phiên bản 2.0 được phát hành vào tháng 12 năm 2013.

Neo4j phiên bản 3.0 được phát hành vào tháng 4 năm 2016.

Vào tháng 11 năm 2016 Neo4j đã đảm bảo thành công 36$ triệu trong Tài trợ Series E do One Peak Partners và Morgan Stanley Expansion Capital dẫn đầu, với sự tham gia của các nhà đầu tư khác bao gồm Creandum, Eight Roads và Greenbridge Partners.

## 2.2. CẤP PHÉP VÀ PHIÊN BẢN

Neo4j có hai phiên bản: Cộng đồng và Doanh nghiệp. Nó được cấp phép kép: [GPL v3](https://en.wikipedia.org/wiki/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.

**2.3. ĐẶC ĐIỂM**

Biểu diễn mô hình dữ liệu hướng đồ thị một các trực quan.

Quản lý lưu trữ trên đĩa cứng, hoàn toàn tối ưu cho việc lưu trữ các cấu trúc đồ thị với hiệu năng lớn nhất và có khả năng mở rộng tối đa.

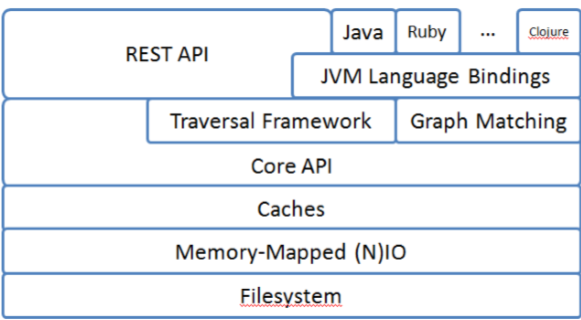
Neo4j có thể xử lý các đồ thị với hàng tỉ các node/ các mối quan hệ/ các thuộc tính trên 1 máy tính và có thể mở rộng ra trên nhiều máy khác.

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).

## 2.4. KIẾN TRÚC CỦA NEO4J

Kiến trúc của Neo4j được mô tả theo hình sau:



*Hình 2.4. 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 đến bất kỳ 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. - CoreAPI : 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,…

## 2.5. NODES

Đơn vị chủ chốt hình thành nên 1 đồ thị là các node và các mối quan hệ. Trong Neo4j, các node và các mối quan hệ có thể có các thuộc tính liên quan.

Các node thường được dùng để biểu diễn các thuộc tính, phụ thuộc vào miền của các mối quan hệ mà có thể sử dụng cho nhiều mục đích khác nhau.

Node

Relationships

Properties

Can have

Can have

*Hình 2.5.1. Node, relationships và properties*

Với một đồ thị đơn giản chỉ chứa 1 node và có thể chỉ chứa 1 thuộc tính.

Name: Thu

*Hình 2.5.2. Ví dụ về một đồ thị đơn giản với 1 node và 1 properties*

Trong ví dụ trên, đồ thị có 1 thuộc tính tên là “name”, với giá trị là “ Thu”.

## 2.6. RELATIONSHIPS

Trong cơ sở dữ liệu đồ thị, các mối quan hệ giữa các node (relationships) là một phần khóa của cơ sở dữ liệu. Chúng cho phép tìm kiếm các dữ liệu có liên quan với nhau. Cũng giống như các node, các relationships cũng có tập các thuộc tính.

A relationships

Start node

End node

Relationships type

Properties

Name

has a has a has a can have

uniquely identified by

*Hình 2.6. Relationships và các quan hệ liên quan*

Một relationship kết nối 2 node với nhau, gồm node bắt đầu và node kết thúc. Mỗi relationship có loại quan hệ “relationship type”, mỗi “relationship type” này được xác định bởi một định danh duy nhất. Một relationship có tập các thuộc tính với các giá trị của các thuộc tính này.

Một relationship luôn có hướng. Chúng biểu diễn quan hệ đầu vào của 1 node, và quan hệ đầu ra của 1 node, dùng cho phép duyệt đồ thị:

Relationship

End node

Start node

*Hình 2.6.1. Một relationships giữa start node và end node*

Incoming relationshipoutgoing

Relationship

Node

*Hình 2.6.2. Quan hệ vào và quan hệ ra của 1 node*

Một node cũng có thể tự có quan hệ với chính nó:

Node

*Hình 2.6.3. Quan hệ trên 1 node*

Mặc dù đồ thị luôn có một hướng, nhưng chúng ta có thể bỏ qua hướng của đồ thị trong ứng dụng nhỉ của chúng ta.

Để nâng cao hơn phép duyệt đồ thị tất cả các relationships có một “relationship type” có thể được xem là nhãn của một relationship.

William

Oscar

Alice

Maja

follows follows

Follows

Blocks

*Hình 2.6.4. Ví dụ một mạng xã hội*

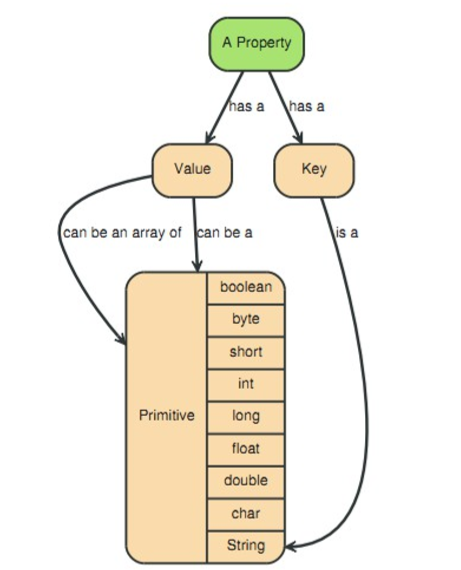
Trong mạng xã hội trên, có 2 “relationship type” là “follows” và “blocks”, thể hiện mối quan hệ giữa các node : Maja, Alice, Oscar, và William.

## 2.7. PROPERTIES

Cả node và relationships đều có tập 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 properties có value = null nghĩa là không tồn tại key đó trong tập properties của nodes hay relationships.



*Hình 2.7. Biểu diễn 1 Properties*

## 2.8. 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

A Path

Start Node

Relationships

End Node

Node

Has an

Has a can containt one

Or more

Accompanied by a

*Hình 2.8.1. Path*

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

Node 1

*Hình 2.8.2. Đường đi đến chính node đó bằng 0*

Đường đi của một node đến node khác được biểu diễn như sau:

Node 1

Relationships 1

Node 2

*Hình 2.8.3 Đường đi từ node 1 đến node 2 có quan hệ relatonship 1*

## 2.9. DUYỆT ĐỒ THỊ

Duyệt một đồ thị nghĩa là đi thăm các node của đồ thị đó, dựa trên các relationships giữa các node theo một số quy tắc. Trong một số trường, chỉ có một đồ thị con được thăm.

Có 2 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.

## 2.10. ƯU ĐIỂM NEO4J

* Thời gian phát triển nhanh và tính linh hoạt
* Nhanh chóng thêm chức năng mới mà không ảnh hưởng đến việc triển khai hiện có
* Không phải cố ép dữ liệu mới vào một mô hình dữ liệu cứng nhắc

## 2.11. NHƯỢC ĐIỂM NEO4J

* Lưu trữ graph bằng Neo4j thì toàn bộ graph phải nằm trên một máy duy nhất. (Đây là nhược điểm rất lớn của Neo4j, điều này khiến việc lưu trữ một big graph bao gồm rất nhiều đỉnh và cạnh bằng Neo4j là rất khó khan)

## 2.12 SO SÁNH NEO4J VỚI CƠ SỞ DỮ LIỆU QUAN HỆ (RDBMD)

|  |  |  |
| --- | --- | --- |
|  | Cơ sở dữ liệu quan hệ | Neo4j, Cơ sở dữ liệu đồ thị gốc |
| Lưu trữ dữ liệu | Lưu trữ trong các bảng cố điịnh, được xác điịnh trước với các hàng và cột có dữ liệu được kết nối thường rời rạc giữa các bảng, làm tê liệt hiệu quả truy vấn | Cấu trúc lưu trữ đồ thị 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 cơ sở dữ liệu phải được phát triển với các trình tạo mô hình và được dịch từ mô hình logic sang mô hình vật lý. Vì các loại dữ liệu và nguồn phải được biết trước, nên bất kỳ thay đổi nào cũng cần nhiều tuần ngừng hoạt động để thực hiện. | Mô hình dữ liệu linh hoạt, “thân thiện với bảng trắng” không cí sự phù hợp giữa mô hình logic và mô hình vật lý. Các kiểu dữ liệu và nguồn có thể thêm được thêm hoặc thay đổi bất cứ lúc nào, dẫn đến thời gian phát triển ngắn hơn đáng kể và lặp lại nhanh nhẹn thực sự. |
| 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 trể bằng không và hiệu suất thời gian thực, bất kể số lượng hoặc độ sâu của mối quan hệ. |
| Ngôn ngữ truy vấn | SQL: một ngôn ngữ truy vấn tăng độ phức tạp với số lượng THAM GIA cần thiết cho các truy vấn dữ liệu được kết nối. | 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 | Hỗ trợ giao dịch ACID được yêu cầu bởi các ứng dụng doanh nghiệp cho dữ liệu phù hợp và đáng tin cậy. | Giữ lại các giao dịch ACID cho 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 mô | Quy mô thông qua nhân rộng và mở rộng kiến ​​trúc là có thể nhưng tốn kém.Mối quan hệ dữ liệu phức tạp không được thu hoạch ở quy mô. | Mô hình đồ thị vốn đã chia 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 tốn kém về mua, sử dụng năng lượng và thời gian quản lý. | D Dữ liệu và các mối quan hệ được lưu trữ nguyên bản cùng với cải thiện hiệu suất khi độ phức tạp 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. |

## 2.13. 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 không cung cấp quyền kiểm soát ở cấp thiết kế. Đặt áp lực không đáng có lên cấp độ ứng dụng để nắm bắt và giải quyết vấn đề. | Mô hình dữ liệu linh hoạt, "thân thiện với bảng trắng" cho phép kiểm soát chi tiết kiến ​​trúc dữ liệu.Mô hình dữ liệu trực quan giúp giảm bớt giao tiếp giữa các nhà phát triển, kiến ​​trúc sư và DBA. |
| 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 | Kiến trúc quy mô giả định truy cập liên tục vào ngày càng nhiều phần cứng hàng hóa mà không tính chi phí năng lượng, lỗ hổng mạng và các rủi ro khác.. | Dữ liệu và các mối quan hệ được lưu trữ nguyên bản cùng với cải thiện hiệu suất khi độ phức tạp và quy mô tăng lên. Điều này dẫn đến việc hợp nhất máy chủ và sử dụng phần cứng cực kỳ hiệu quả. |

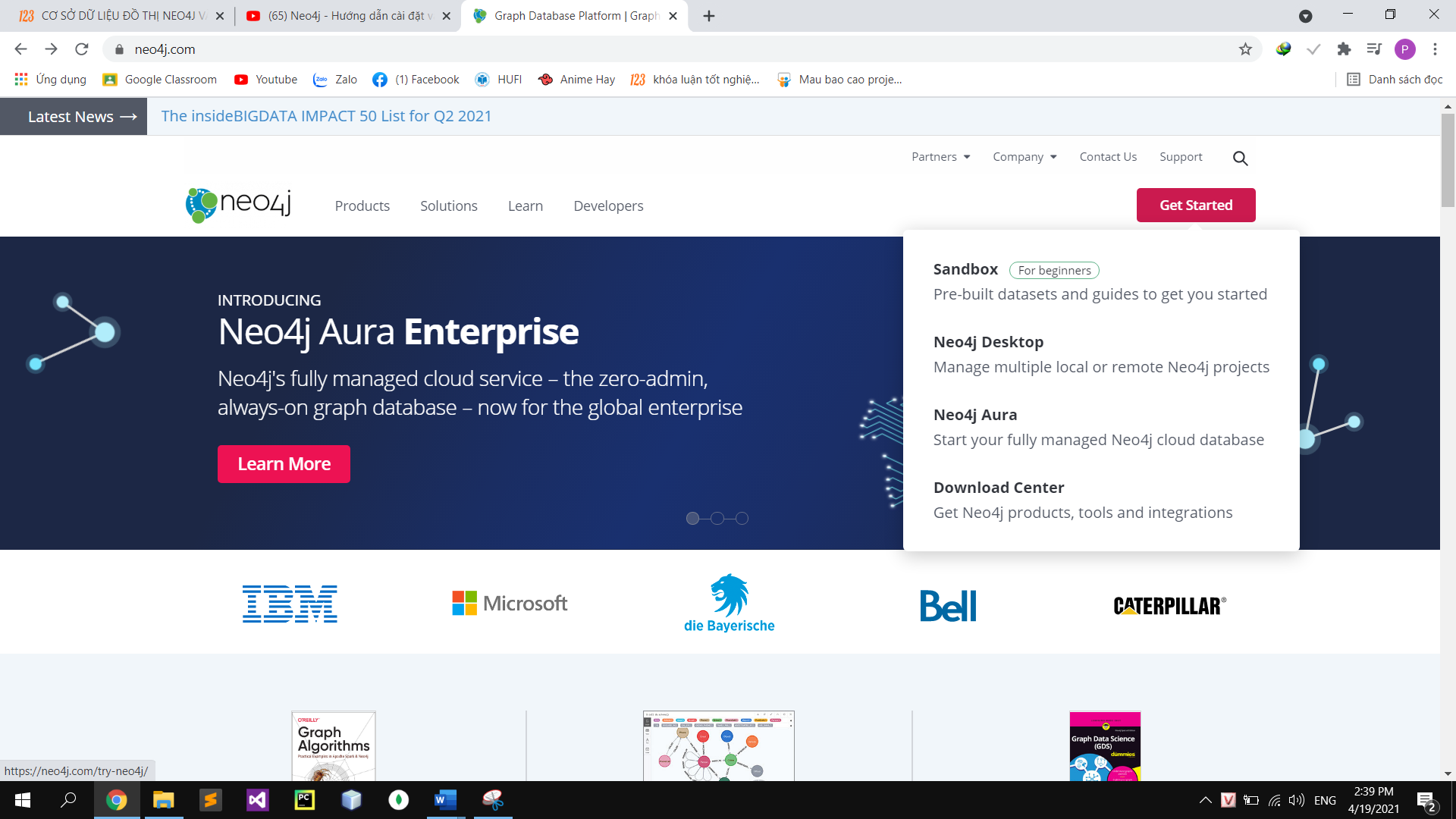
# CHƯƠNG 3. CÀI ĐẶT NEO4J

## 3.1. CÀI ĐẶT NEO4J

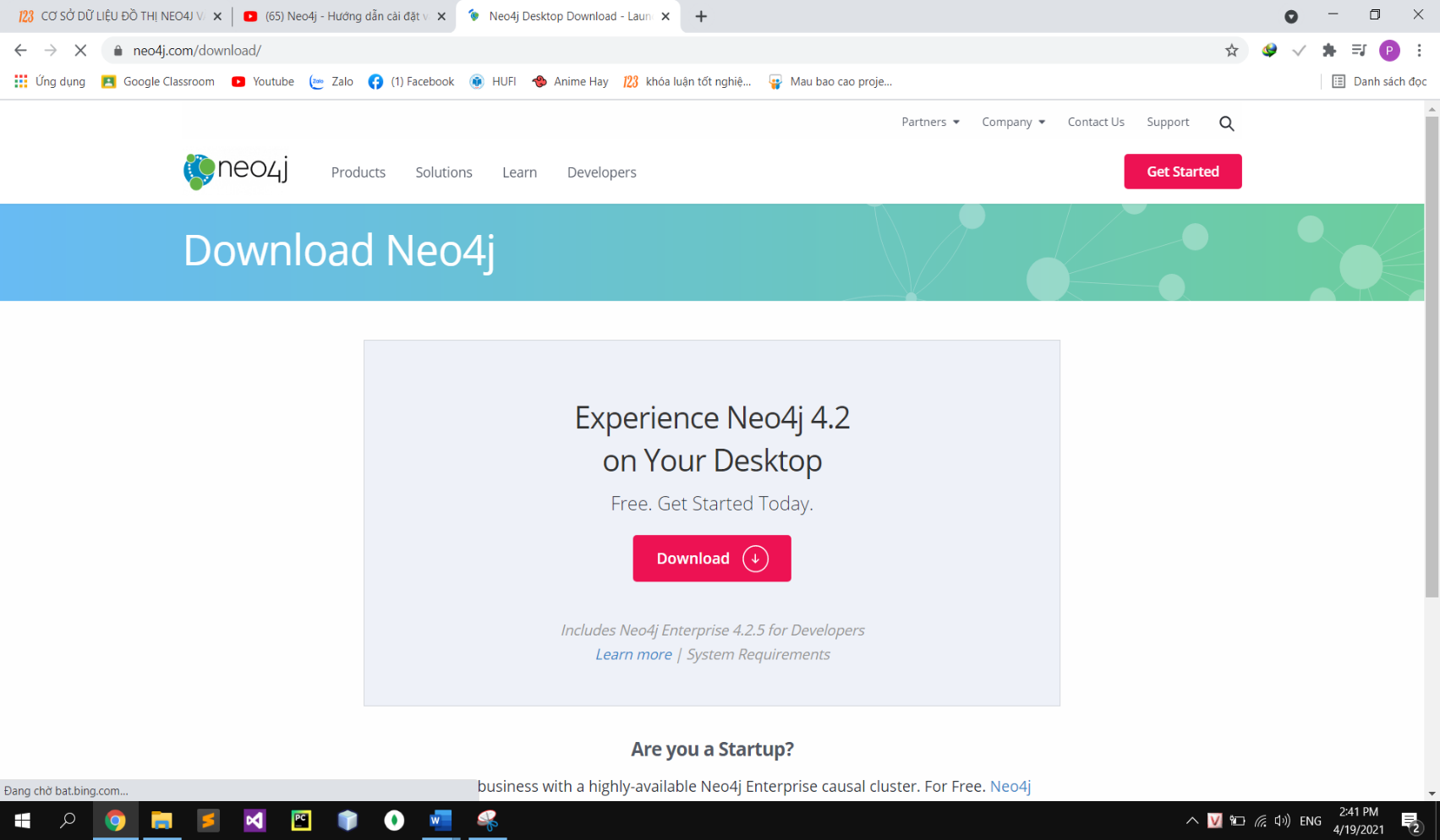
Có hai phiên bản để chúng ta có thể lựa chọn đó là bản *Community* dành cho nghiên cứu, học tập những project nhỏ (không cần license) bản *Enterprise* là bản full option dành cho những project lớn và yêu cầu nhiều tác vụ xử lý nhanh (cần license). (<http://neo4j.com/download/>)

Địa chỉ của neo4j: [https://neo4j.com](https://neo4j.com/)

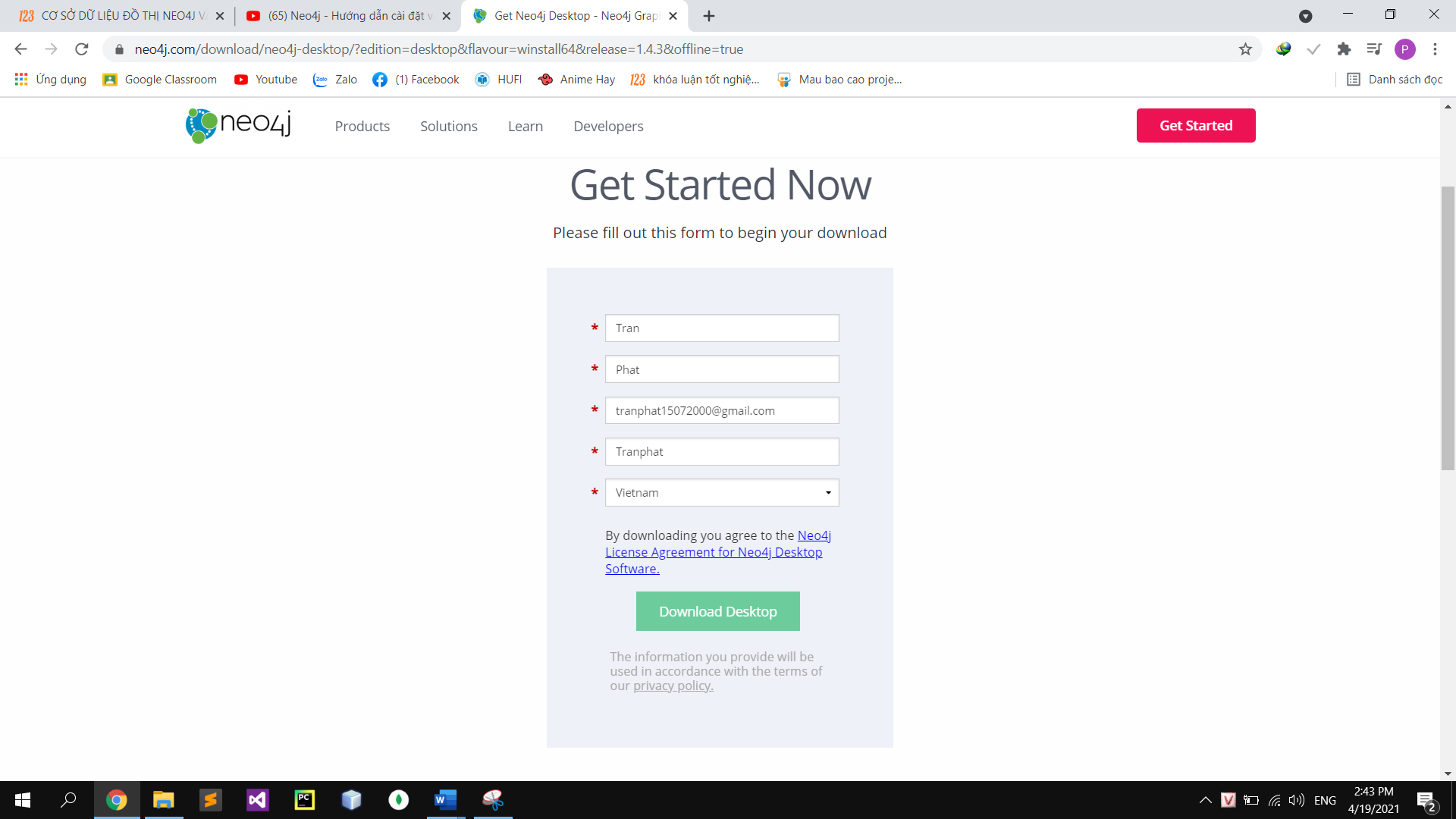
Bước 1: Chọn Get Strated -> Neo4J Desktop



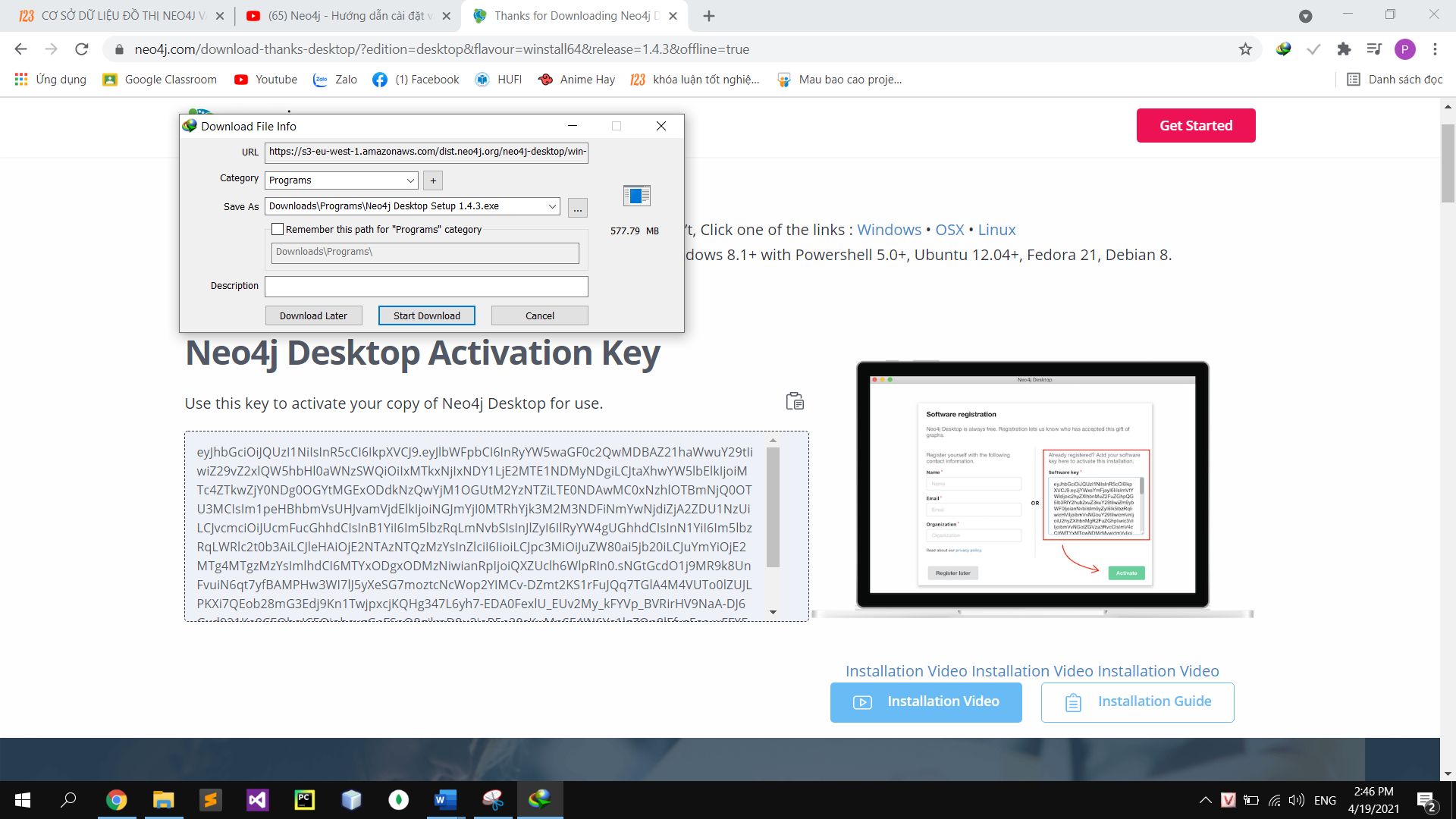
Bước 2: chọn Dowload



Bước 3: Điền thông tin và click chọn Download Desktop

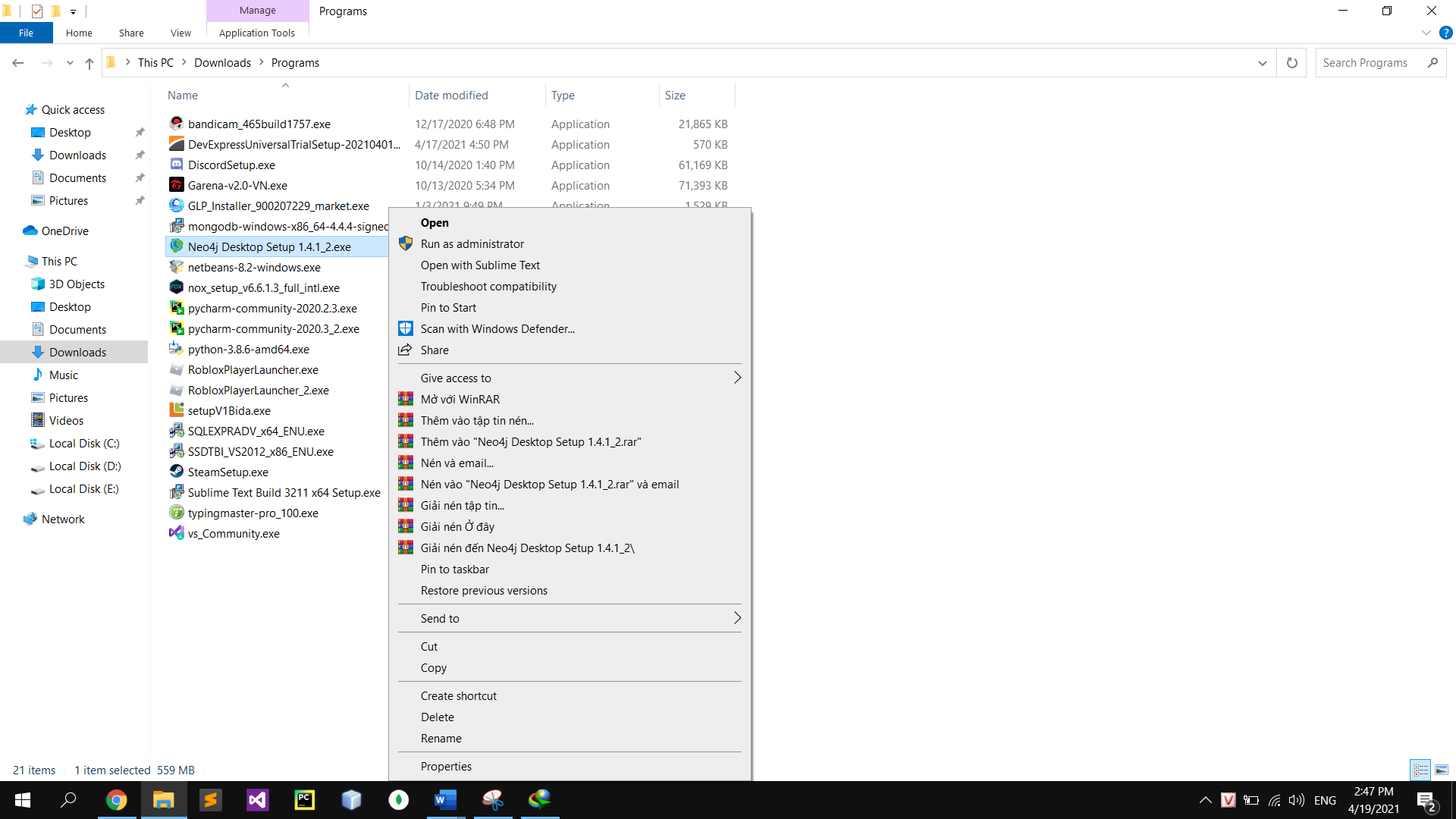


Bước 4: Bấm tải và copy Key

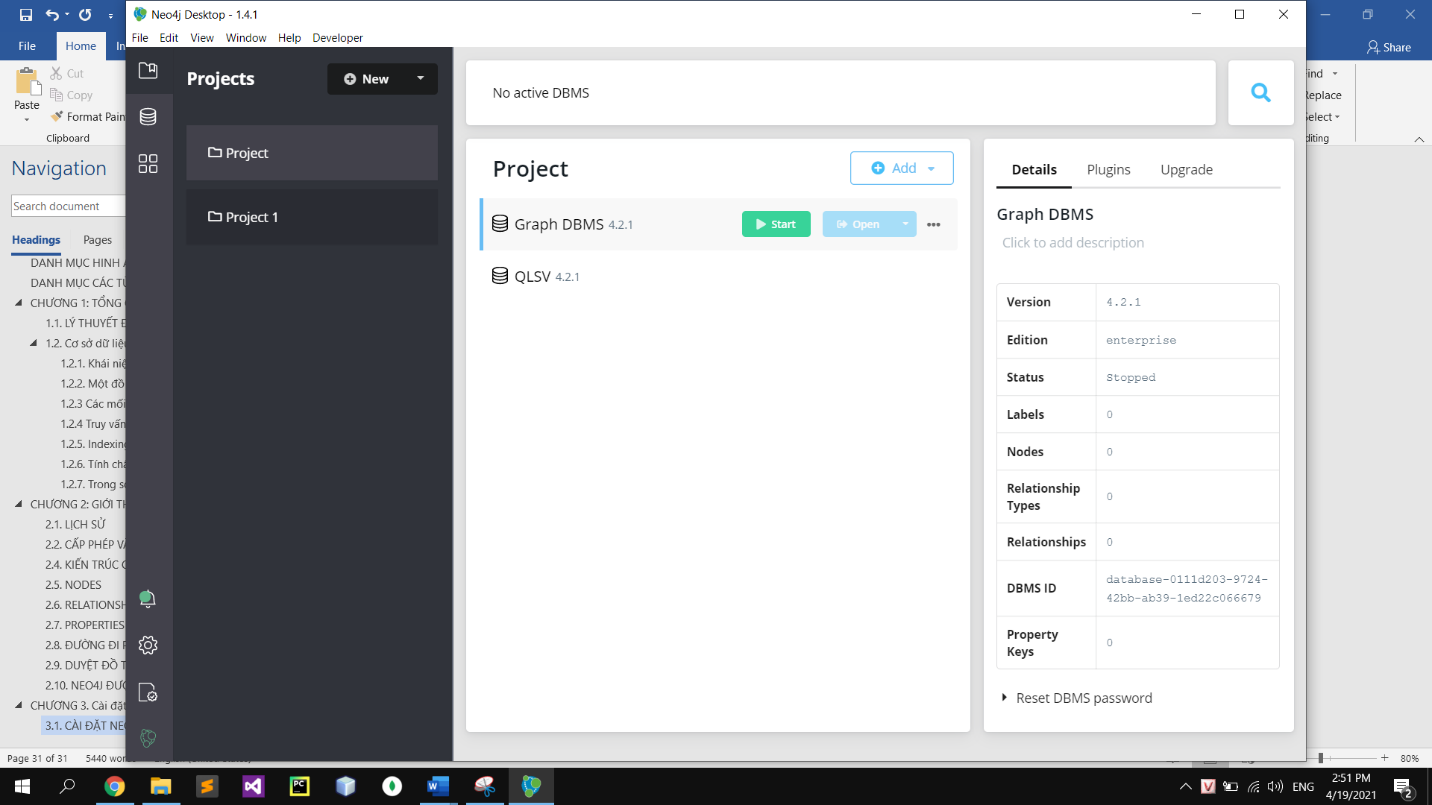


Tải xuống của bạn sẽ bắt đầu tự động trong một vài phút.

Sau đó vô Download và tiến hành cài đặt



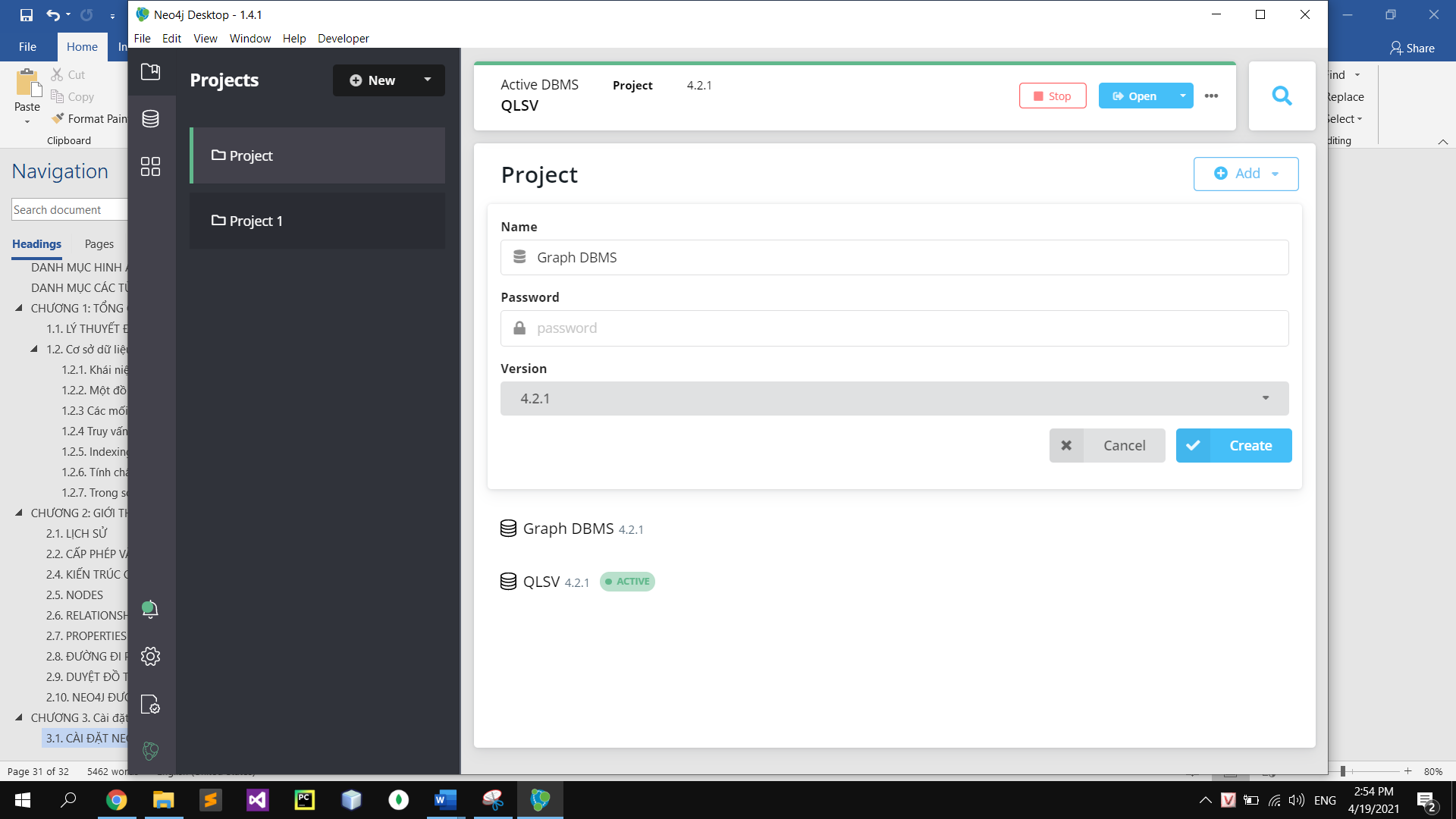
Sau quá trình cài đặt đây là giao diện của neo4j Dekstop



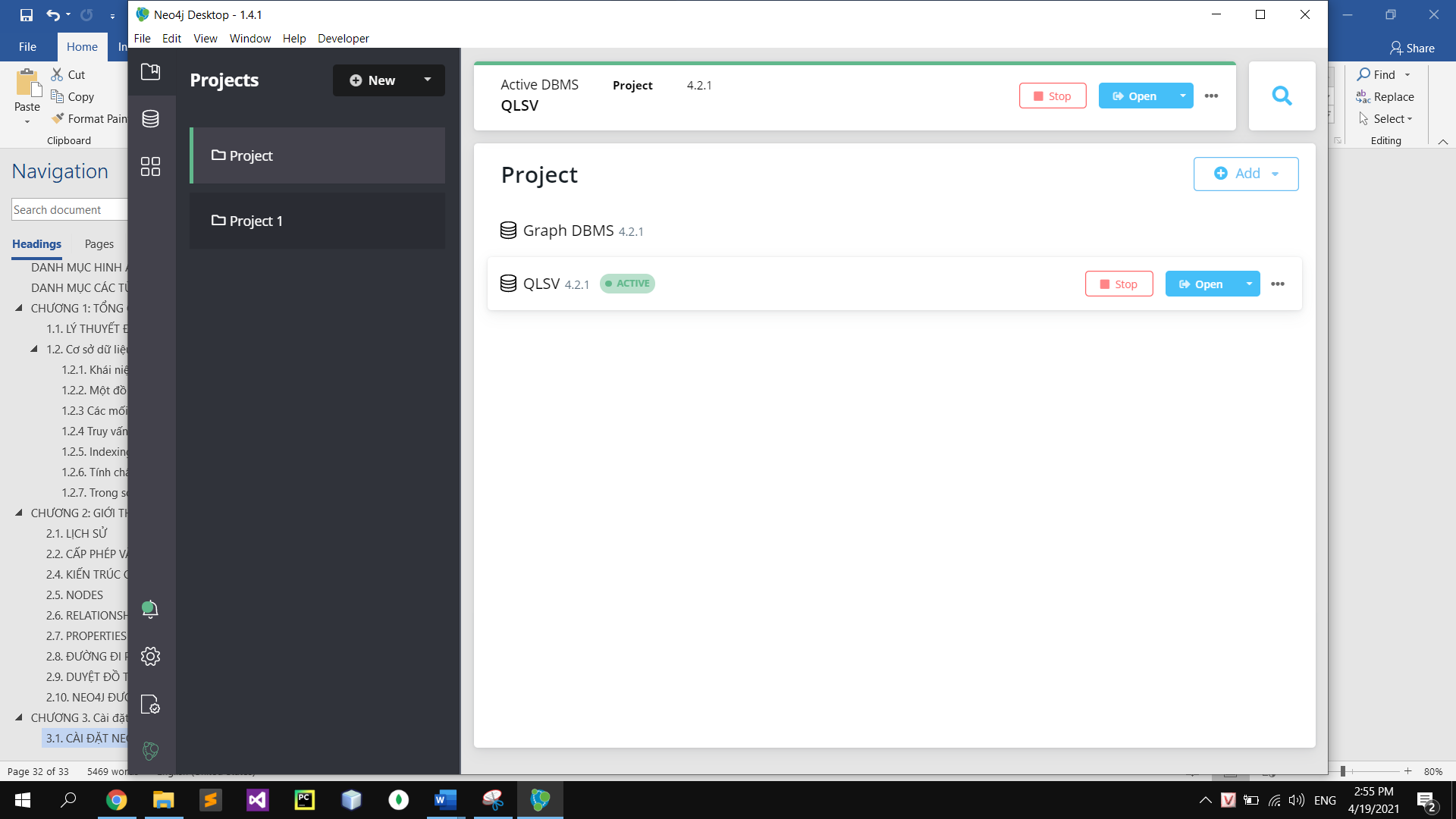
Để tạo 1 Project mới chọn Add -> Local DBMS



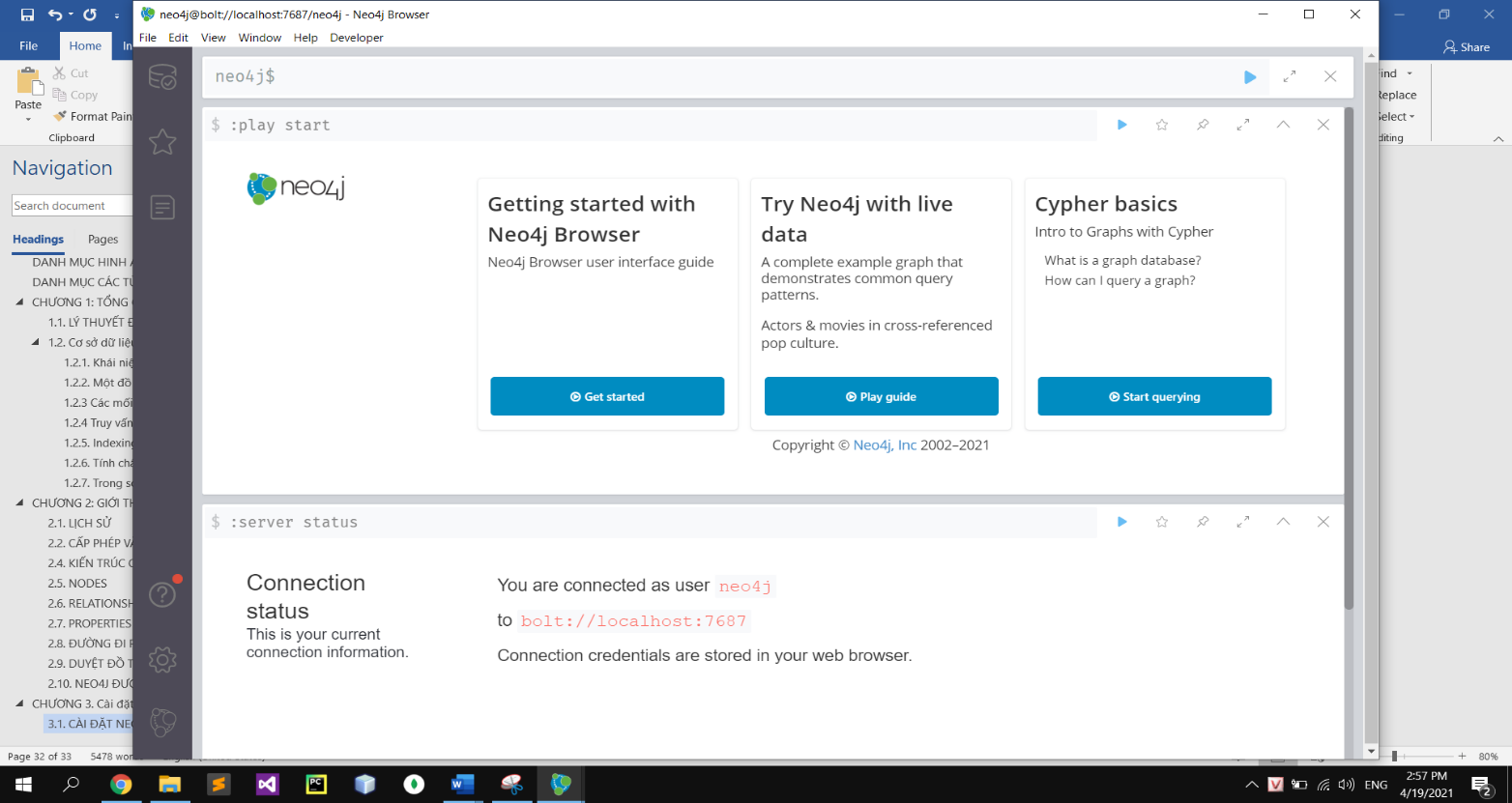
Đặt tên mà mật khẩu -> Create



Sau đó bấm start và đợi sau đó open



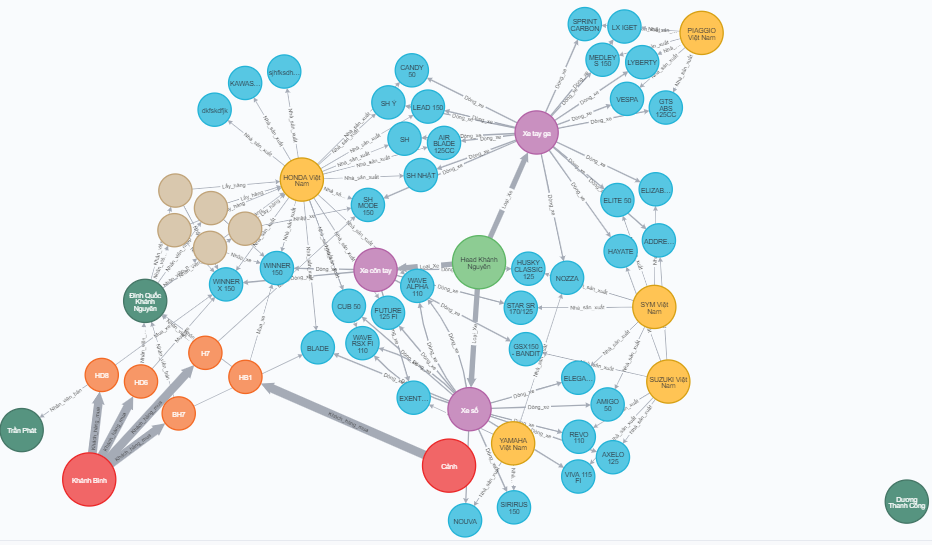
🡪Như vậy là chúng ta đã sẵn sàn để thiết kế dữ liệu đồ thị rồi.



Trình duyệt Neo4j là một công cụ để các nhà phát triển khám phá cơ sở dữ liệu Neo4j của họ, thực hiện các truy vấn Cypher và xem kết quả ở dạng bảng hoặc biểu đồ. Bạn thậm chí có thể sử dụng Trình duyệt để:

* [Nhập dữ liệu](https://neo4j.com/developer/guide-importing-data-and-etl/)
* Gọi [các thủ tục do người dùng định nghĩa](https://neo4j.com/docs/developer-manual/current/procedures/) trong Java
* [Hồ sơ truy vấn](https://neo4j.com/download-thanks-desktop/docs/developer-manual/current/cypher/#how-do-i-profile-a-query) , nhìn vào kế hoạch thực hiện với **EXPLAIN**và **PROFILE**.

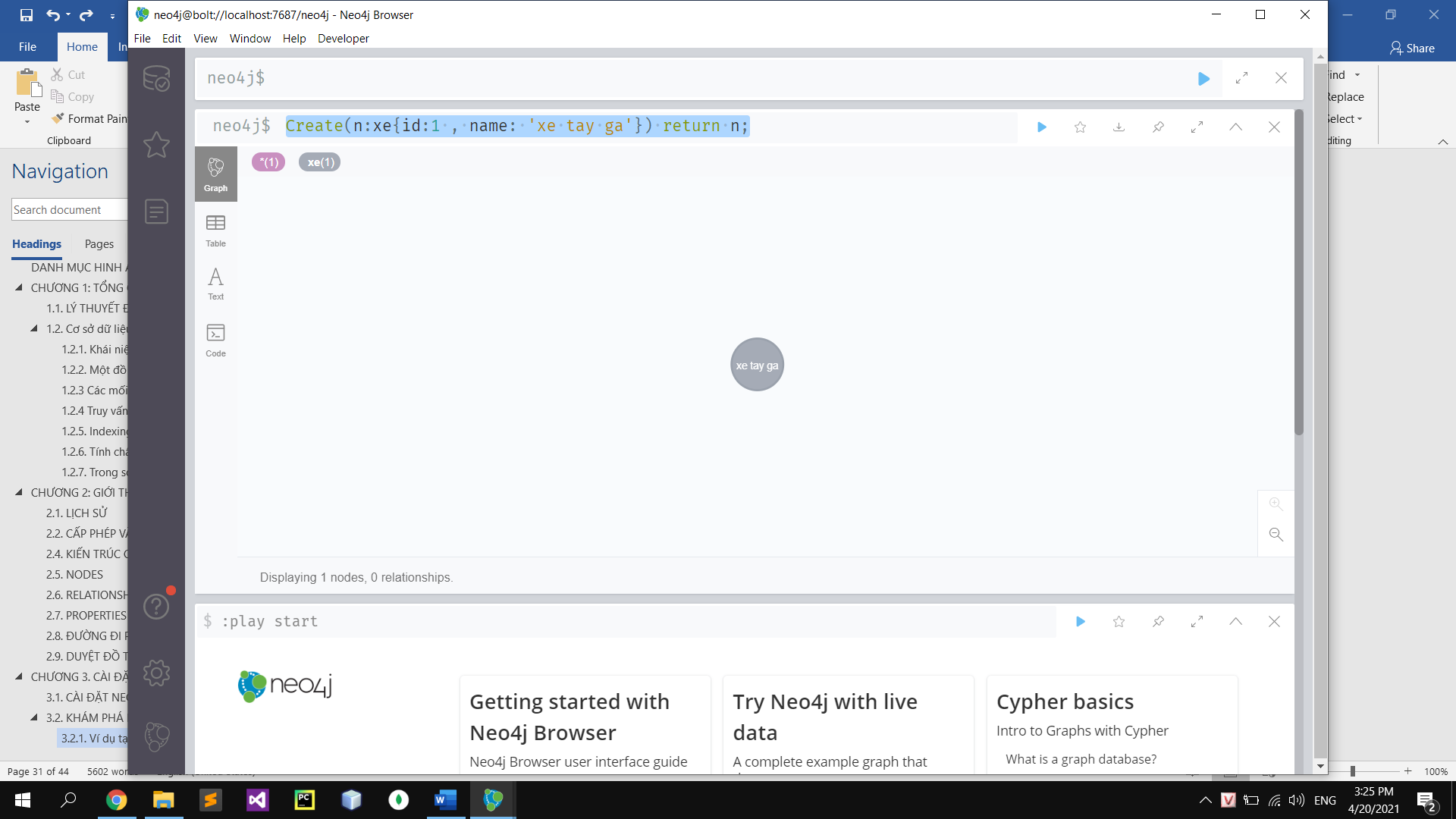
## 3.2. KHÁM PHÁ BỘ DỮ LIỆU



*Hình 3.2. Cơ sở dữ liệu đồ thị*

### 3.2.1. Ví dụ tạo các node tạo khóa cho nó

Create(n:xe{id:1 , name: 'xe tay ga'}) return n;





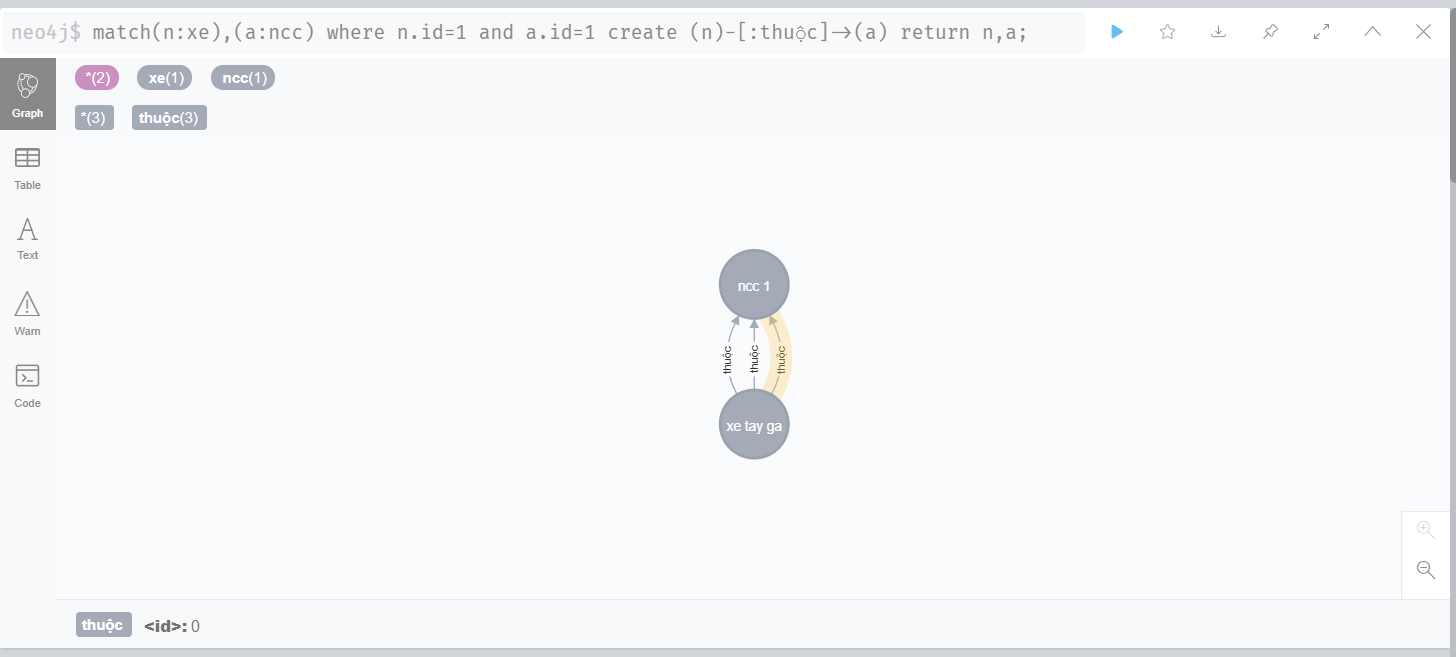
### 3.2.2. Ví dụ tạo các relationship cho các đối tượng

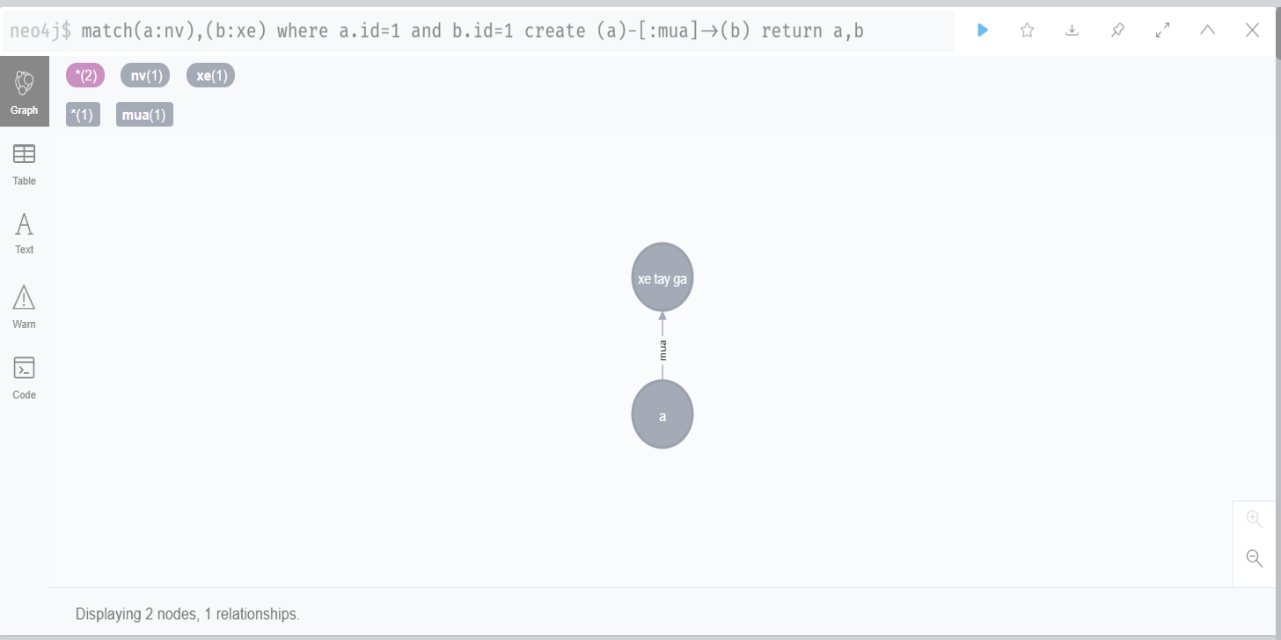
Match(n:xe),(a:ncc) *//lấy ra 2 lớp đối tượng là sách và tác giả*

Where n.id = 1 and a.id = 1 *// điều kiện để hệ thống trỏ vào đúng xe và ncc cần set relationship*

Create (n)-[r:thuộc]->(a) *//tạo mối quan hệ từ điều kiện*

Return a,n *//trả về 2 node a và n*





### 3.2.3. Ví dụ về truy vấn trong Neo4J

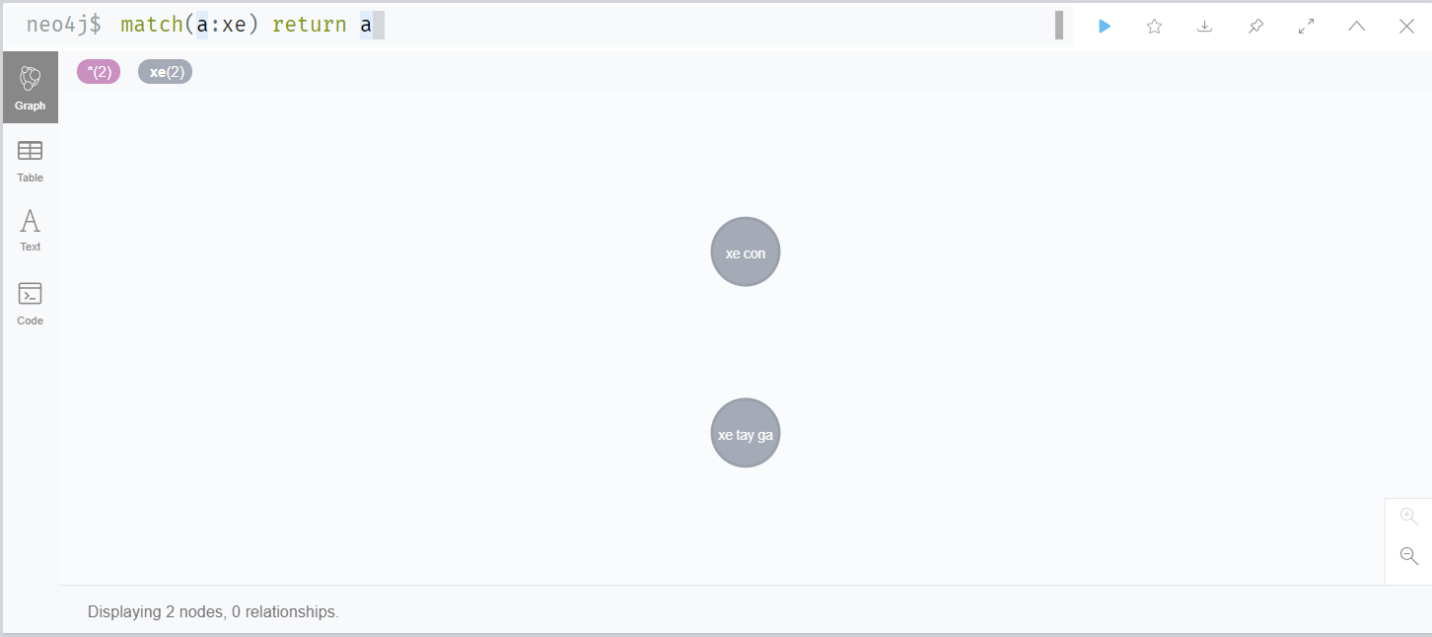
Một số truy vấn đơn giản với cơ sở đồ thị ta vừa tạo:

Để hiển thị ra các quyển sách ta thực hiện câu lệnh sau

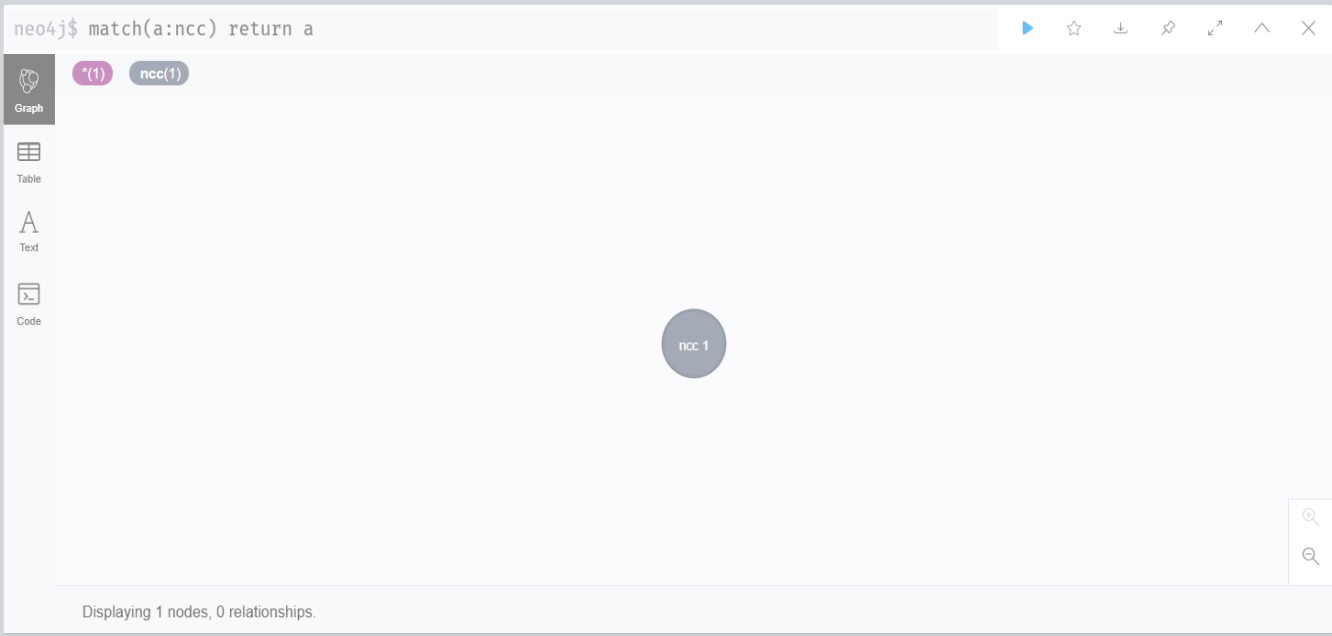
Match(a:xe) *//tìm những node xe*

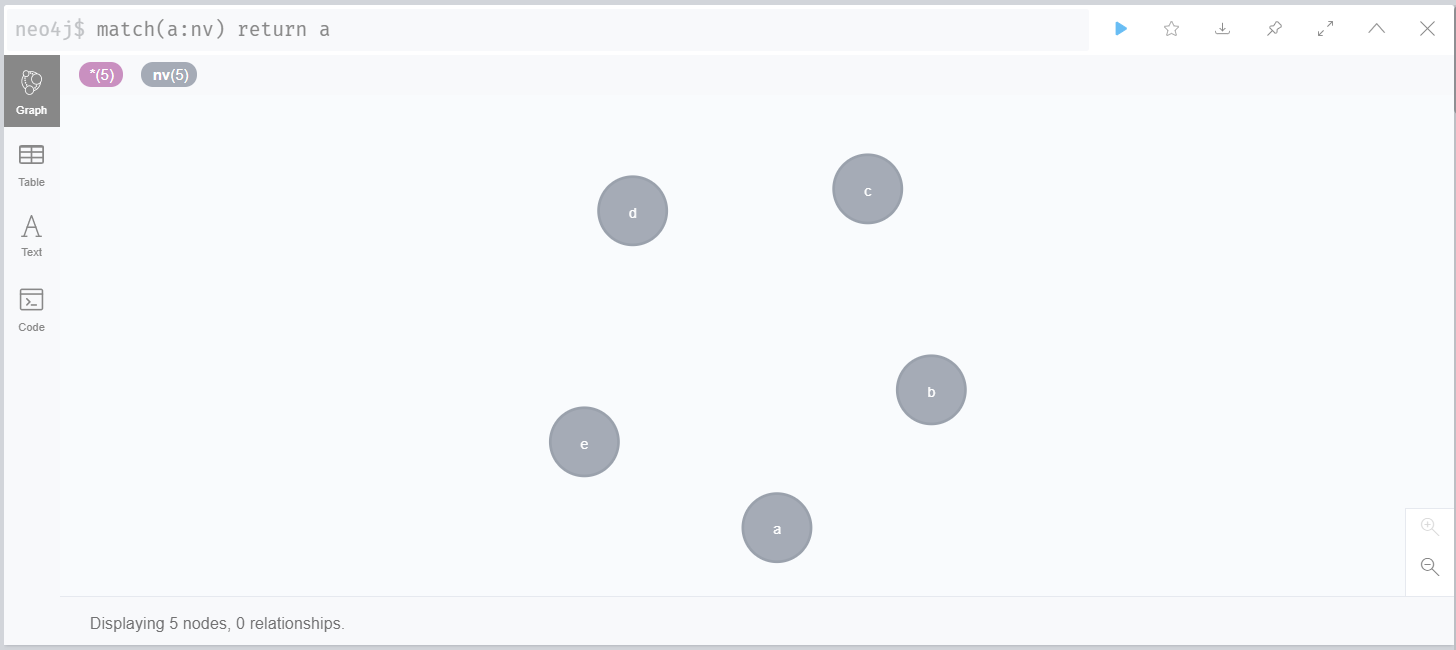
Return a *//trả về, hiển thị ra những loại xe*

Kết quả hiển thị



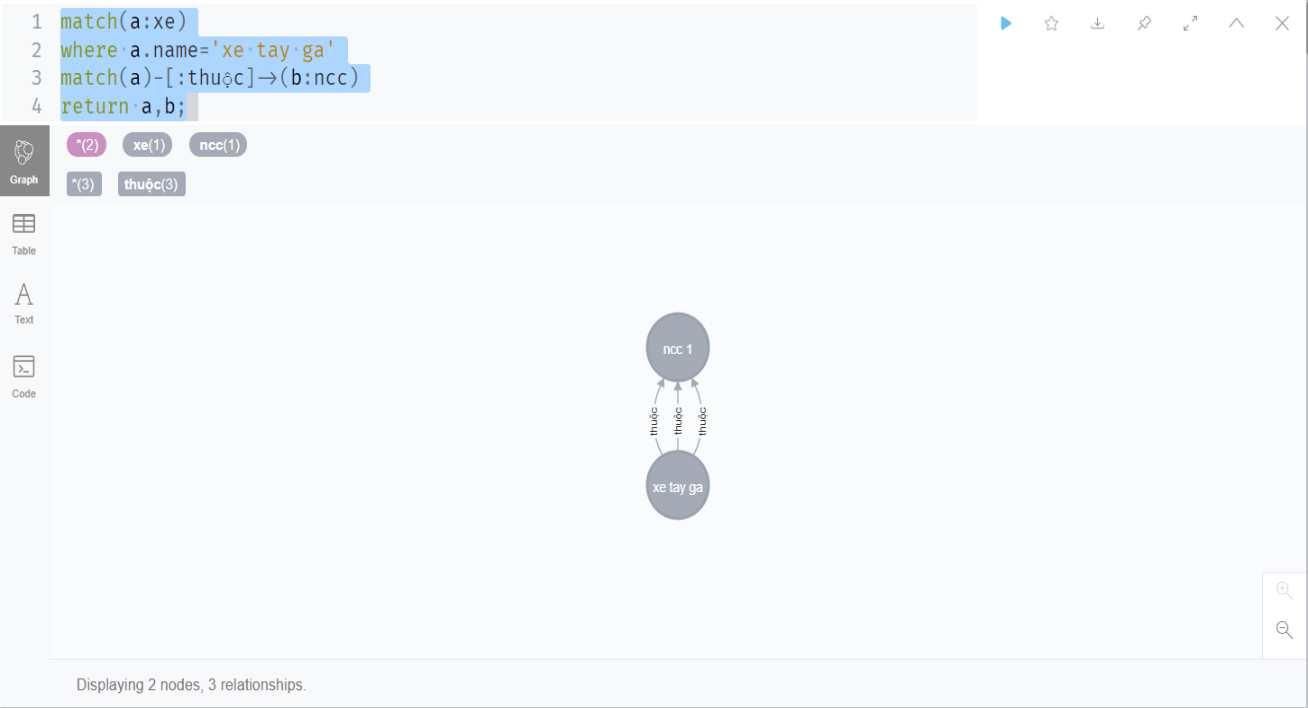
Và một số bảng tương tự

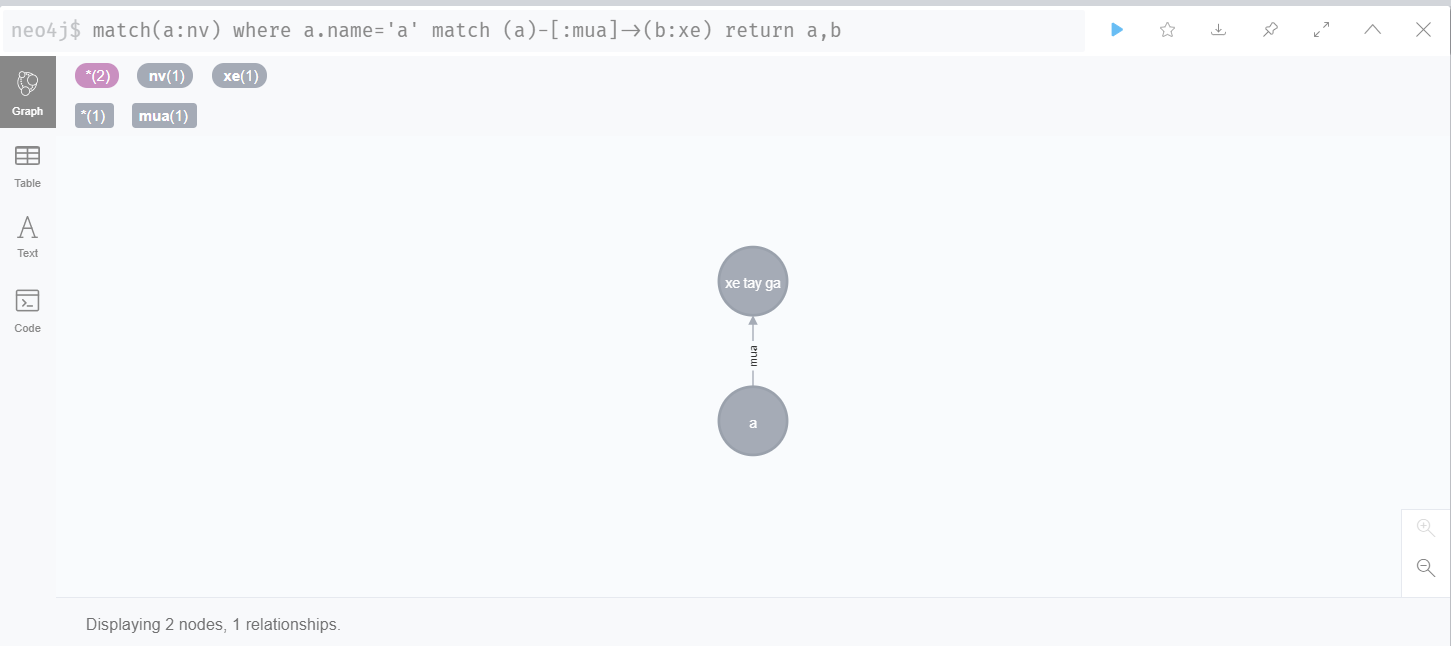




### 3.2.4. Ví dụ về truy vấn where

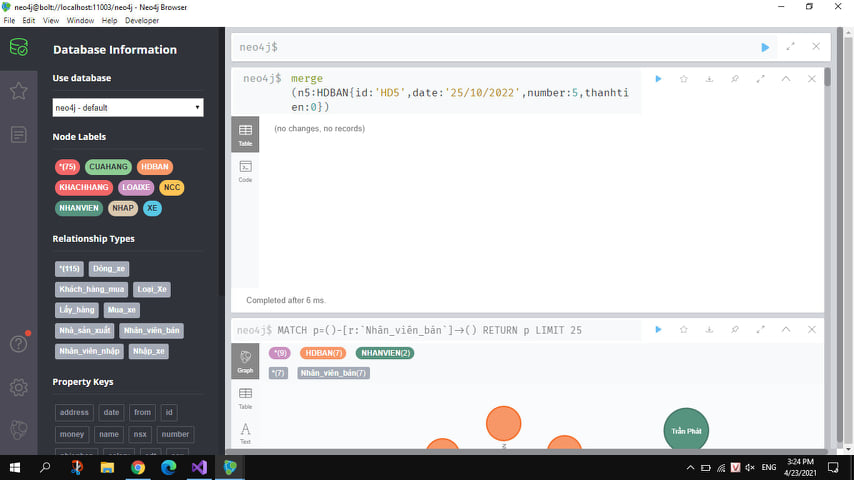
Match(a:xe) //tìm lớp xeWhere a.name = 'xe tay ga' //điều kiện xe có nane là xe tay gamatch (a)-[r:thuộc]->(b:ncc) //tìm tất cả những mối quan hệ thuộc từ xe tay gareturn a,b

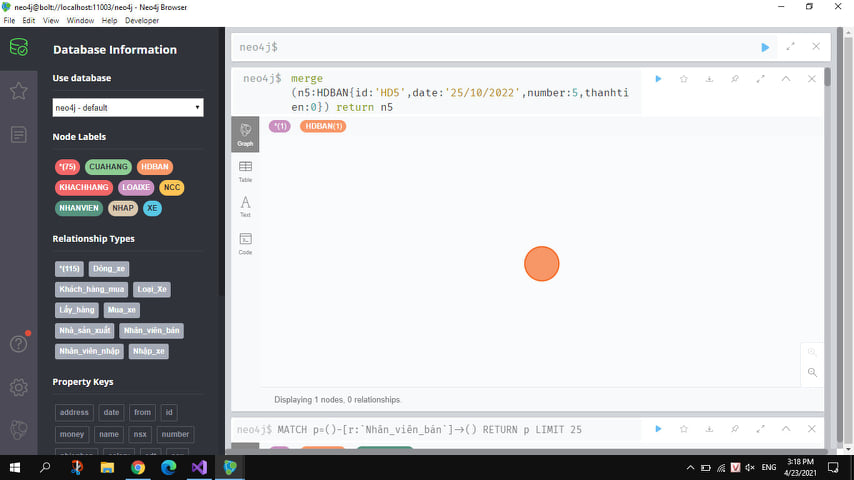


Ví dụ tương tự

Trong quá trình tìm hiểu thì chúng em thấy giữa MERGE với CREATE

+ Merge sẽ tốt hơn vì trong quá trình thêm 1 dữ liệu lần đầu thì sẽ được thêm nhưng nếu thêm lần 2 với cùng một kiễu dữ liệu đó thì merge sẽ không được thêm vào, nó sẽ giúp ta tối ưu hơn create vì nếu thêm bằng Create thì dễ bị trùng code khi sữ dụng





## 3.3. THƯ VIỆN APOC NEO4J

APOC là viết tắt của **A**wesome **P**rocedures **o**n **C**ypher. Trước khi phát hành APOC, các nhà phát triển cần viết các thủ tục và chức năng của riêng họ cho các chức năng phổ biến mà Cypher hoặc cơ sở dữ liệu Neo4j chưa triển khai để được hỗ trợ. Mỗi nhà phát triển có thể viết phiên bản của riêng mình cho các chức năng này, gây ra nhiều sự trùng lặp.

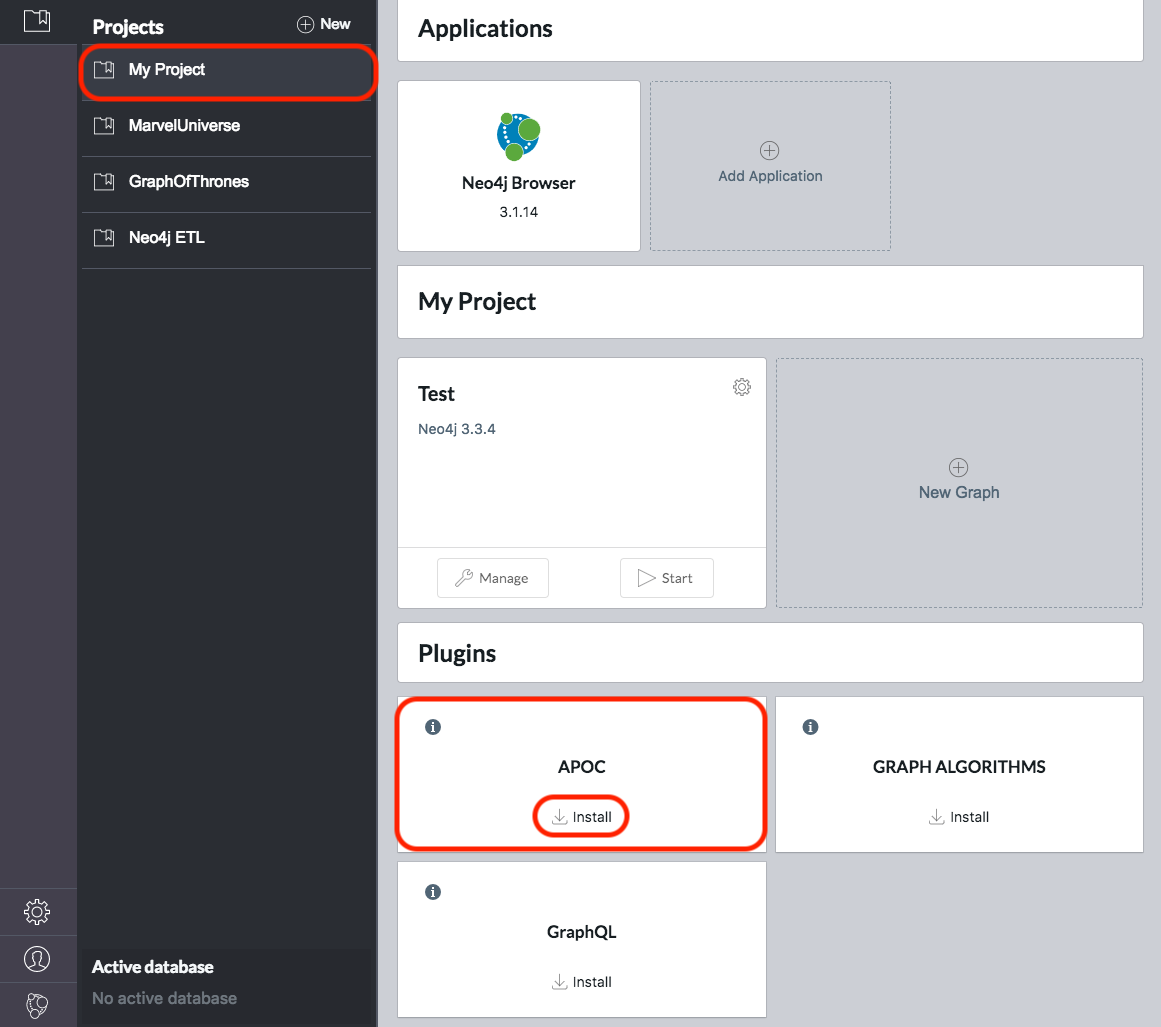
Vì vậy, một trong những nhà phát triển Neo4j của chúng tôi đã tạo thư viện APOC như một thư viện tiện ích tiêu chuẩn cho các thủ tục và chức năng phổ biến. Điều này cho phép các nhà phát triển trên các nền tảng và ngành công nghiệp sử dụng thư viện tiêu chuẩn cho các thủ tục chung và chỉ viết chức năng của riêng họ cho logic nghiệp vụ và nhu cầu của từng trường hợp sử dụng.

Thư viện APOC được cho là thư viện mở rộng lớn nhất và được sử dụng rộng rãi nhất cho Neo4j. Nó bao gồm hơn 450 quy trình tiêu chuẩn, cung cấp chức năng cho các tiện ích, chuyển đổi, cập nhật biểu đồ, v.v. Chúng được hỗ trợ tốt và rất dễ chạy dưới dạng các hàm riêng biệt hoặc đưa vào các truy vấn Cypher. Trước khi bạn bắt đầu viết các hàm adhoc cho ứng dụng của mình, hãy nhớ kiểm tra APOC trước để xem nó có tồn tại ở đó không!

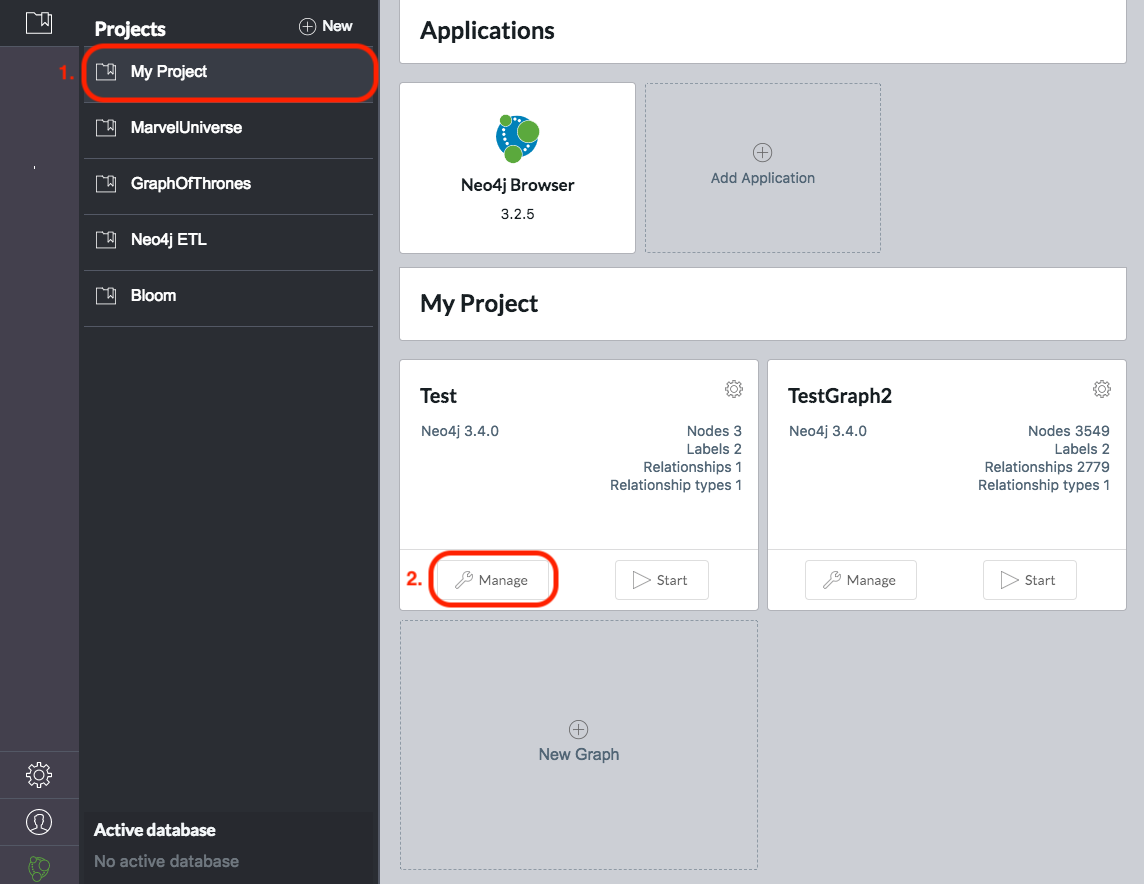
* APOC giống như là là Functions và Procedures của cơ sở dữ liệu SQL

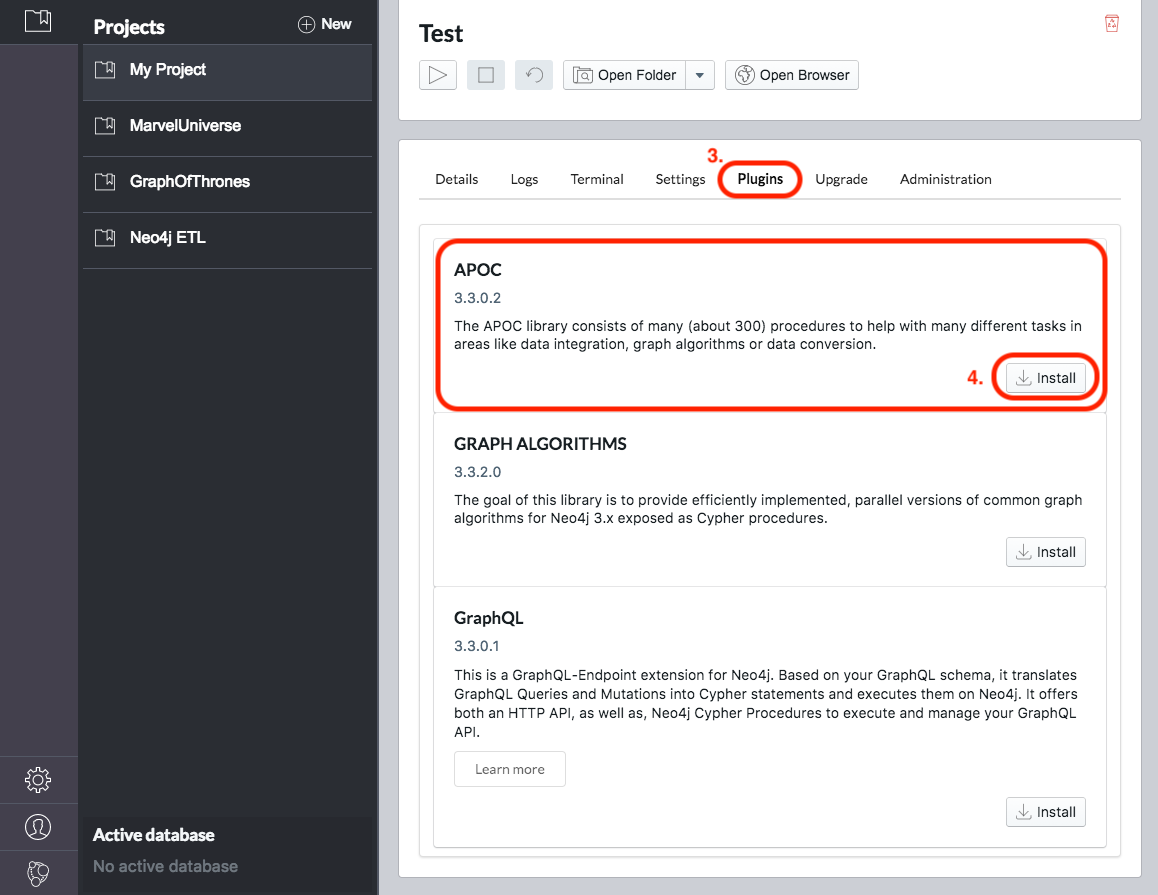
### 3.3.1. Cài đặt thư viện APOC

Ở cấp độ dự án:



Ở cấp độ cơ sở dữ liệu:





### 3.3.2. Các Apoc được sử dụng trong đề tài

**- Apoc.do.case ([điều kiện, truy vấn, điều kiện, truy vấn,…], elseQuery: '', params: {})** giá trị lợi nhuận - được cung cấp danh sách các cặp truy vấn có điều kiện / viết, thực hiện truy vấn được liên kết với điều kiện đầu tiên đánh giá thành true (hoặc truy vấn khác nếu không có câu nào đúng) với các tham số đã cho

Match (n:XE)

CALL apoc.do.case( [ n.year>=1999 and n.year<2019 ,'set n.phienban="Phiên bản củ" return n', n.year>=2019 and n.year<2021 ,'set n.phienban="Phiên bản mới" return n'], set n.phienban="Phiên bản bình thường" return n',{n:n})

YIELD value

Return value.n.name as Name , value.n.year as Year , value.n.phienban as Phienban

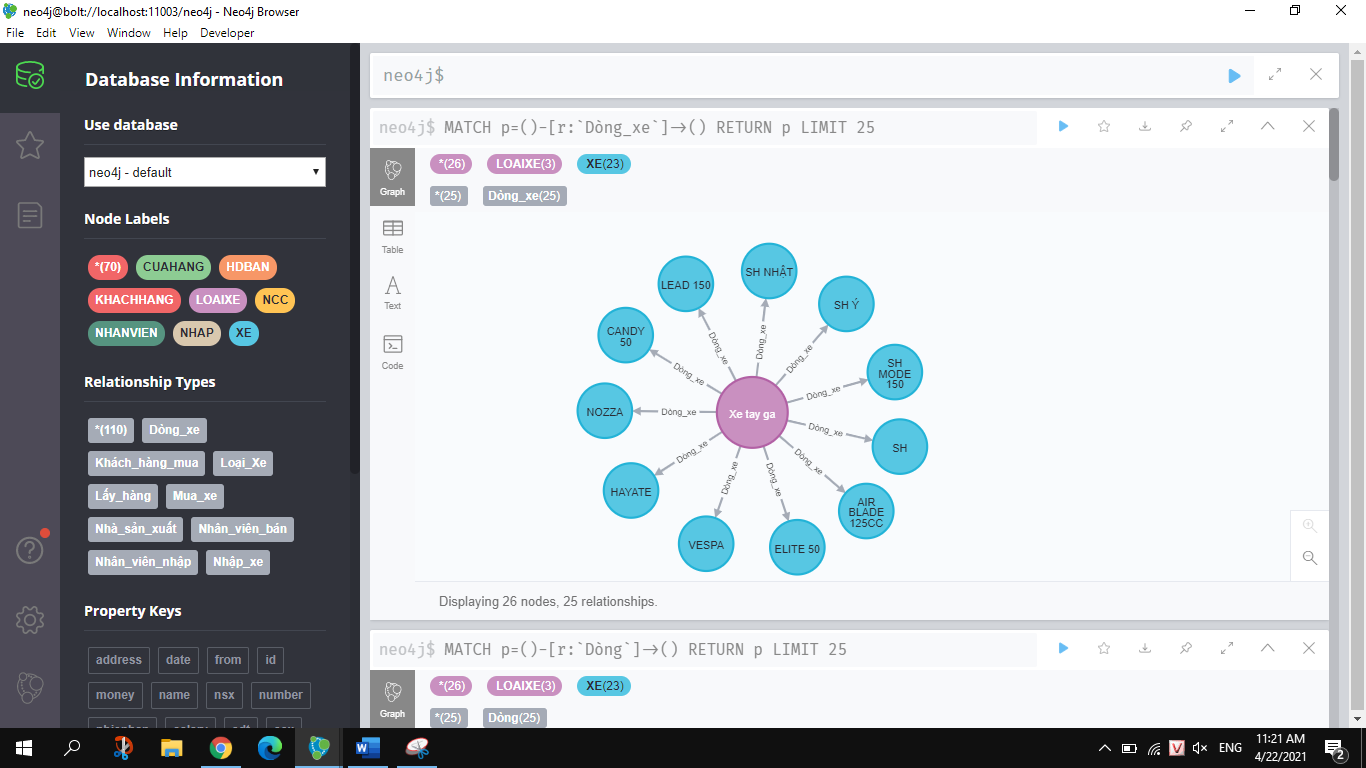
=> Câu ví dụ trên dùng để xác định phiên bản của xe bằng cách dùng apoc.do.case để kiểm tra giá trị theo năm để biết được xe đó thuộc phiên bản nào thẻo phiên bản đã cho

**- Apoc.search.multiSearchRednced.** Thực hiện tìm kiếm song song trên nhiều chỉ mục trả về biểu diễn rút gọn của các nút được tìm thấy: id nút, nhãn và các thuộc tính được tìm kiếm. apoc.search.multiSearchRednced (bản đồ nhãn và thuộc tính sẽ được tìm kiếm, toán tử: EXACT | CONTAINS | STARTS WITH | ENDS WITH, searchValue). Nhiều kết quả tìm kiếm cho cùng một nút được hợp nhất thành một bản ghi.

CALL apoc.search.multiSearchReduced( {CUAHANG: "name", XE: "name"}, "CONTAINS", "H" );

**- Apoc.refactor.setType (rel, 'NEW-TYPE')** thay đổi kiểu quan hệ

MATCH (n:LOAIXE)-[rel:dòng\_xe]->(m:XE) CALL apoc.refactor.setType(rel, 'Dòng\_xe') YIELD input, output RETURN input, outpu



**- Apoc.refactor.to** mối quan hệ chuyển hướng apoc.refactor.to (rel, endNode) để sử dụng nút cuối mới

MATCH (n:HDBAN{id:'HD13'})-[rel:Nhân\_viên\_bán]->()

MATCH (jennifer:NHANVIEN{id:'NV2'})

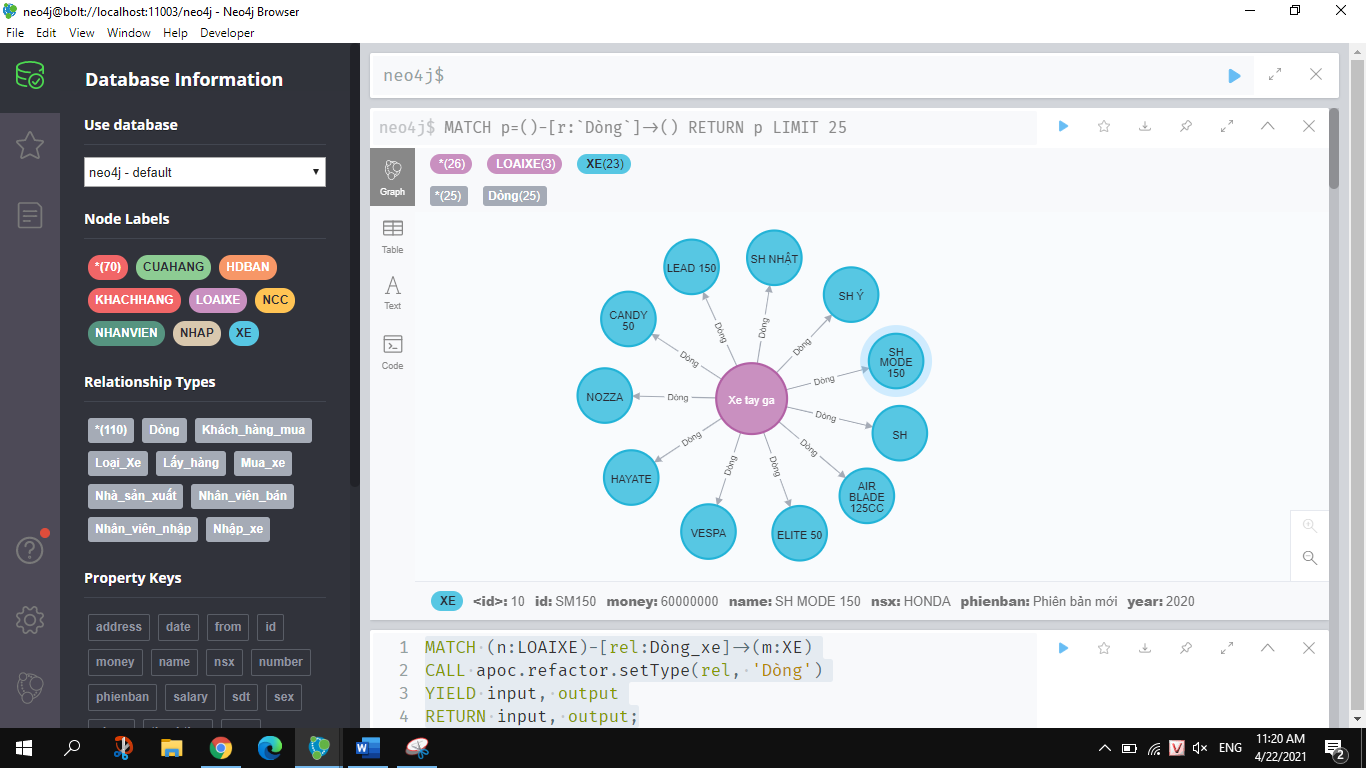
CALL apoc.refactor.to(rel, jennifer)

YIELD input, output

RETURN input, output;

**- Apoc.refactor.invert apoc.refactor.invert (rel)** đảo ngược hướng quan hệ

MATCH ()-[rel:Dòng\_xe]->() CALL apoc.refactor.invert(rel) YIELD input, output RETURN input, output;



# CHƯƠNG 4: GIỚI THIỆU ĐỀ TÀI

## 4.1. LÝ DO CHỌN ĐỀ TÀI

Việc lưu trữ quản lý dữ liệu bằng sổ sách, giấy tờ làm tốn nhiều thời gian, công sức và tiền của. Trong khi đó, công nghệ thông tin đã và đang phát triển mạnh mẽ. Các ứng dụng của công nghệ thông tin đã mang lại nhiều hiệu quả cho công tác lưu trữ, chế bản, đặc biệt là công tác quản lý dữ liệu. Các đơn vị hành chính, các công ty, cửa hàng rất cần một hệ thống có đủ khả năng giải quyết, đáp ứng các nhu cầu về việc quản lý hồ sơ, giấy tờ hay các dữ liệu khác. Qua quá trình học tập và nghiên cứu tại trường ĐH Công Nghiệp thực phẩm TP HCM, kết hợp với thời gian tìm hiểu thực tế tại.. Em nhận thấy cần phải có một hệ thống dành riêng cho Cửa hàng trong công tác quản lý để đáp ứng phần nào những nhu cầu trên, quản lý được các thông tin nhập, xuất, gói gọn dữ liệu. Vì vậy em đã chọn đề tài: “Quản lý cửa hàng xe máy” để làm đề tài nghiên cứu khoa học của nhóm em. Chương trình này được xây dựng bằng ngôn ngữ lập trình Visual C Sharp (C#) kết hợp với cơ sở dữ liệu Neo4j Dekstop

## 4.2. MỤC ĐÍCH VÀ NHIỆM VỤ NGHIÊN CỨU

Mục đích nghiên cứu: Nghiên cứu các giải pháp xây dựng phần mềm quản lý cửa hàng xe máy sao cho phù hợp và sát thực tế, giúp cho nhà quản lý thực hiện tốt các công việc cần thiết đồng thời tiết kiệm được công sức và tiền của.

Nhiệm vụ nghiên cứu:

- Làm sáng tỏ cơ sở lý luận của đề tài.

- Tìm hiểu thực trạng kinh doanh của cửa hàng.

- Đưa ra các giải pháp, xây dựng phần mềm quản lý cửa hàng bán xe máy.

- Thực hiện việc ứng dụng tin học và công tác quản lý trong các đơn vị hành chính, các công ty, cửa hàng.

## 4.3. ĐỐI TƯỢNG VÀ PHẠM VI NGHIÊN CỨU

Đối tượng nghiên cứu trong đề tài “Quản lý cửa hàng xe máy” là quy trình nghiệp vụ bán hàng tại cửa hàng.

Phạm vi nghiên cứu: Vì lý do thời gian và trình độ của mình nên trong phạm vi của đề tài nghiên cứu này em chỉ tập trung nghiên cứu vào quy trình quản lý tại cửa hàng.

## 4.4. Ý NGHĨA KHOA HỌC VÀ THỰC TIỄN CỦA ĐỀ TÀI

Chương trình được xây dựng sẽ đáp ứng được các yêu cầu trong công tác quản lý tại cửa hàng bán xe máy.

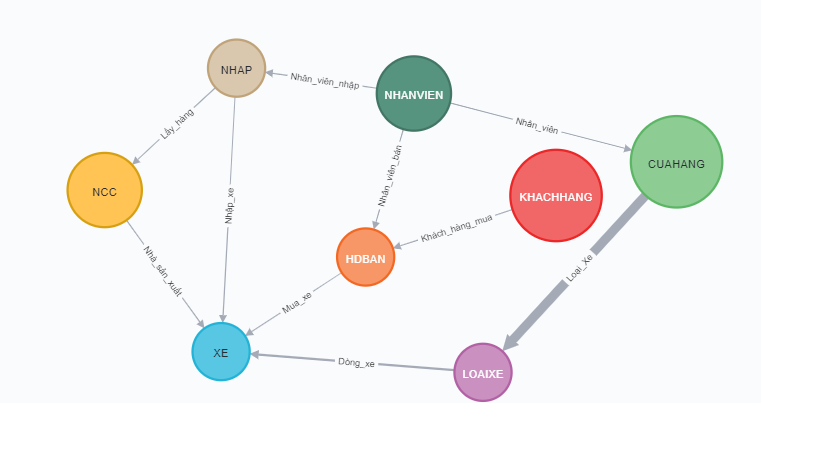
## 4.5. PHƯƠNG PHÁP NGHIÊN CỨU

Phương pháp nghiên cứu lý luận: Nghiên cứu qua việc đọc sách, báo và các tài liệu liên quan nhằm xây dựng cơ sở lý thuyết của đề tài và các biện pháp cần thiết để giải quyết các vấn đề của đề tài.

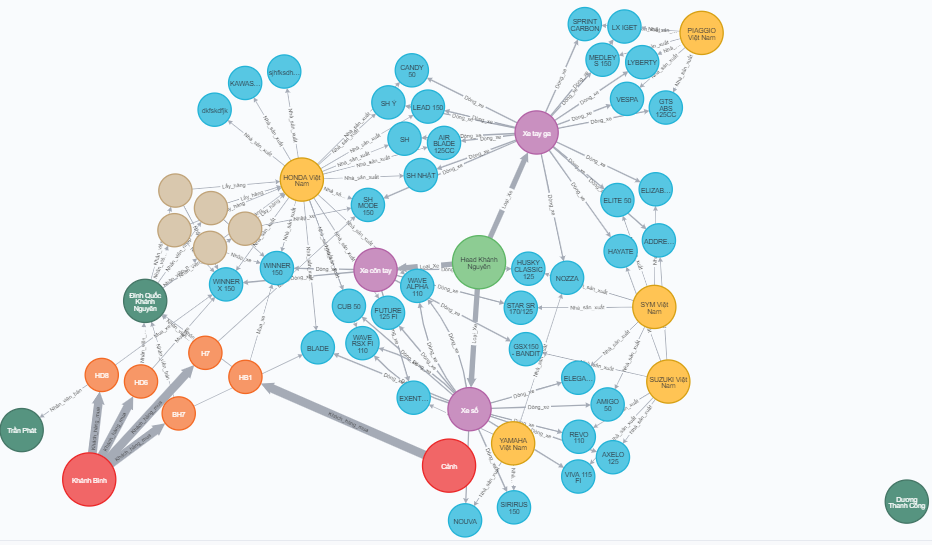
Phương pháp chuyên gia: Tham khảo ý kiến của các chuyên gia để có thể thiết kế chương trình phù hợp với yêu cầu thực tiễn, nội dung xử lý nhanh đáp ứng yêu cầu ngày càng cao của ngƣời sử dụng.

Phương pháp thực nghiệm: Thông qua quan sát thực tế, yêu cầu của cửa hàng đặt ra để hoàn thành tốt đề tài một cách khách quan

# CHƯƠNG 5: THIẾT KẾ GIAO DIỆN



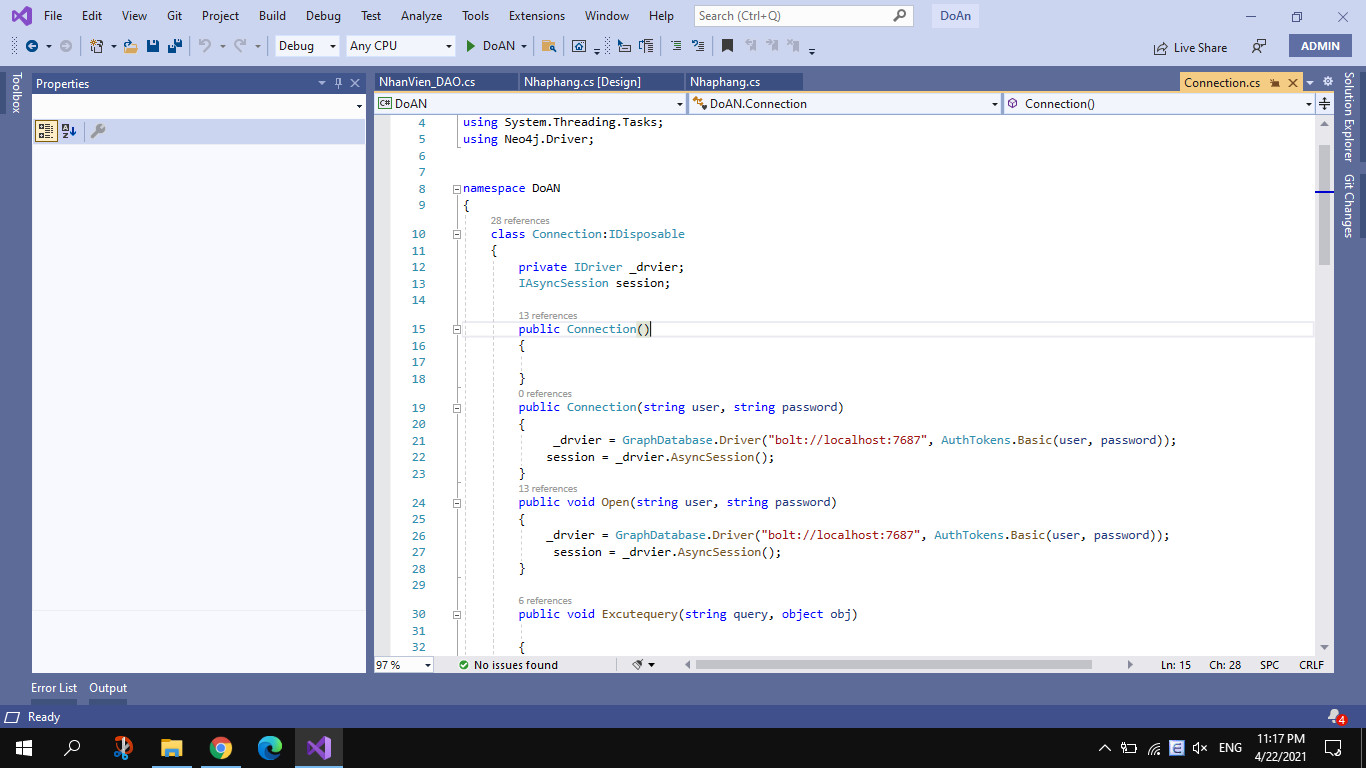
*Hình 5.1. Database*

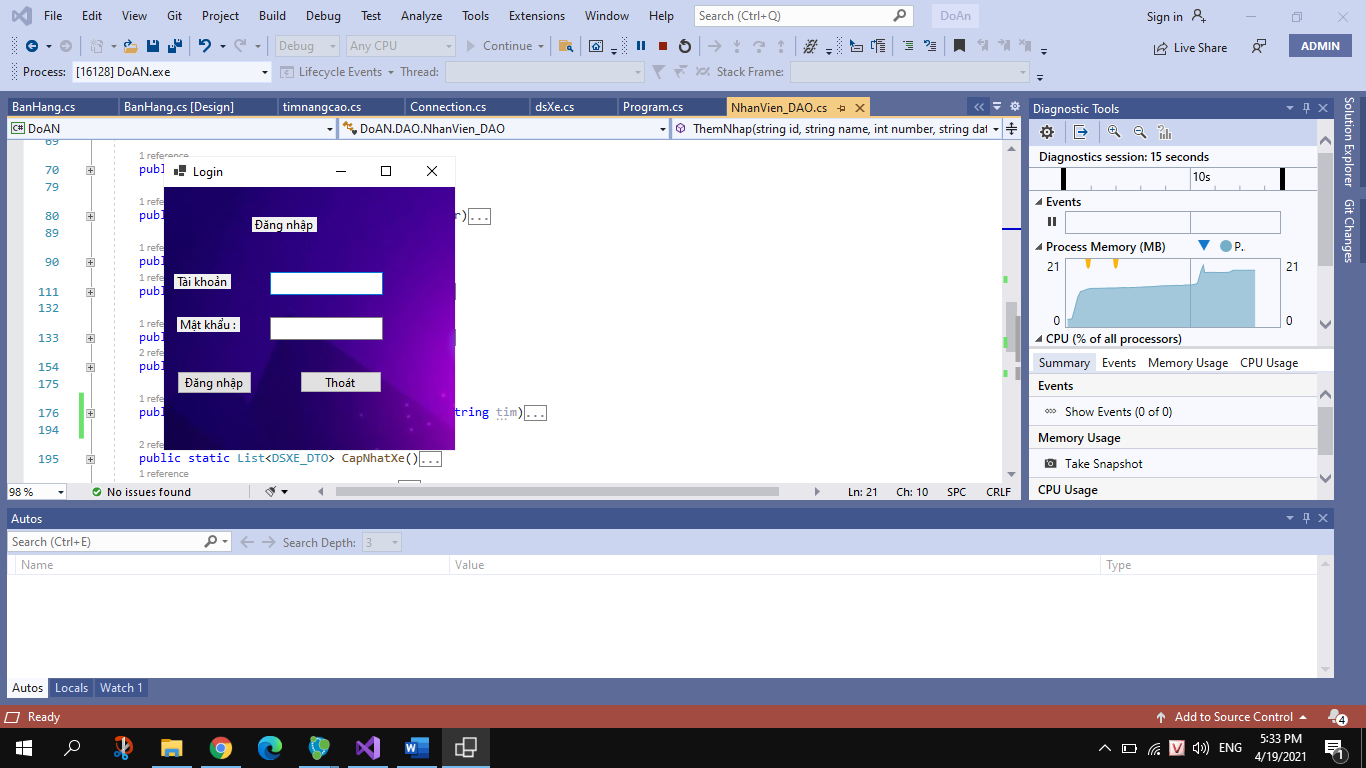


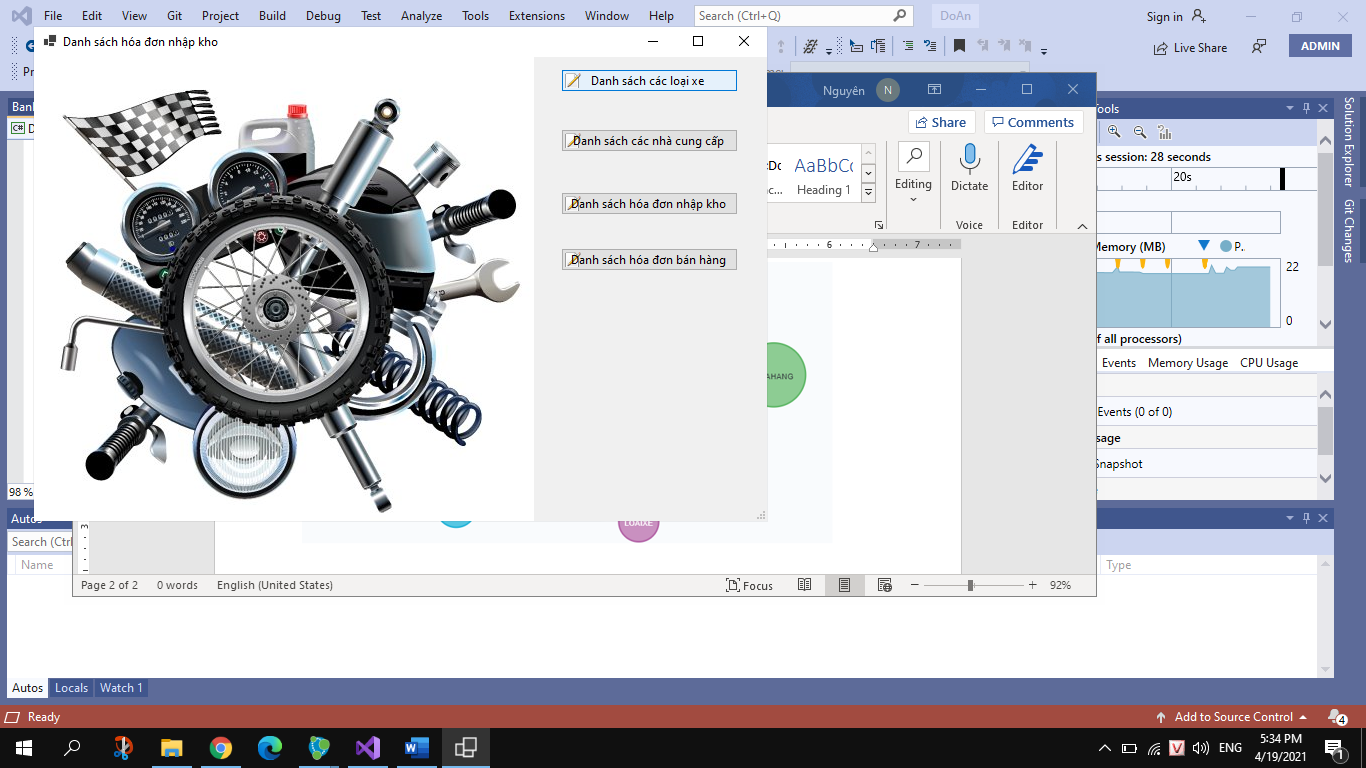
*Hình 5.2. Cơ sở dữ liệu đồ thị*

Khi muốn kết nối neo4j thì ta cần phải cài đặt Nuget với tên là neo4j drive

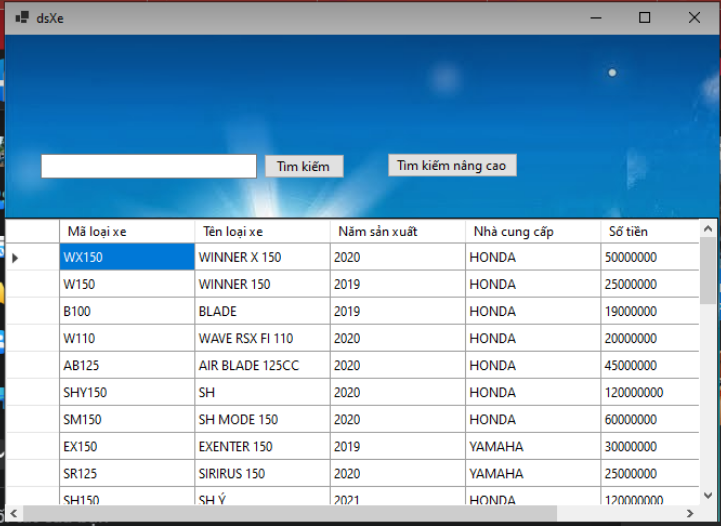


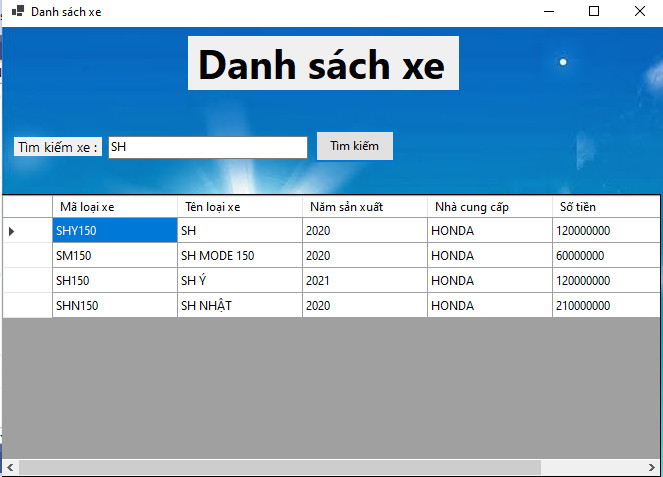




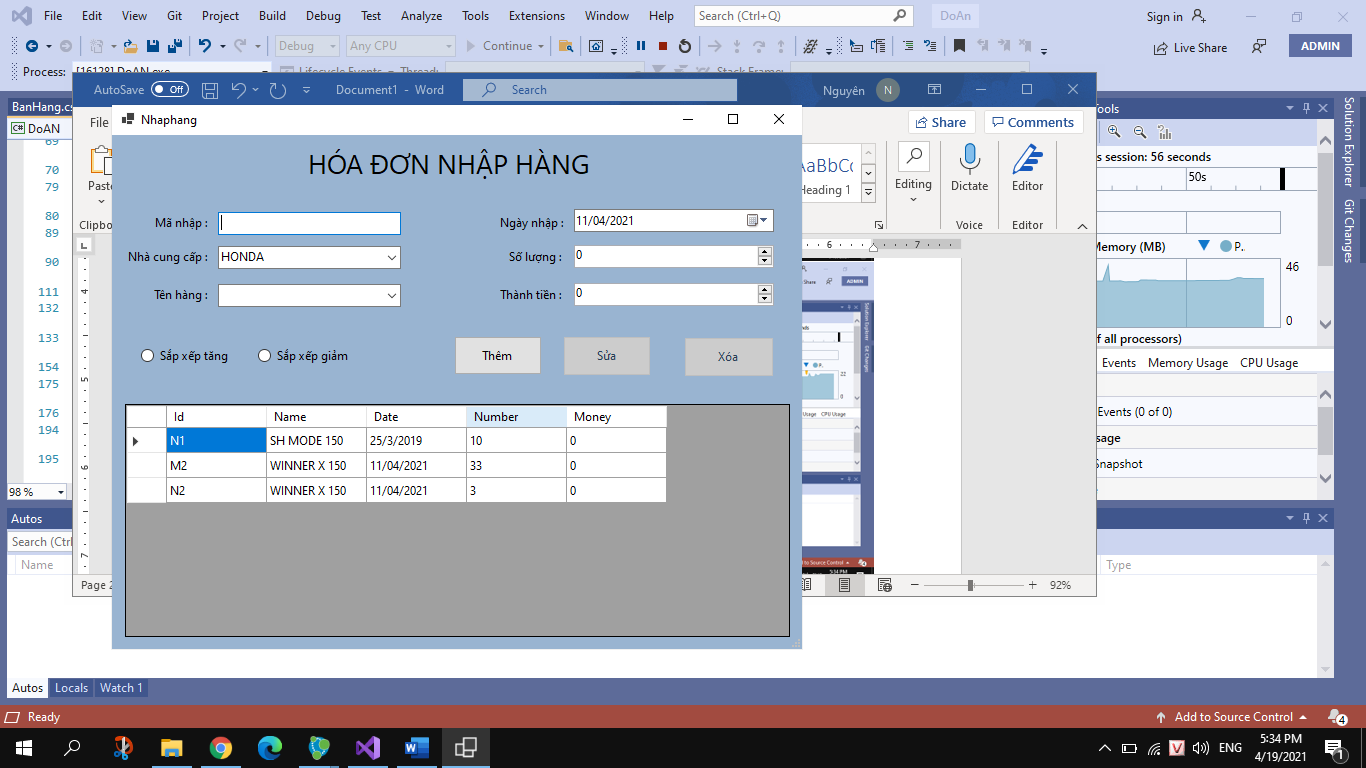
*Hình 5.3. Form đăng nhập*

*Hình 5.4. Form hệ thống*

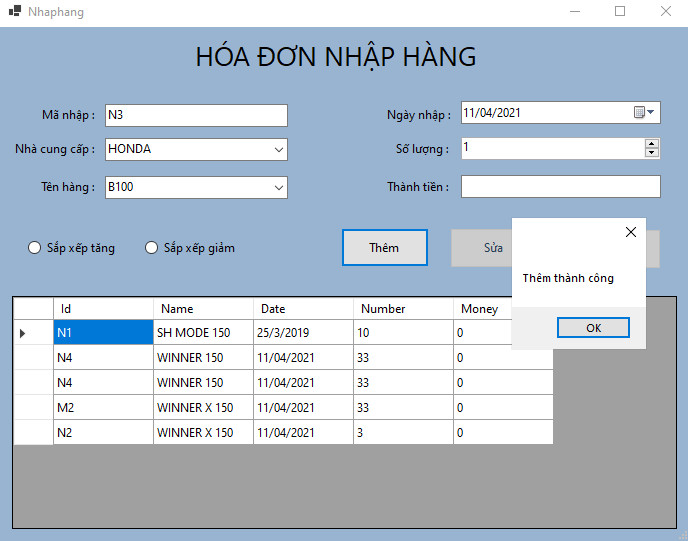


*Hình 5.5. Form danh sách các loại xe*

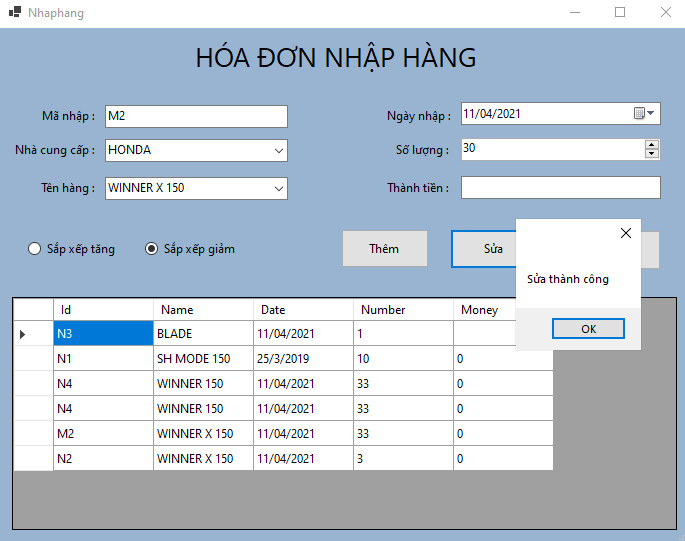
*Hình 5.6. Tìm kiếm xe*



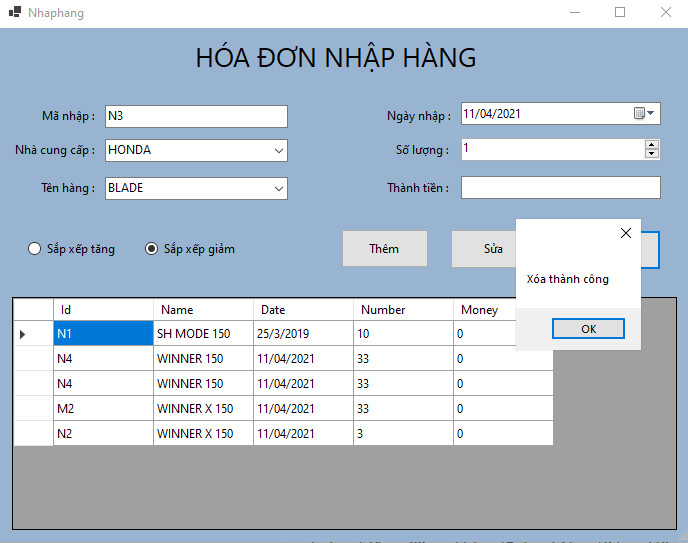
*Hình 5.7. Form hóa đơn nhập hàng*



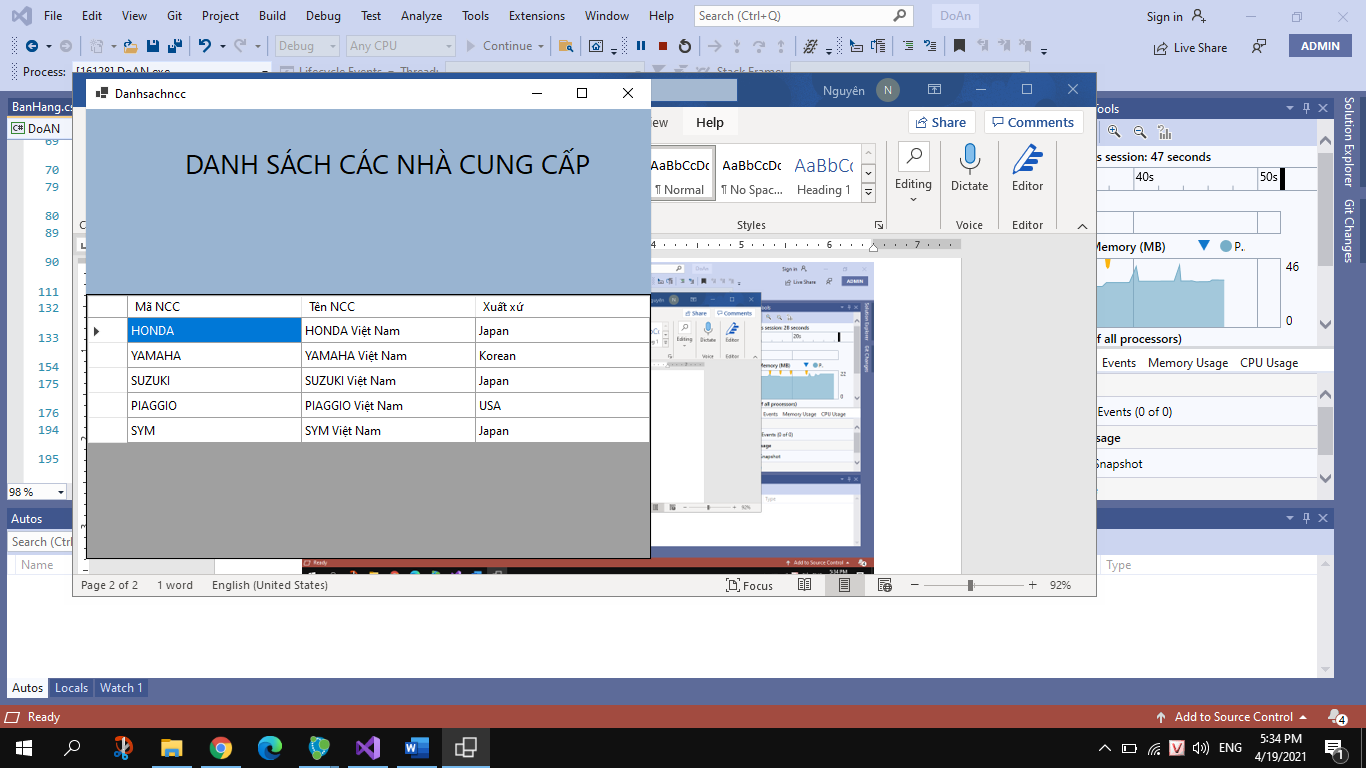
*Hình 5.8. Thêm hóa đơn nhập hàng*



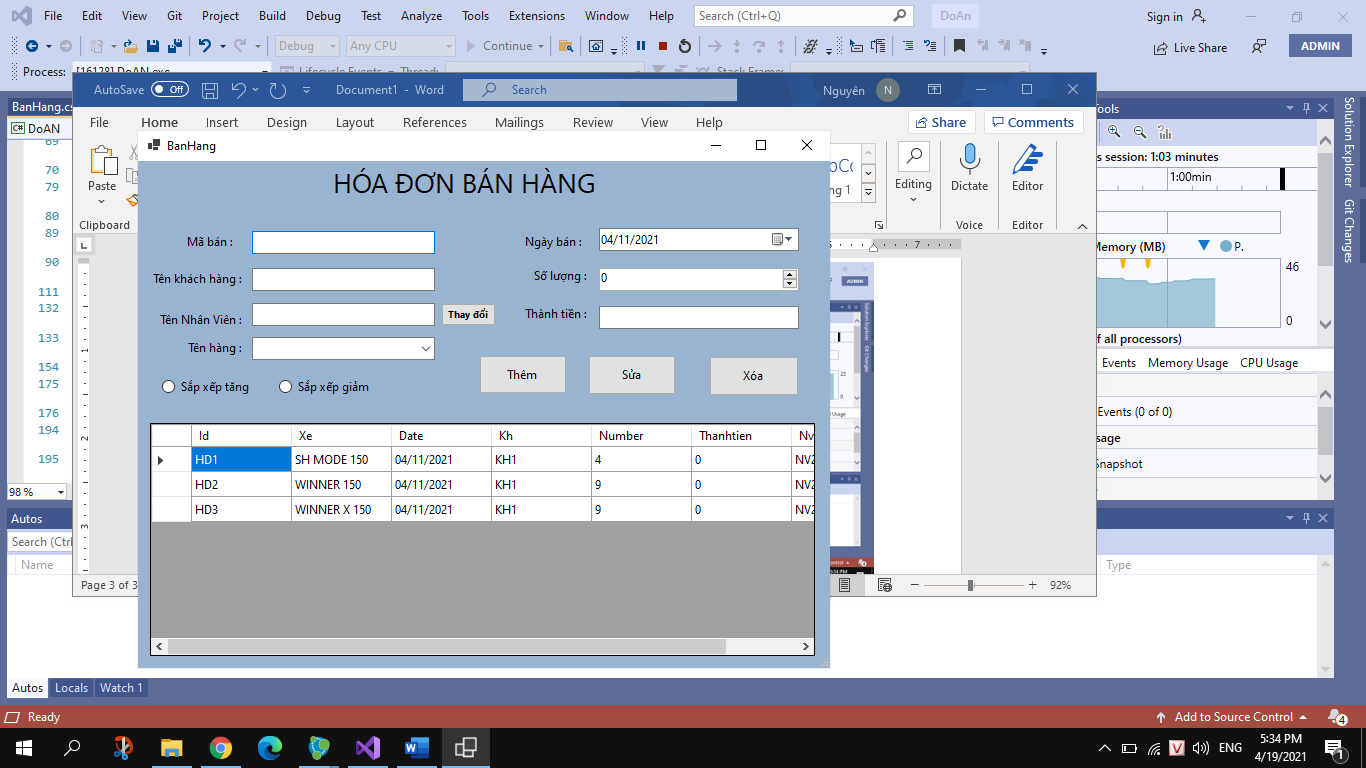
*Hình 5.9. Sửa hóa đơn nhập hàng*



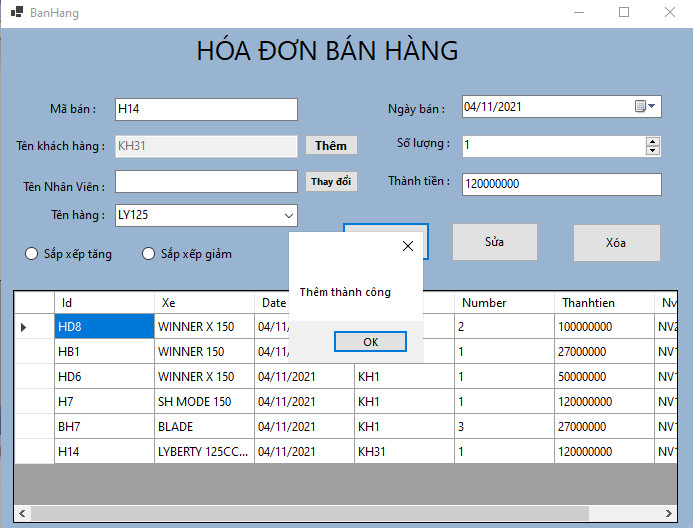
*Hình 5.10. Xóa hóa đơn nhập hàng*



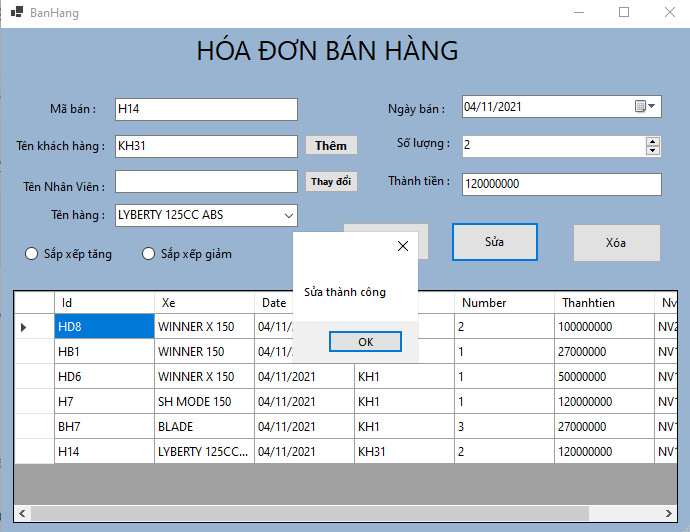
*Hình 5.11. form danh sách các nhà cung cấp*



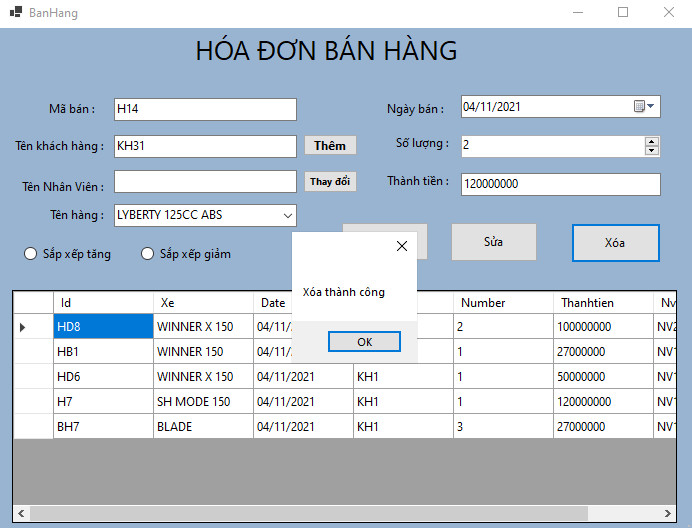
*Hình 5.12. Form hóa đơn bán hàng*



*Hình 5.13. Thêm hóa đơn bán hàng*



*Hình 5.14. Sửa hóa đơn bán hàng*



*Hình 5.15. Xóa hóa đơn bán hàng*

# TÀI LIỆU THAM KHẢO

1. [*https://xemtailieu.com/tai-lieu/chuong-trinh-quan-ly-cua-hang-ban-xe-may-duy-hien-1181428.html*](https://xemtailieu.com/tai-lieu/chuong-trinh-quan-ly-cua-hang-ban-xe-may-duy-hien-1181428.html)
2. <https://viblo.asia/p/bat-dau-voi-neo4j-4P856Ny35Y3>
3. <https://123doc.net/document/2636650-co-so-du-lieu-do-thi-neo4j-va-ung-dung.htm>
4. <https://neo4j.com/>
5. <https://neo4j.com/developer/neo4j-apoc/>