**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**----o0o----**

****

**HỌC PHẦN: CƠ SỞ DỮ LIỆU NOSQL**

**ĐỀ TÀI:** **THIẾT KẾ CSDL ĐỒ THỊ QUẢN LÝ DỮ LIỆU CÔNG DÂN VÀ XÂY DỰNG ỨNG DỤNG MINH HỌA.**

**GVHD: Trần Trương Tấn Phát**

**Lớp: 12DHTH15**

**Nhóm: 8**

|  |  |
| --- | --- |
| **1. Vương Hoàng Khải** | **MSSV: 2001215845** |
| **2. Nguyễn Thanh Vi** | **MSSV: 2001216310** |
| **3. Cao Hoàng Nhật Thăng** | **MSSV: 2001216157** |
|  |  |

***TP. HCM, THÁNG 10 NĂM 2024***

**MỤC LỤC**

[BẢNG PHÂN CÔNG 1](#_Toc179544324)

[PHẦN MỞ ĐẦU 2](#_Toc179544325)

[PHẦN NỘI DUNG 3](#_Toc179544326)

[I. THIẾT KẾ MÔ HÌNH DỮ LIỆU 3](#_Toc179544327)

[1.1. Khảo sát hệ thống 3](#_Toc179544328)

[1.1.1. Tổng quan hệ thống 3](#_Toc179544329)

[1.1.2. Yêu cầu quản lý dữ liệu công dân 3](#_Toc179544330)

[1.1.3. Mô hình hóa dữ liệu trong hệ thống 3](#_Toc179544331)

[1.1.4. Mô hình dữ liệu đồ thị trong Neo4j 4](#_Toc179544332)

[1.1.5. Các yêu cầu chức năng 4](#_Toc179544333)

[1.1.6. Các yêu cầu phi chức năng 5](#_Toc179544334)

[1.1.7. Kết luận khảo sát 5](#_Toc179544335)

[1.2. Thiết mô hình dữ liệu 6](#_Toc179544336)

[1.2.1. Nút (Node) 6](#_Toc179544337)

[1.2.2. Mối quan hệ (Relationship) 7](#_Toc179544338)

[1.2.3. Mô hình đồ thị tổng quát 8](#_Toc179544339)

[1.3. Xây dựng database dựa vào mô hình đã thiết kế 8](#_Toc179544340)

[1.3.1. Tạo nút 9](#_Toc179544341)

[1.3.2. Tạo mối quan hệ 10](#_Toc179544342)

[II. KHAI THÁC DỮ LIỆU 12](#_Toc179544343)

[2.1. Các yêu cầu khai thác dữ liệu và lệnh truy vấn tương ứng 12](#_Toc179544344)

[2.2. Sao lưu và phục hồi cơ sở dữ liệu 12](#_Toc179544345)

[III. ỨNG DỤNG 12](#_Toc179544346)

[PHẦN KẾT LUẬN 13](#_Toc179544347)

[TÀI LIỆU THAM KHẢO 14](#_Toc179544348)

BẢNG PHÂN CÔNG

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **Họ tên MSSV** | **Nhiệm vụ** | **Hoàn Thành** |
| 1 | Vương Hoàng Khải - 2001215845 | Truy vấn mối quan hệ, địa chỉ Tạo form mối quan hệ, địa chỉ  Làm ppt, word | 100% |
| 2 | Nguyễn Thanh Vi - 2001216310 | Truy vấn trường học, thống kê  Tạo form trường học, thống kê, main  Setup cơ bản mô hình 3 lớp DAL-BUS-GUI và class kết nối Neo4J  Sao lưu, phục hồi | 100% |
| 3 | Cao Hoàng Nhật Thăng - 2001216157 | Truy vấn công đân, công việc  Tạo form công dân, công việc | 100% |

PHẦN MỞ ĐẦU

Trong bối cảnh phát triển mạnh mẽ của công nghệ thông tin và nhu cầu quản lý dữ liệu ngày càng lớn, việc tổ chức và khai thác thông tin hiệu quả trở thành yếu tố quan trọng trong nhiều lĩnh vực. Đặc biệt, trong quản lý dữ liệu công dân, việc thu thập, lưu trữ và phân tích thông tin một cách chính xác và nhanh chóng có ý nghĩa to lớn đối với chính quyền, các tổ chức và cả cộng đồng. Tuy nhiên, với sự phức tạp và đa dạng của các mối quan hệ giữa các đối tượng dữ liệu, các hệ quản trị cơ sở dữ liệu truyền thống gặp nhiều khó khăn trong việc xử lý và truy vấn thông tin phức tạp một cách hiệu quả.

Neo4j, một trong những hệ quản trị cơ sở dữ liệu đồ thị nổi bật hiện nay, đã chứng tỏ được tính ưu việt trong việc quản lý các dữ liệu có mối liên hệ chặt chẽ và phức tạp. Thay vì lưu trữ dữ liệu dưới dạng bảng và quan hệ cột-dòng như các hệ quản trị CSDL quan hệ, Neo4j sử dụng mô hình đồ thị, giúp biểu diễn trực quan các mối quan hệ giữa các thực thể, từ đó cải thiện khả năng truy vấn và phân tích dữ liệu phức tạp.

Với đề tài "Thiết kế CSDL Neo4j đồ thị Quản lý dữ liệu công dân và xây dựng ứng dụng minh họa", mục tiêu chính là nghiên cứu và thiết kế một cơ sở dữ liệu đồ thị sử dụng Neo4j để quản lý thông tin công dân, đồng thời phát triển một ứng dụng minh họa giúp trực quan hóa và khai thác dữ liệu một cách hiệu quả. Đề tài không chỉ tập trung vào việc tổ chức dữ liệu một cách khoa học, mà còn nhấn mạnh vào tính khả dụng và hiệu suất của hệ thống, từ đó hỗ trợ các nhà quản lý trong việc ra quyết định chính xác và kịp thời.

**PHẦN NỘI DUNG**

1. **THIẾT KẾ MÔ HÌNH DỮ LIỆU**

## Khảo sát hệ thống

### 1.1.1. Tổng quan hệ thống

Trong bối cảnh dữ liệu ngày càng phức tạp và đa dạng, việc quản lý thông tin công dân đòi hỏi một hệ thống có khả năng xử lý khối lượng dữ liệu lớn và các mối quan hệ phức tạp. Cơ sở dữ liệu đồ thị Neo4j là một giải pháp phù hợp nhờ vào khả năng mô tả dữ liệu thông qua các nút (node) và các quan hệ (relationship), giúp dễ dàng lưu trữ và truy vấn các mối quan hệ giữa các thực thể như công dân, địa chỉ, công việc, và trường học. Mô hình này đặc biệt hữu ích cho các ứng dụng yêu cầu quản lý thông tin có tính liên kết cao, chẳng hạn như quản lý thông tin công dân.

### 1.1.2. Yêu cầu quản lý dữ liệu công dân

Hệ thống quản lý dữ liệu công dân cần đảm bảo việc lưu trữ và truy xuất thông tin cá nhân một cách hiệu quả, bao gồm thông tin cơ bản (họ tên, ngày sinh, CCCD), địa chỉ cư trú, quá trình học tập và công việc. Đồng thời, hệ thống phải hỗ trợ việc quản lý và theo dõi các mối quan hệ giữa các công dân như quan hệ gia đình (cha mẹ, anh chị em), bạn bè và đồng nghiệp, nhằm phục vụ cho các mục đích phân tích xã hội hoặc phục vụ cho các cơ quan quản lý nhà nước.

### 1.1.3. Mô hình hóa dữ liệu trong hệ thống

* **Công dân**: Đây là thực thể chính của hệ thống, với các thuộc tính như họ tên, ngày sinh, giới tính, số điện thoại, email, quốc tịch, nghề nghiệp và số CCCD. Các công dân được liên kết với nhiều thực thể khác qua các mối quan hệ như nơi cư trú, nơi làm việc và quá trình học tập.
* **Địa chỉ**: Mỗi công dân có thể có một hoặc nhiều địa chỉ cư trú, thông tin địa chỉ bao gồm các thuộc tính như đường, quận, thành phố và mã bưu điện. Địa chỉ giúp hệ thống theo dõi nơi cư trú của mỗi công dân qua các giai đoạn khác nhau.
* **Công việc**: Mỗi công dân có thể làm việc cho một hoặc nhiều công ty, và thông tin công việc bao gồm tên công ty, vị trí công việc, ngày bắt đầu và mức lương. Hệ thống cần đảm bảo khả năng lưu trữ và truy xuất thông tin về nơi làm việc của công dân.
* **Trường học**: Thực thể này lưu trữ thông tin về quá trình học tập của công dân, bao gồm tên trường, ngày bắt đầu, ngày kết thúc và bằng cấp. Việc quản lý quá trình học tập sẽ giúp theo dõi sự phát triển của công dân từ cấp tiểu học đến bậc đại học.
* **Mối quan hệ**: Hệ thống sẽ lưu trữ các mối quan hệ giữa các công dân như quan hệ gia đình (cha mẹ - con cái, anh chị em) và quan hệ bạn bè. Các mối quan hệ này không chỉ phục vụ cho việc quản lý thông tin mà còn giúp thực hiện các phân tích mạng xã hội.

### 1.1.4. Mô hình dữ liệu đồ thị trong Neo4j

Hệ thống sử dụng cơ sở dữ liệu Neo4j để mô hình hóa các thực thể dưới dạng các **nút** (nodes) và các **quan hệ** (relationships):

* Các công dân được lưu trữ dưới dạng các nút :Citizen, mỗi nút đại diện cho một công dân với các thuộc tính liên quan (họ tên, ngày sinh, CCCD, v.v.).
* Các địa chỉ được lưu trữ dưới dạng nút :Address, và được kết nối với các công dân qua quan hệ :LIVES\_AT.
* Các công việc được lưu trữ dưới dạng nút :Job, và được kết nối với các công dân qua quan hệ :WORKS\_AT.
* Các trường học được lưu trữ dưới dạng nút :School, và được kết nối với các công dân qua quan hệ :STUDIED\_AT.
* Các mối quan hệ giữa các công dân được biểu diễn qua các quan hệ như :PARENT\_OF, :SIBLING\_OF, và :FRIEND\_OF.

### 1.1.5. Các yêu cầu chức năng

Hệ thống cần cung cấp các chức năng chính sau:

* **Quản lý công dân**: Thêm, xóa, sửa thông tin cá nhân của công dân (bao gồm CCCD, họ tên, ngày sinh, giới tính, địa chỉ, nghề nghiệp, quá trình học tập).
* **Quản lý địa chỉ**: Cập nhật thông tin về nơi cư trú của công dân, lưu trữ lịch sử thay đổi địa chỉ qua các thời kỳ.
* **Quản lý công việc**: Theo dõi lịch sử công việc của công dân, bao gồm nơi làm việc, vị trí, mức lương và thời gian làm việc.
* **Quản lý quá trình học tập**: Lưu trữ thông tin về các trường học mà công dân đã theo học và bằng cấp đạt được.
* **Quản lý mối quan hệ**: Thêm, xóa và quản lý các mối quan hệ xã hội của công dân (gia đình, bạn bè, đồng nghiệp).

### 1.1.6. Các yêu cầu phi chức năng

* **Hiệu năng**: Hệ thống phải đảm bảo khả năng truy vấn dữ liệu nhanh chóng và chính xác, đặc biệt là khi truy xuất các mối quan hệ phức tạp giữa các công dân.
* **Tính mở rộng**: Hệ thống phải có khả năng mở rộng khi số lượng công dân và dữ liệu tăng lên, đồng thời hỗ trợ các yêu cầu phân tích dữ liệu và báo cáo.
* **Tính bảo mật**: Đảm bảo bảo mật thông tin cá nhân của công dân, chỉ những người có quyền mới được phép truy cập và chỉnh sửa thông tin.
* **Tính toàn vẹn**: Dữ liệu trong hệ thống phải được đảm bảo tính toàn vẹn, không để xảy ra tình trạng trùng lặp hoặc thiếu sót thông tin.

### 1.1.7. Kết luận khảo sát

Qua quá trình khảo sát và phân tích yêu cầu, chúng em nhận thấy việc sử dụng Neo4j để thiết kế cơ sở dữ liệu đồ thị cho hệ thống quản lý dữ liệu công dân là một lựa chọn hợp lý và tối ưu. Neo4j cung cấp khả năng lưu trữ và truy vấn dữ liệu mạnh mẽ, đặc biệt là trong việc xử lý các mối quan hệ phức tạp, từ đó giúp cải thiện hiệu suất và hiệu quả của hệ thống.

## Thiết mô hình dữ liệu

A diagram of a company

Description automatically generated

### 1.2.1. Nút (Node)

Các nút đại diện cho các thực thể chính trong hệ thống. Mỗi nút có các **thuộc tính** (properties) để lưu trữ thông tin chi tiết về thực thể đó.

* **Citizen (Công dân)**: Lưu trữ thông tin cá nhân của mỗi công dân.
  + **Thuộc tính**:
    - id: Mã định danh của công dân (mã duy nhất).
    - name: Họ và tên.
    - birthdate: Ngày sinh.
    - gender: Giới tính.
    - phone: Số điện thoại.
    - email: Email liên hệ.
    - nationality: Quốc tịch.
    - occupation: Nghề nghiệp.
    - cccd: Số căn cước công dân (mã duy nhất).
* **Address (Địa chỉ)**: Lưu trữ thông tin về nơi cư trú của công dân.
  + **Thuộc tính**:
    - street: Đường.
    - district: Quận/Huyện.
    - city: Thành phố.
    - country: Quốc gia.
    - postal\_code: Mã bưu điện.
* **Job (Công việc)**: Lưu trữ thông tin về công việc mà công dân đang hoặc đã làm.
  + **Thuộc tính**:
    - company: Tên công ty.
    - position: Vị trí công việc.
    - start\_date: Ngày bắt đầu làm việc.
    - salary: Mức lương.
* **School (Trường học)**: Lưu trữ thông tin về các trường học mà công dân đã theo học.
  + **Thuộc tính**:
    - name: Tên trường.
    - start\_date: Ngày bắt đầu học.
    - end\_date: Ngày kết thúc học.
    - degree: Bằng cấp.

### 1.2.2. Mối quan hệ (Relationship)

Các mối quan hệ giữa các nút mô tả cách các thực thể tương tác và liên kết với nhau.

* **LIVES\_AT**: Mối quan hệ giữa công dân và địa chỉ nơi họ cư trú.
  + **Mô hình**: (Citizen)-[:LIVES\_AT]->(Address)
* **WORKS\_AT**: Mối quan hệ giữa công dân và công việc mà họ đang làm hoặc đã làm.
  + **Mô hình**: (Citizen)-[:WORKS\_AT]->(Job)
* **STUDIED\_AT**: Mối quan hệ giữa công dân và trường học mà họ đã học.
  + **Mô hình**: (Citizen)-[:STUDIED\_AT]->(School)
* **PARENT\_OF**: Mối quan hệ giữa công dân và con cái của họ.
  + **Mô hình**: (Citizen)-[:PARENT\_OF]->(Citizen)
* **SIBLING\_OF**: Mối quan hệ giữa các công dân là anh chị em.
  + **Mô hình**: (Citizen)-[:SIBLING\_OF]->(Citizen)
* **FRIEND\_OF**: Mối quan hệ giữa các công dân là bạn bè.
  + **Mô hình**: (Citizen)-[:FRIEND\_OF]->(Citizen)

### 1.2.3. Mô hình đồ thị tổng quát

Mô hình đồ thị của hệ thống sẽ được cấu trúc như sau:

* Mỗi **Công dân** (Citizen) sẽ được liên kết với:
  + **Địa chỉ** (Address) thông qua quan hệ LIVES\_AT.
  + **Công việc** (Job) thông qua quan hệ WORKS\_AT.
  + **Trường học** (School) thông qua quan hệ STUDIED\_AT.
  + Các **Công dân khác** qua các quan hệ xã hội như PARENT\_OF, SIBLING\_OF, và FRIEND\_OF.

## Xây dựng database dựa vào mô hình đã thiết kế

A diagram of a network

Description automatically generated

### 1.3.1. Tạo nút

*a) Tạo Công Dân*

CREATE (a:Citizen {id: '001', name: 'Nguyen Van A', birthdate: '1990-01-01', gender: 'Male', phone: '0123456789', email: 'nva@gmail.com', nationality: 'Vietnamese', occupation: 'Engineer', cccd: '123456789012'});

CREATE (b:Citizen {id: '002', name: 'Tran Thi B', birthdate: '1995-05-10', gender: 'Female', phone: '0987654321', email: 'ttb@gmail.com', nationality: 'Vietnamese', occupation: 'Doctor', cccd: '234567890123'});

CREATE (c:Citizen {id: '003', name: 'Le Van C', birthdate: '1987-03-22', gender: 'Male', phone: '0932123456', email: 'lvc@gmail.com', nationality: 'Vietnamese', occupation: 'Teacher', cccd: '345678901234'});

CREATE (d:Citizen {id: '004', name: 'Pham Thi D', birthdate: '2000-08-15', gender: 'Female', phone: '0978312456', email: 'ptd@gmail.com', nationality: 'Vietnamese', occupation: 'Student', cccd: '456789012345'});

CREATE (e:Citizen {id: '005', name: 'Nguyen Thi E', birthdate: '1992-02-20', gender: 'Female', phone: '0912345678', email: 'nte@gmail.com', nationality: 'Vietnamese', occupation: 'Nurse', cccd: '567890123456'});

CREATE (f:Citizen {id: '006', name: 'Tran Van F', birthdate: '1985-12-12', gender: 'Male', phone: '0901234567', email: 'tvf@gmail.com', nationality: 'Vietnamese', occupation: 'Manager', cccd: '678901234567'});

*b) Tạo Địa Chỉ*

CREATE (addr1:Address {street: '123 Le Loi', district: '1', city: 'HCM', country: 'Vietnam', postal\_code: '700000'});

CREATE (addr2:Address {street: '456 Hai Ba Trung', district: '3', city: 'HCM', country: 'Vietnam', postal\_code: '700001'});

CREATE (addr3:Address {street: '789 Dien Bien Phu', district: 'Binh Thanh', city: 'HCM', country: 'Vietnam', postal\_code: '700002'});

CREATE (addr4:Address {street: '101 Tran Hung Dao', district: '1', city: 'HCM', country: 'Vietnam', postal\_code: '700003'});

CREATE (addr5:Address {street: '202 Nguyen Thi Minh Khai', district: '3', city: 'HCM', country: 'Vietnam', postal\_code: '700004'});

*c) Tạo Công Việc*

CREATE (job1:Job {company: 'ABC Corp', position: 'Engineer', start\_date: '2015-05-01', salary: 50000});

CREATE (job2:Job {company: 'XYZ Ltd', position: 'Manager', start\_date: '2018-07-10', salary: 70000});

CREATE (job3:Job {company: '123 Tech', position: 'Developer', start\_date: '2020-01-15', salary: 60000});

CREATE (job4:Job {company: 'HealthCare Inc', position: 'Nurse', start\_date: '2019-06-01', salary: 40000});

CREATE (job5:Job {company: 'Education Org', position: 'Teacher', start\_date: '2016-01-01', salary: 45000})

*d) Tạo Trường Học*

CREATE (school1:School {name: 'ĐH Bách Khoa', start\_date: '2008-09-01', end\_date: '2012-06-01', degree: 'Bachelor of Engineering'});

CREATE (school2:School {name: 'ĐH Kinh Tế', start\_date: '2010-09-01', end\_date: '2014-06-01', degree: 'Bachelor of Economics'});

CREATE (school3:School {name: 'Cao đẳng Công Nghệ Thông Tin', start\_date: '2016-09-01', end\_date: '2019-06-01', degree: 'Diploma in IT'});

CREATE (school4:School {name: 'Trường Trung Học Phổ Thông', start\_date: '2005-09-01', end\_date: '2008-06-01', degree: 'High School Diploma'});

CREATE (school5:School {name: 'Trường Tiểu Học', start\_date: '2000-09-01', end\_date: '2005-06-01', degree: 'Elementary School'});

### 1.3.2. Tạo mối quan hệ

*a) Kết Nối Công Dân với Địa Chỉ*

MATCH (a:Citizen {name: 'Nguyen Van A'}), (addr1:Address {street: '123 Le Loi'})

CREATE (a)-[:LIVES\_AT]->(addr1);

MATCH (b:Citizen {name: 'Tran Thi B'}), (addr2:Address {street: '456 Hai Ba Trung'})

CREATE (b)-[:LIVES\_AT]->(addr2);

MATCH (c:Citizen {name: 'Le Van C'}), (addr3:Address {street: '789 Dien Bien Phu'})

CREATE (c)-[:LIVES\_AT]->(addr3);

MATCH (d:Citizen {name: 'Pham Thi D'}), (addr1:Address {street: '123 Le Loi'})

CREATE (d)-[:LIVES\_AT]->(addr1);

MATCH (e:Citizen {name: 'Nguyen Thi E'}), (addr4:Address {street: '101 Tran Hung Dao'})

CREATE (e)-[:LIVES\_AT]->(addr4);

MATCH (f:Citizen {name: 'Tran Van F'}), (addr5:Address {street: '202 Nguyen Thi Minh Khai'})

CREATE (f)-[:LIVES\_AT]->(addr5);

*b) Kết Nối Công Dân với Trường Học*

MATCH (a:Citizen {name: 'Nguyen Van A'}), (school1:School {name: 'ĐH Bách Khoa'})

CREATE (a)-[:STUDIED\_AT]->(school1);

MATCH (b:Citizen {name: 'Tran Thi B'}), (school2:School {name: 'ĐH Kinh Tế'})

CREATE (b)-[:STUDIED\_AT]->(school2);

MATCH (c:Citizen {name: 'Le Van C'}), (school3:School {name: 'Cao đẳng Công Nghệ Thông Tin'})

CREATE (c)-[:STUDIED\_AT]->(school3);

MATCH (d:Citizen {name: 'Pham Thi D'}), (school4:School {name: 'Trường Trung Học Phổ Thông'})

CREATE (d)-[:STUDIED\_AT]->(school4);

MATCH (e:Citizen {name: 'Nguyen Thi E'}), (school5:School {name: 'Trường Tiểu Học'})

CREATE (e)-[:STUDIED\_AT]->(school5);

*c) Kết Nối Công Dân với Công Việc*

MATCH (a:Citizen {name: 'Nguyen Van A'}), (job1:Job {company: 'ABC Corp'})

CREATE (a)-[:WORKS\_AT]->(job1);

MATCH (b:Citizen {name: 'Tran Thi B'}), (job2:Job {company: 'XYZ Ltd'})

CREATE (b)-[:WORKS\_AT]->(job2);

MATCH (c:Citizen {name: 'Le Van C'}), (job3:Job {company: '123 Tech'})

CREATE (c)-[:WORKS\_AT]->(job3);

MATCH (e:Citizen {name: 'Nguyen Thi E'}), (job4:Job {company: 'HealthCare Inc'})

CREATE (e)-[:WORKS\_AT]->(job4);

MATCH (f:Citizen {name: 'Tran Van F'}), (job2:Job {company: 'Education Org'})

CREATE (f)-[:WORKS\_AT]->(job2);

*d) Tạo Mối Quan Hệ Giữa Các Công Dân*

MATCH (a:Citizen {name: 'Nguyen Van A'}), (d:Citizen {name: 'Pham Thi D'})

CREATE (a)-[:PARENT\_OF]->(d);

MATCH (c:Citizen {name: 'Le Van C'}), (b:Citizen {name: 'Tran Thi B'})

CREATE (c)-[:SIBLING\_OF]->(b);

MATCH (a:Citizen {name: 'Nguyen Van A'}), (c:Citizen {name: 'Le Van C'})

CREATE (a)-[:FRIEND\_OF]->(c);

MATCH (b:Citizen {name: 'Tran Thi B'}), (d:Citizen {name: 'Pham Thi D'})

CREATE (b)-[:FRIEND\_OF]->(d);

MATCH (e:Citizen {name: 'Nguyen Thi E'}), (a:Citizen {name: 'Nguyen Van A'})

CREATE (e)-[:FRIEND\_OF]->(a);

MATCH (f:Citizen {name: 'Tran Van F'}), (b:Citizen {name: 'Tran Thi B'})

CREATE (f)-[:FRIEND\_OF]->(b);

# KHAI THÁC DỮ LIỆU

## Các yêu cầu khai thác dữ liệu

A diagram of a person with text and arrows

Description automatically generated

**1. CRUD Quản lý công việc**

**Mô tả:**

Hệ thống cho phép Admin thực hiện các thao tác quản lý thông tin công việc của công dân như thêm, sửa, xóa, và xem thông tin công việc.

**Các hành động chính:**

* **Thêm công việc**:
  + Admin có thể nhập thông tin về công ty, vị trí làm việc, ngày bắt đầu và mức lương.
  + Thông tin công việc được liên kết với một công dân cụ thể.
  + Thực hiện kiểm tra trùng lặp trước khi thêm công việc mới.
* **Sửa công việc**:
  + Admin có thể cập nhật thông tin công việc của công dân khi có sự thay đổi như: thăng chức, thay đổi lương, thay đổi công ty hoặc vị trí công việc.
* **Xóa công việc**:
  + Admin có thể xóa công việc khi một công dân nghỉ việc hoặc không còn liên kết với công ty nào đó.
* **Xem thông tin công việc**:
  + Admin có thể xem lịch sử công việc của công dân, bao gồm các công việc trước đây và hiện tại.

**Các bước thực hiện:**

* **Thêm công việc**:
  1. Admin chọn công dân cần quản lý công việc.
  2. Nhập thông tin công việc (công ty, vị trí, ngày bắt đầu, lương).
  3. Lưu thông tin công việc vào cơ sở dữ liệu.
* **Sửa công việc**:
  1. Admin chọn công dân và công việc cần sửa.
  2. Cập nhật các thông tin công việc.
  3. Lưu thay đổi vào cơ sở dữ liệu.
* **Xóa công việc**:
  1. Admin chọn công việc cần xóa.
  2. Xác nhận xóa thông tin công việc của công dân.
* **Xem thông tin công việc**:
  1. Admin truy xuất thông tin công việc từ cơ sở dữ liệu và hiển thị cho người dùng.

**2. CRUD Quản lý trường học**

**Mô tả:**

Quản lý thông tin về học vấn của công dân, cho phép thêm, sửa, xóa và xem thông tin về trường học và các bằng cấp liên quan.

**Các hành động chính:**

* **Thêm thông tin trường học**:
  + Admin nhập tên trường, ngày bắt đầu, ngày kết thúc, và bằng cấp nhận được.
  + Liên kết thông tin này với công dân cụ thể.
* **Sửa thông tin trường học**:
  + Cập nhật thông tin về trường học, bằng cấp hoặc thay đổi ngày học của công dân khi có yêu cầu.
* **Xóa thông tin trường học**:
  + Xóa thông tin về một trường học nếu có sai sót hoặc yêu cầu từ phía công dân.
* **Xem thông tin học vấn**:
  + Hiển thị danh sách trường học và bằng cấp của công dân.

**Các bước thực hiện:**

* **Thêm thông tin trường học**:
  1. Admin chọn công dân cần quản lý thông tin học vấn.
  2. Nhập tên trường, bằng cấp và các thông tin khác.
  3. Lưu thông tin vào cơ sở dữ liệu.
* **Sửa thông tin trường học**:
  1. Admin chọn công dân và trường học cần cập nhật.
  2. Cập nhật thông tin trường học.
  3. Lưu thay đổi vào cơ sở dữ liệu.
* **Xóa thông tin trường học**:
  1. Admin chọn trường học cần xóa.
  2. Xác nhận và thực hiện xóa.
* **Xem thông tin học vấn**:
  1. Admin truy xuất thông tin học vấn của công dân từ cơ sở dữ liệu và hiển thị.

**3. CRUD Quản lý mối quan hệ**

**Mô tả:**

Quản lý các mối quan hệ giữa các công dân, như mối quan hệ gia đình (cha mẹ, con cái) và các quan hệ khác (bạn bè, anh chị em,...).

**Các hành động chính:**

* **Thêm mối quan hệ**:
  + Admin thêm mối quan hệ giữa hai công dân như: Cha mẹ - Con cái, Anh chị em, Bạn bè.
* **Sửa mối quan hệ**:
  + Cập nhật hoặc chỉnh sửa mối quan hệ giữa các công dân.
* **Xóa mối quan hệ**:
  + Admin có thể xóa mối quan hệ giữa hai công dân nếu không còn phù hợp.
* **Xem mối quan hệ**:
  + Hiển thị mối quan hệ giữa công dân với các cá nhân khác.

**Các bước thực hiện:**

* **Thêm mối quan hệ**:
  1. Admin chọn hai công dân cần thiết lập mối quan hệ.
  2. Chọn loại mối quan hệ (Cha mẹ - Con cái, Bạn bè,...).
  3. Lưu mối quan hệ vào cơ sở dữ liệu.
* **Sửa mối quan hệ**:
  1. Admin chọn mối quan hệ cần chỉnh sửa.
  2. Cập nhật thông tin mối quan hệ.
* **Xóa mối quan hệ**:
  1. Admin chọn mối quan hệ cần xóa.
  2. Xác nhận xóa mối quan hệ.
* **Xem mối quan hệ**:
  1. Admin truy xuất thông tin mối quan hệ của công dân từ cơ sở dữ liệu.

**4. CRUD Quản lý địa chỉ**

**Mô tả:**

Quản lý địa chỉ của công dân bao gồm các thao tác thêm, sửa, xóa và xem thông tin địa chỉ nơi ở của công dân.

**Các hành động chính:**

* **Thêm địa chỉ**:
  + Admin nhập thông tin địa chỉ nơi công dân sinh sống (đường, quận, thành phố, mã bưu chính).
* **Sửa địa chỉ**:
  + Cập nhật địa chỉ của công dân khi có thay đổi.
* **Xóa địa chỉ**:
  + Xóa địa chỉ nếu không còn phù hợp hoặc cần chỉnh sửa.
* **Xem thông tin địa chỉ**:
  + Hiển thị thông tin địa chỉ nơi ở của công dân.

**Các bước thực hiện:**

* **Thêm địa chỉ**:
  1. Admin chọn công dân cần thêm địa chỉ.
  2. Nhập thông tin địa chỉ (đường, quận, thành phố,...).
  3. Lưu địa chỉ vào cơ sở dữ liệu.
* **Sửa địa chỉ**:
  1. Admin chọn công dân và địa chỉ cần chỉnh sửa.
  2. Cập nhật thông tin địa chỉ.
* **Xóa địa chỉ**:
  1. Admin chọn địa chỉ cần xóa.
  2. Xác nhận xóa địa chỉ.
* **Xem thông tin địa chỉ**:
  1. Admin truy xuất thông tin địa chỉ của công dân từ cơ sở dữ liệu và hiển thị.

**5. CRUD Quản lý công dân**

**Mô tả:**

Quản lý thông tin cá nhân của công dân, cho phép thêm, sửa, xóa và xem chi tiết thông tin công dân.

**Các hành động chính:**

* **Thêm công dân**:
  + Admin nhập các thông tin cơ bản của công dân (tên, ngày sinh, số CCCD, quốc tịch,...).
* **Sửa thông tin công dân**:
  + Cập nhật thông tin cá nhân khi có sự thay đổi.
* **Xóa công dân**:
  + Xóa hồ sơ công dân khi không còn hợp lệ.
* **Xem thông tin công dân**:
  + Hiển thị chi tiết thông tin cá nhân công dân.

**Các bước thực hiện:**

* **Thêm công dân**:
  1. Admin nhập thông tin công dân.
  2. Lưu thông tin công dân vào cơ sở dữ liệu.
* **Sửa thông tin công dân**:
  1. Admin chọn công dân cần chỉnh sửa.
  2. Cập nhật thông tin cần thay đổi.
* **Xóa công dân**:
  1. Admin chọn công dân cần xóa.
  2. Xác nhận xóa thông tin công dân.
* **Xem thông tin công dân**:
  1. Admin truy xuất thông tin cá nhân của công dân từ cơ sở dữ liệu và hiển thị.

**6. Thống kê và báo cáo**

**Mô tả:**

Chức năng này cho phép Admin tạo các báo cáo và thống kê dựa trên thông tin công dân và dữ liệu liên quan.

**Các hành động chính:**

* **Thống kê theo tiêu chí**:
  + Admin có thể thực hiện thống kê số lượng công dân dựa trên tiêu chí như:
    - Giới tính.
    - Độ tuổi.
    - Nghề nghiệp.
    - Địa chỉ (thành phố, quận, mã bưu chính).
    - Trình độ học vấn.
* **Xuất báo cáo**:
  + Xuất báo cáo dữ liệu công dân dưới các dạng (CSV, Excel, PDF).

**Các bước thực hiện:**

* **Thống kê**:
  1. Admin chọn tiêu chí thống kê.
  2. Hệ thống truy xuất dữ liệu từ cơ sở dữ liệu.
  3. Hiển thị kết quả thống kê.
* **Xuất báo cáo**:
  1. Admin chọn định dạng báo cáo cần xuất (CSV, PDF, Excel).
  2. Xuất dữ liệu và lưu vào file.

## Các lệnh truy vấn tương ứng

1. CRUD Quản lý công việc
2. CRUD Quản lý trường học
   1. Tạo trường học mới

CREATE (school:School {name: 'ĐH Công Nghệ', start\_date: '2012-09-01', end\_date: '2016-06-01', degree: 'Bachelor of IT'});

* 1. Lấy thông tin tất cả trường học

MATCH (s:School)

RETURN s;

* 1. Tìm trường học theo tên

MATCH (s:School {name: 'ĐH Bách Khoa'})

RETURN s;

* 1. Cập nhật thông tin trường học

MATCH (s:School {name: 'ĐH Bách Khoa'})

SET s.name = 'ĐH Bách Khoa - HCM'

RETURN s;

* 1. Xóa trường học theo tên

MATCH (s:School {name: 'Cao đẳng Công Nghệ Thông Tin'})

DELETE s;

* 1. Thêm thuộc tính location cho trường học

MATCH (s:School {name: 'ĐH Bách Khoa'})

SET s.location = 'HCM'

RETURN s;

* 1. Xóa thuộc tính location

MATCH (s:School {name: 'ĐH Bách Khoa'})

REMOVE s.location

RETURN s;

* 1. Tạo 1 công dân và kết nối với trường học

MATCH (s:School {name: 'ĐH Bách Khoa'})

CREATE (c:Citizen {name: 'Nguyen Van G', birthdate: '1993-07-15'})

CREATE (c)-[:STUDIED\_AT]->(s);

* 1. Lấy danh sách công dân học tại 1 trường học

MATCH (c:Citizen)-[:STUDIED\_AT]->(s:School {name: 'ĐH Bách Khoa'})

RETURN c.name;

* 1. Cập nhật ngày bắt đầu và kết thúc của 1 trường học

MATCH (s:School {name: 'ĐH Bách Khoa'})

SET s.start\_date = '2009-09-01', s.end\_date = '2013-06-01'

RETURN s;

* 1. Xóa tất cả các kết nối giữa công dân và trường học

MATCH (c:Citizen)-[r:STUDIED\_AT]->(s:School {name: 'ĐH Bách Khoa'})

DELETE r;

* 1. Xóa tất cả trường học không có công dân nào học

MATCH (s:School)

WHERE NOT (s)<-[:STUDIED\_AT]-(:Citizen)

DELETE s;

* 1. Tìm trường học có thời gian học dài nhất

MATCH (s:School)

RETURN s, duration.between(date(s.start\_date), date(s.end\_date)) AS duration

ORDER BY duration DESC LIMIT 1;

* 1. Tìm tất cả công dân đã học tại 1 trường trong khoảng thời gian cụ thể

MATCH (c:Citizen)-[:STUDIED\_AT]->(s:School)

WHERE s.start\_date > '2010-01-01' AND s.end\_date < '2020-01-01'

RETURN c, s;

* 1. Thêm nhiều công dân vào 1 trường học

MATCH (s:School {name: 'ĐH Bách Khoa'})

MATCH (c1:Citizen {name: 'Nguyen Van H'}), (c2:Citizen {name: 'Tran Thi I'})

CREATE (c1)-[:STUDIED\_AT]->(s), (c2)-[:STUDIED\_AT]->(s);

* 1. Xóa công dân khỏi trường học

MATCH (c:Citizen {name: 'Nguyen Van H'})-[r:STUDIED\_AT]->(s:School {name: 'ĐH Bách Khoa'})

DELETE r;

* 1. Đếm số công dân đã học tại 1 trường học

MATCH (s:School)<-[:STUDIED\_AT]-(c:Citizen)

RETURN s.name, count(c) AS student\_count;

* 1. Lấy thông tin trường học gần nhất 1 công dân đã học

MATCH (c:Citizen {name: 'Tran Thi B'})-[:STUDIED\_AT]->(s:School)

RETURN s ORDER BY s.end\_date DESC LIMIT 1;

* 1. Tìm tất cả công dân học cùng 1 trường

MATCH (c1:Citizen)-[:STUDIED\_AT]->(s:School)<-[:STUDIED\_AT]-(c2:Citizen)

WHERE c1 <> c2

RETURN c1.name, c2.name, s.name;

* 1. Tìm các trường học ở 1 khu vực cụ thể

MATCH (s:School)

WHERE s.location = 'HCM'

RETURN s;

1. CRUD Quản lý mối quan hệ
   1. Liệt kê tất cả các mối quan hệ giữa các công dân

MATCH (c1:Citizen)-[r]->(c2:Citizen)

RETURN c1.name AS Citizen1, type(r) AS Relationship, c2.name AS Citizen2;

* 1. Liệt kê tất cả các mối quan hệ "bạn bè" (FRIEND\_OF) giữa các công dân

MATCH (c1:Citizen)-[:FRIEND\_OF]->(c2:Citizen)

RETURN c1.name AS Friend1, c2.name AS Friend2;

* 1. Tìm tất cả các con cái (PARENT\_OF) của Nguyễn Văn A

MATCH (p:Citizen {name: 'Nguyen Van A'})-[:PARENT\_OF]->(c:Citizen)

RETURN c.name AS Children;

* 1. Tìm tất cả các cha mẹ (PARENT\_OF) của Phạm Thị D

MATCH (c:Citizen {name: 'Pham Thi D'})<-[:PARENT\_OF]-(p:Citizen)

RETURN p.name AS Parent;

* 1. Liệt kê tất cả các mối quan hệ anh/chị/em (SIBLING\_OF) giữa các công dân

MATCH (c1:Citizen)-[:SIBLING\_OF]->(c2:Citizen)

RETURN c1.name AS Sibling1, c2.name AS Sibling2;

* 1. Tìm tất cả các bạn bè của Trần Thị B

MATCH (c1:Citizen {name: 'Tran Thi B'})-[:FRIEND\_OF]->(c2:Citizen)

RETURN c2.name AS Friend;

* 1. Liệt kê tất cả các mối quan hệ gia đình (PARENT\_OF, SIBLING\_OF) của Nguyễn Văn A

MATCH (c1:Citizen {name: 'Nguyen Van A'})-[r:PARENT\_OF|SIBLING\_OF]->(c2:Citizen)

RETURN c2.name AS FamilyMember, type(r) AS Relationship;

* 1. Tìm tất cả những người có mối quan hệ với Lê Văn C

MATCH (c1:Citizen {name: 'Le Van C'})-[r]->(c2:Citizen)

RETURN c2.name AS RelatedPerson, type(r) AS Relationship;

* 1. Tìm tất cả các công dân có quan hệ với Nguyễn Thị E

MATCH (c1:Citizen {name: 'Nguyen Thi E'})-[r]->(c2:Citizen)

RETURN c2.name AS RelatedPerson, type(r) AS Relationship;

* 1. Tìm những người bạn chung của Nguyễn Văn A và Trần Thị B

MATCH (a:Citizen {name: 'Nguyen Van A'})-[:FRIEND\_OF]->(c:Citizen)<-[:FRIEND\_OF]-(b:Citizen {name: 'Tran Thi B'})

RETURN c.name AS MutualFriend;

* 1. Liệt kê tất cả các mối quan hệ cha mẹ (PARENT\_OF)

MATCH (p:Citizen)-[:PARENT\_OF]->(c:Citizen)

RETURN p.name AS Parent, c.name AS Child;

* 1. Tìm những người bạn của bạn bè Nguyễn Văn A

MATCH (a:Citizen {name: 'Nguyen Van A'})-[:FRIEND\_OF]->(f:Citizen)-[:FRIEND\_OF]->(fof:Citizen)

RETURN fof.name AS FriendOfFriend;

* 1. Kiểm tra xem Nguyễn Văn A có phải là cha của Phạm Thị D hay không

MATCH (p:Citizen {name: 'Nguyen Van A'})-[:PARENT\_OF]->(c:Citizen {name: 'Pham Thi D'})

RETURN p.name AS Parent, c.name AS Child;

* 1. Tìm tất cả anh/chị/em của Trần Thị B

MATCH (c1:Citizen {name: 'Tran Thi B'})-[:SIBLING\_OF]->(c2:Citizen)

RETURN c2.name AS Sibling;

* 1. Liệt kê tất cả bạn bè của Nguyễn Thị E

MATCH (c1:Citizen {name: 'Nguyen Thi E'})-[:FRIEND\_OF]->(c2:Citizen)

RETURN c2.name AS Friend;

* 1. Tìm tất cả các con cái của Trần Văn F

MATCH (p:Citizen {name: 'Tran Van F'})-[:PARENT\_OF]->(c:Citizen)

RETURN c.name AS Child;

* 1. Liệt kê các mối quan hệ bạn bè mà không có quan hệ gia đình (PARENT\_OF, SIBLING\_OF)

MATCH (c1:Citizen)-[:FRIEND\_OF]->(c2:Citizen)

WHERE NOT (c1)-[:PARENT\_OF|SIBLING\_OF]->(c2)

RETURN c1.name AS Friend1, c2.name AS Friend2;

* 1. Tìm tất cả bạn bè của các anh/chị/em của Lê Văn C

MATCH (c:Citizen {name: 'Le Van C'})-[:SIBLING\_OF]->(s:Citizen)-[:FRIEND\_OF]->(f:Citizen)

RETURN f.name AS FriendOfSibling;

* 1. Tìm tất cả những người có mối quan hệ với cả Nguyễn Văn A và Lê Văn C

MATCH (c:Citizen)-[:FRIEND\_OF]->(a:Citizen {name: 'Nguyen Van A'})

MATCH (c)-[:FRIEND\_OF]->(b:Citizen {name: 'Le Van C'})

RETURN c.name AS MutualFriend;

* 1. Liệt kê các mối quan hệ không phải là bạn bè (FRIEND\_OF)

MATCH (c1:Citizen)-[r]->(c2:Citizen)

WHERE type(r) <> 'FRIEND\_OF'

RETURN c1.name AS Citizen1, type(r) AS Relationship, c2.name AS Citizen2;

1. CRUD Quản lý địa chỉ
   1. Lấy thông tin của tất cả công dân và địa chỉ họ sống

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address)

RETURN c.name AS Citizen, a.street AS Street, a.district AS District, a.city AS City;

* 1. Tìm công dân sống tại địa chỉ "123 Le Loi"

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address)

RETURN c.name AS Citizen, a.street AS Street, a.district AS District, a.city AS City;

* 1. Liệt kê các công dân sống ở quận 1

MATCH (a:Address {district: '1'})

RETURN a.street AS Street, a.city AS City;

* 1. Liệt kê các công dân sống tại quận 3

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address {district: '3'})

RETURN c.name AS Citizen, a.street AS Street;

* 1. Tìm công dân nào đang sống tại "202 Nguyen Thi Minh Khai"

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address {street: '202 Nguyen Thi Minh Khai'})

RETURN c.name AS Citizen;

* 1. Liệt kê các địa chỉ có mã bưu điện là '700001'

MATCH (a:Address {postal\_code: '700001'})

RETURN a.street AS Street, a.district AS District, a.city AS City;

* 1. Tìm tất cả các công dân sống tại thành phố Hồ Chí Minh

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address {city: 'HCM'})

RETURN c.name AS Citizen, a.street AS Street, a.district AS District;

* 1. Tìm tất cả địa chỉ mà Nguyễn Văn A từng sống

MATCH (a:Citizen {name: 'Nguyen Van A'})-[:LIVES\_AT]->(addr:Address)

RETURN addr.street AS Street, addr.district AS District, addr.city AS City;

* 1. Liệt kê các công dân và mã bưu điện nơi họ sinh sống

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address)

RETURN c.name AS Citizen, a.postal\_code AS PostalCode;

* 1. Tìm công dân sống ở địa chỉ "789 Dien Bien Phu"

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address {street: '789 Dien Bien Phu'})

RETURN c.name AS Citizen;

* 1. Liệt kê tất cả địa chỉ trong quận Bình Thạnh

MATCH (a:Address {district: 'Binh Thanh'})

RETURN a.street AS Street, a.city AS City;

* 1. Tìm tất cả các công dân có địa chỉ sống ở quận 1 hoặc quận 3

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address)

WHERE a.district IN ['1', '3']

RETURN c.name AS Citizen, a.street AS Street, a.district AS District;

* 1. Liệt kê tất cả địa chỉ ở thành phố Hồ Chí Minh với mã bưu điện '700000'

MATCH (a:Address {city: 'HCM', postal\_code: '700000'})

RETURN a.street AS Street, a.district AS District;

* 1. Tìm địa chỉ của Phạm Thị D

MATCH (d:Citizen {name: 'Pham Thi D'})-[:LIVES\_AT]->(a:Address)

RETURN a.street AS Street, a.district AS District, a.city AS City;

* 1. Liệt kê tất cả địa chỉ và số công dân sống ở đó

MATCH (a:Address)<-[:LIVES\_AT]-(c:Citizen)

RETURN a.street AS Street, COUNT(c) AS ResidentsCount;

* 1. Tìm các công dân sống trong cùng thành phố với Nguyễn Thị E

MATCH (e:Citizen {name: 'Nguyen Thi E'})-[:LIVES\_AT]->(addr:Address)

MATCH (c:Citizen)-[:LIVES\_AT]->(addr)

WHERE c.name <> e.name

RETURN c.name AS Citizen, addr.street AS Street;

* 1. Liệt kê tất cả các địa chỉ có số nhà bắt đầu bằng '123'

MATCH (a:Address)

WHERE a.street STARTS WITH '123'

RETURN a.street AS Street, a.district AS District;

* 1. Tìm địa chỉ của tất cả công dân tên bắt đầu bằng 'Nguyen'

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address)

WHERE c.name STARTS WITH 'Nguyen'

RETURN c.name AS Citizen, a.street AS Street, a.district AS District;

* 1. Tìm tất cả các công dân sống ở địa chỉ với mã bưu điện '700002'

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address {postal\_code: '700002'})

RETURN c.name AS Citizen, a.street AS Street;

* 1. Liệt kê tất cả công dân và địa chỉ nơi họ sinh sống, sắp xếp theo quận

MATCH (c:Citizen)-[:LIVES\_AT]->(a:Address)

RETURN c.name AS Citizen, a.street AS Street, a.district AS District

ORDER BY a.district;

1. CRUD Quản lý công dân
2. Thống kê và báo cáo
   1. Đếm tổng số công dân trong hệ thống

MATCH (c:Citizen)

RETURN count(c) AS total\_citizens;

* 1. Đếm số công dân theo giới tính

MATCH (c:Citizen)

RETURN c.gender, count(c) AS gender\_count;

* 1. Đếm số công dân theo nghề nghiệp

MATCH (c:Citizen)

RETURN c.occupation, count(c) AS occupation\_count;

* 1. Đếm số công dân theo quốc tịch

MATCH (c:Citizen)

RETURN c.nationality, count(c) AS nationality\_count;

* 1. Thống kê mức lương trung bình của các công việc

MATCH (j:Job)

RETURN avg(j.salary) AS average\_salary;

* 1. Đếm số công dân đã học một cấp độ bằng cấp cụ thể

MATCH (c:Citizen)-[:STUDIED\_AT]->(s:School)

WHERE s.degree = 'Bachelor'

RETURN count(c) AS bachelor\_degree\_count;

* 1. Thống kê các trường có nhiều sinh viên học nhất

MATCH (s:School)<-[:STUDIED\_AT]-(c:Citizen)

RETURN s.name, count(c) AS student\_count

ORDER BY student\_count DESC;

* 1. Đếm số công việc theo vị trí làm việc

MATCH (j:Job)

RETURN j.position, count(j) AS position\_count;

* 1. Tìm công việc có mức lương cao nhất

MATCH (j:Job)

RETURN j

ORDER BY j.salary DESC LIMIT 1;

* 1. Đếm số công dân sống ở từng địa chỉ

MATCH (a:Address)<-[:LIVES\_AT]-(c:Citizen)

RETURN a.street, count(c) AS resident\_count;

* 1. Đếm số công dân có cùng mức lương

MATCH (c:Citizen)-[:WORKS\_AT]->(j:Job)

RETURN j.salary, count(c) AS citizens\_count;

* 1. Thống kê số công dân là cha mẹ

MATCH (c:Citizen)-[:PARENT\_OF]->(child:Citizen)

RETURN count(c) AS parent\_count;

* 1. Thống kê số công dân có ít nhất một người bạn

MATCH (c:Citizen)-[:FRIEND\_OF]->(friend:Citizen)

RETURN count(DISTINCT c) AS citizens\_with\_friends;

* 1. Tìm công dân có nhiều bạn bè nhất

MATCH (c:Citizen)-[:FRIEND\_OF]->(friend:Citizen)

RETURN c.name, count(friend) AS friend\_count

ORDER BY friend\_count DESC LIMIT 1;

* 1. Tính số công dân đã học nhiều hơn 1 trường

MATCH (c:Citizen)-[:STUDIED\_AT]->(s:School)

WITH c, count(s) AS school\_count

WHERE school\_count > 1

RETURN count(c) AS multiple\_school\_citizens;

* 1. Đếm số công dân đã học cùng trường với con của họ

MATCH (p:Citizen)-[:PARENT\_OF]->(child:Citizen)-[:STUDIED\_AT]->(s:School)<-[:STUDIED\_AT]-(p)

RETURN count(DISTINCT p) AS parent\_child\_same\_school;

* 1. Tính tuổi trung bình của công dân

MATCH (c:Citizen)

RETURN avg(date().year - date(c.birthdate).year) AS average\_age;

* 1. Thống kê số công dân có số điện thoại bắt đầu bằng '09'

MATCH (c:Citizen)

WHERE c.phone STARTS WITH '09'

RETURN count(c) AS phone\_09\_count;

* 1. Đếm số công dân có email theo nhà cung cấp cụ thể (ví dụ: Gmail)

MATCH (c:Citizen)

WHERE c.email CONTAINS 'gmail.com'

RETURN count(c) AS gmail\_users;

* 1. Tìm các công dân có mối quan hệ cha mẹ - con cái với nhiều thế hệ

MATCH (p:Citizen)-[:PARENT\_OF]->(c:Citizen)-[:PARENT\_OF]->(gc:Citizen)

RETURN p.name, c.name, gc.name;

## Sao lưu và phục hồi cơ sở dữ liệu

Sao lưu

Bước 1: Chọn Project -> Tìm dấu 3 chấm ở kế bên nút Start, Open -> Ấn vào dấu 3 chấm

A screenshot of a computer

Description automatically generated

Bước 2: Chọn Dump để tạo ra file sao lưu, khi thành công sẽ có thông báo của Neo4J desktop như sau và có 1 file là neo4j.dump

A screenshot of a computer

Description automatically generated

Bước 3: Chọn vào dấu 3 chấm ở trên file neo4j.dump như sau và chọn Reveal in File Explorer thì Neo4J desktop sẽ đưa bạn đến nơi lưu trữ của file sao lưu

A screenshot of a computer

Description automatically generated

Bước 4: Bạn có thể lưu ở bất kỳ nơi khác hoặc chia sẻ cho bạn bè bằng cách cơ bản Ctrl + C và Ctrl + V với file neo4j.dump

A screenshot of a computer

Description automatically generated

Phục hồi

Bước 1: Tạo project mới và chọn Reveal files in File Explorer kế bên chữ I để hệ thống đưa bạn tới nơi lưu trữ file của project

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

Bước 2: Copy file neo4j.dump mà bạn có( có thể tử bạn bè,…) vào trong nơi lưu trữ trên và tắt đi File Explorer vì lúc này trong Neo4J desktop sẽ có file trên

A screenshot of a computer

Description automatically generated

Bước 3: Chọn dấu 3 chấm kê bên Open của file bạn vừa copy và chọn Create new DBMS from dump

A screenshot of a computer

Description automatically generated

Bước 4: Nhập thông tin căn bản mà bạn muốn và chọn Create để hệ thống tự phục hồi file

A screenshot of a computer

Description automatically generated

Bước 5: Chạy và kiểm tra dữ liệu

A computer screen shot of a network

Description automatically generated

# ỨNG DỤNG

Xây dựng ứng dụng bằng C# kết nối cơ sở dữ liệu đồ thị

PHẦN KẾT LUẬN

Sau quá trình nghiên cứu và thực hiện đề tài "Thiết kế CSDL Neo4j đồ thị Quản lý dữ liệu công dân và xây dựng ứng dụng minh họa", chúng em đã hoàn thành việc xây dựng mô hình cơ sở dữ liệu đồ thị, cung cấp một cách tiếp cận mới trong quản lý thông tin công dân. Thông qua việc ứng dụng hệ quản trị cơ sở dữ liệu Neo4j, hệ thống không chỉ đảm bảo khả năng lưu trữ thông tin một cách có tổ chức mà còn tối ưu hóa quá trình truy vấn và phân tích dữ liệu phức tạp, đặc biệt là các mối quan hệ giữa các đối tượng dữ liệu.

Bài nghiên cứu đã làm rõ các lợi ích của mô hình đồ thị trong việc quản lý dữ liệu có tính liên kết cao, đồng thời xây dựng ứng dụng minh họa giúp trực quan hóa và khai thác dữ liệu một cách hiệu quả. Tuy nhiên, đây mới chỉ là bước đầu, hệ thống vẫn còn nhiều tiềm năng phát triển và cải thiện trong tương lai, đặc biệt là khả năng mở rộng và tích hợp với các hệ thống khác nhằm phục vụ nhu cầu quản lý thông tin ngày càng đa dạng.

Kết quả của đề tài không chỉ đóng góp vào việc nâng cao hiệu quả quản lý dữ liệu công dân mà còn là minh chứng cho sự ứng dụng hiệu quả của công nghệ đồ thị vào các bài toán thực tế. Hy vọng rằng nghiên cứu này sẽ mở ra thêm nhiều hướng phát triển mới trong việc ứng dụng Neo4j vào các lĩnh vực khác, đồng thời giúp ích cho các bạn sinh viên và nhà nghiên cứu trong tương lai.

TÀI LIỆU THAM KHẢO