**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue and white logo with a book and a cap

Description automatically generated

**MÔN: NOSQL**

**QUẢN LÝ DANH SÁCH KHÁCH HÀNG VÀ ĐỊA CHỈ TƯƠNG ỨNG CỦA CHUỖI CỬA HÀNG THỰC PHẨM ORGANIC BẰNG CƠ SỞ DỮ LIỆU ĐỒ THỊ NEO4J**

Giảng viên hướng dẫn: Nguyễn Thị Thu Tâm

Sinh viên thực hiện: **Nhóm 04**

1. 2001220299 Lê Quân Bảo
2. 2001223708 Lê Hoàng Phú
3. 2001224327 Đinh Nhật Tấn
4. 2001220423 Nguyễn Thị Thu Cẩm

*TP. Hồ Chí Minh, tháng 10 năm 2025*

**BỘ CÔNG THƯƠNG**

**TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

A blue and white logo with a book and a cap

Description automatically generated

**MÔN: NOSQL**

**QUẢN LÝ DANH SÁCH KHÁCH HÀNG VÀ ĐỊA CHỈ TƯƠNG ỨNG CỦA CHUỖI CỬA HÀNG THỰC PHẨM ORGANIC BẰNG CƠ SỞ DỮ LIỆU ĐỒ THỊ NEO4J**

Giảng viên hướng dẫn: Nguyễn Thị Thu Tâm

Sinh viên thực hiện: **Nhóm 01**

1. 2001220299 Lê Quân Bảo
2. 2001223708 Lê Hoàng Phú
3. 2001224327 Đinh Nhật Tấn
4. 2001220423 Nguyễn Thị Thu Cẩm

*TP. Hồ Chí Minh, tháng 10 năm 2025*

MỤC LỤC

[MỤC LỤC i](#_Toc184856121)

[DANH MỤC HÌNH ẢNH ii](#_Toc184856122)

[NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN iii](#_Toc184856123)

[CHƯƠNG 1: HƯỚNG DẪN CÀI ĐẶT 1](#_Toc184856124)

[CHƯƠNG 2: HƯỚNG DẪN SỬ DỤNG 2](#_Toc184856125)

DANH MỤC HÌNH ẢNH

[Hình 1 Khởi động ứng dụng 2](#_Toc184856486)

[Hình 2 Nhập danh sách và chọn điểm khởi đầu 3](#_Toc184856487)

[Hình 3 Lộ trình tối ưu 4](#_Toc184856488)

[Hình 4 Lưu file html để xem lộ trình 4](#_Toc184856489)

[Hình 5 Thông báo lưu file 5](#_Toc184856490)

[Hình 6 Bản đồ có lộ trình 5](#_Toc184856491)

NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN

Hồ Chí Minh, tháng 10 năm 2025

Giảng viên hướng dẫn

Nguyễn Thị Thu Tâm

# LỜI CAM ĐOAN

# LỜI CẢM ƠN

# DANH MỤC TỪ VIẾT TẮT

# DANH MỤC ĐỊNH NGHĨA

# GIỚI THIỆU

## Giới thiệu Đồ án

Trong kỷ nguyên chuyển đổi số, việc quản lý và phân tích mối quan hệ giữa các đối tượng dữ liệu (như khách hàng, sản phẩm, địa điểm) đóng vai trò then chốt đối với sự phát triển của các mô hình kinh doanh, đặc biệt là chuỗi bán lẻ. Đồ án này tập trung vào giải quyết bài toán quản lý dữ liệu cho một **chuỗi cửa hàng thực phẩm Organic**, nơi dữ liệu không chỉ dừng lại ở các bảng đơn lẻ mà còn bao gồm các mạng lưới quan hệ phức tạp, chẳng hạn như mối liên kết đa tầng giữa **Khách hàng** và **Địa chỉ giao hàng** của họ, hay mối quan hệ giữa **Cửa hàng** và **Sản phẩm** đang được cung cấp.

Thay vì sử dụng Cơ sở dữ liệu Quan hệ (RDBMS) truyền thống—vốn thường yêu cầu các phép nối (JOIN) phức tạp và tốn kém khi truy vấn quan hệ sâu—đồ án đề xuất áp dụng **Cơ sở dữ liệu Đồ thị (Graph Database) Neo4j**. Neo4j cho phép mô hình hóa dữ liệu một cách trực quan thông qua các **Nút (Nodes)** và **Quan hệ (Relationships)**. Cách tiếp cận này giúp tối ưu hóa hiệu suất truy vấn các mối quan hệ đa bước (multi-hop) và mở ra khả năng phân tích nâng cao, từ việc phân tích hành vi mua sắm đến tối ưu hóa logistics dựa trên vị trí địa lý của khách hàng.

### Tên Đề tài

* **Tên chính thức:** Quản lý danh sách khách hàng và địa chỉ tương ứng của chuỗi cửa hàng thực phẩm Organic bằng Cơ sở dữ liệu đồ thị Neo4j.
* **Mục tiêu chính:** Xây dựng một hệ thống cơ sở dữ liệu phi quan hệ, sử dụng mô hình đồ thị để tối ưu hóa việc lưu trữ, truy vấn và phân tích các mối quan hệ phức tạp, đặc biệt là mối quan hệ giữa Khách hàng và vị trí địa lý.

### Bối cảnh Nghiệp vụ (Chuỗi cửa hàng Organic)

**Bối cảnh:** Chuỗi cửa hàng thực phẩm Organic đang mở rộng hoạt động và cần một hệ thống dữ liệu hiệu quả để quản lý Khách hàng và quy trình bán hàng.

**Vấn đề:** Các hệ thống CSDL Quan hệ (RDBMS) truyền thống gặp khó khăn khi truy vấn các mối quan hệ đa tầng (ví dụ: Khách hàng -> Phường -> Quận -> Thành phố)

hoặc khi cần phân tích mạng lưới quan hệ (ví dụ: Gợi ý sản phẩm, Tối ưu hóa giao hàng).

**Giải pháp:** Áp dụng Cơ sở dữ liệu Đồ thị Neo4j để mô hình hóa các thực thể và quan hệ một cách tự nhiên, giúp việc tìm kiếm và phân tích quan hệ trở nên nhanh chóng và trực quan.

## Mục tiêu nghiên cứu

### Mục tiêu Nghiệp vụ

* Quản lý thông tin chi tiết của các thực thể chính: Khách hàng, Sản phẩm, Danh mục, Hóa đơn, và các đối tác (Cửa hàng, Nhà cung cấp).
* Xây dựng mô hình dữ liệu linh hoạt, dễ dàng mở rộng khi chuỗi cửa hàng phát triển hoặc thay đổi cấu trúc địa lý.
* Hỗ trợ các truy vấn phân tích mạng lưới quan hệ, như: tìm kiếm Khách hàng theo vị trí địa lý sâu, phân tích hành vi mua sắm.

### Mục tiêu Kỹ thuật

* Nắm vững kiến thức nền tảng và cơ chế hoạt động của Cơ sở dữ liệu đồ thị Neo4j.
* Xây dựng thành công mô hình dữ liệu đồ thị (Graph Schema) phản ánh đúng các nghiệp vụ của chuỗi cửa hàng Organic.
* Thành thạo ngôn ngữ truy vấn Cypher để thực hiện các thao tác:
  + **CRUD** (Create, Read, Update, Delete) dữ liệu.
  + Thực hiện các truy vấn **Multi-hop** (truy vấn qua nhiều bước) phức tạp.
  + Sử dụng các hàm phân tích đồ thị cơ bản

## Phạm vi nghiên cứu

Phạm vi của đồ án tập trung vào việc xây dựng và thử nghiệm mô hình Cơ sở dữ liệu đồ thị Neo4j cho các nghiệp vụ cốt lõi của một chuỗi cửa hàng bán lẻ thực phẩm Organic, tập trung vào các mối quan hệ phức tạp.

### Phạm vi về Dữ liệu

Đồ án sẽ tập trung mô hình hóa các thực thể chính sau đây, sử dụng chúng làm các Nút (Nodes) trong đồ thị:

* **Khách hàng:** Thông tin tài khoản và cá nhân.
* **Sản phẩm & Danh mục:** Sản phẩm, Phân loại sản phẩm, Khuyến mãi.
* **Giao dịch & Tồn kho:** Hóa đơn, Cửa hàng, Nhà Cung cấp.
* **Địa lý:** Cấu trúc địa lý phân cấp (Thành phố, Quận, Phường) để xác định vị trí của Khách hàng và Cửa hàng.

### Phạm vi về Nghiệp vụ

* **Quản lý Quan hệ Phân cấp:** Mô hình hóa và truy vấn các mối quan hệ địa lý sâu (ví dụ: truy vấn tất cả sản phẩm thuộc 1 danh mục trong 1 cửa hàng nhất định).
* **Quản lý Giao dịch:** Xử lý việc tạo Hóa đơn và mô hình hóa các mặt hàng trong Hóa đơn (sử dụng thuộc tính trên quan hệ).
* **Quản lý Tồn kho Cơ bản:** Áp dụng logic để cập nhật số lượng tồn kho của Sản phẩm tại Cửa hàng.
* **Truy vấn Phân tích Đồ thị:** Thực hiện các truy vấn phức tạp đa bước (multi-hop) để tìm ra các mô hình và mối quan hệ ẩn (ví dụ: đề xuất sản phẩm, tìm kiếm sản phẩm theo danh mục).

### Phạm vi về Công nghệ

Đồ án được thực hiện trên môi trường Cơ sở dữ liệu Đồ thị **Neo4j**, sử dụng ngôn ngữ truy vấn **Cypher**. Để minh họa và tương tác trực quan với dữ liệu, nhóm đã phát triển một **Ứng dụng Web** theo kiến trúc **MVC (Model-View-Controller):** ASP.NET MVC. Ứng dụng này đóng vai trò là giao diện để thực hiện các thao tác quản trị dữ liệu (CRUD) và hiển thị kết quả truy vấn từ Neo4j, từ đó chứng minh tính khả thi của mô hình.

# TỔNG QUAN VỀ CƠ SỞ DỮ LIỆU ĐỒ THỊ VÀ NEO4J

## Cơ sở Dữ liệu Đồ thị và Mô hình Property Graph

Cơ sở dữ liệu Đồ thị là một loại **CSDL phi quan hệ (NoSQL)** được thiết kế chuyên biệt để mô hình hóa, lưu trữ và truy vấn các mối quan hệ. CSDL Đồ thị tập trung vào **các mối liên kết (Relationships)** giữa các thực thể, biến chúng thành các đối tượng cấp một (first-class citizens) trong mô hình dữ liệu.

### Các thành phần cốt lõi (Mô hình Property Graph)

* **Nút (Nodes):** Đại diện cho các **thực thể** chính trong hệ thống. Mỗi Nút được xác định duy nhất và có thể chứa nhiều thuộc tính.
  + *Ví dụ:* Nút KhachHang, nút SanPham.
* **Quan hệ (Relationships):** Đại diện cho **mối liên kết** hoặc tương tác **có hướng** giữa hai nút (nút bắt đầu và nút kết thúc). Mỗi Quan hệ phải có **một kiểu (Type)** và **một hướng (Direction)**.
  + *Ví dụ:* Quan hệ [TAO\_HOADON], [THUOC\_DANHMUC], [CUNG\_CAP].
* **Tính chất:** Tương tự như Nút, Quan hệ cũng là đối tượng và có thể có các **Thuộc tính** riêng, giúp mô tả chi tiết ngữ cảnh của mối liên kết.
* **Thuộc tính (Properties):** Là các cặp khóa-giá trị (Key-Value pairs) lưu trữ **dữ liệu chi tiết** về Nút hoặc Quan hệ.
  + *Ví dụ:* Nút KhachHang có thuộc tính HoTen, Quan hệ [CHUA\_SANPHAM] có thuộc tính SoLuong.
* **Nhãn (Labels):** Là các thẻ đặt tên (Named Tags) được sử dụng để **phân loại** các Nút hoặc Quan hệ. Nhãn là cơ sở để lập chỉ mục (Indexing) và giúp nhóm các thực thể cùng loại.

### Khác biệt cốt giữa CSDL Đồ thị so với RDBMS

Sự khác biệt lớn nhất và là lợi thế cạnh tranh của CSDL Đồ thị so với RDBMS nằm ở cách chúng xử lý các truy vấn quan hệ sâu (Multi-hop Query)

|  |  |  |
| --- | --- | --- |
| **Tính năng** | **CSDL Quan hệ (RDBMS)** | **CSDL Đồ thị (Neo4j)** |
| **Mục tiêu lưu trữ** | Tối ưu hóa việc lưu trữ dữ liệu *tách biệt* và *tính toàn vẹn*. | Tối ưu hóa việc lưu trữ *mối quan hệ* và *duyệt đồ thị*. |
| **Cơ chế truy vấn** | Sử dụng phép **JOIN** giữa các bảng thông qua Khóa ngoại. | Sử dụng cơ chế **Traversal (Duyệt đồ thị)** theo các con trỏ vật lý (Physical Pointers). |
| **Hiệu suất (Quan hệ Sâu)** | **Giảm theo cấp số nhân** khi số lượng JOIN tăng (Multi-hop Query). Thời gian truy vấn phụ thuộc vào kích thước toàn bộ CSDL. | **Ổn định (Constant Time Complexity)**. Hiệu suất chỉ phụ thuộc vào số lượng nút và quan hệ được duyệt, không phụ thuộc vào kích thước toàn bộ CSDL. |
| **Tính linh hoạt** | Mọi thay đổi cấu trúc quan hệ (Schema) đều **tốn kém** và yêu cầu thời gian ngừng hoạt động (downtime). | Cấu trúc đồ thị cực kỳ linh hoạt, cho phép thêm nút hoặc quan hệ mới ngay lập tức mà không cần định hình lại toàn bộ CSDL. |

## Ứng dụng Thực tiễn của CSDL Đồ thị

Khả năng mô hình hóa và truy vấn quan hệ sâu làm cho mô hình đồ thị trở thành giải pháp lý tưởng cho nhiều lĩnh vực:

* **Khoa học Xã hội và Mạng lưới:** Tìm ra những người có tầm ảnh hưởng (bậc của đỉnh). CSDL đồ thị được sử dụng để tìm **bậc của đỉnh** hay **số lượng cạnh nối với đỉnh đó**.
* **Nghiên cứu Sinh học:** Các thành phần sinh học (protein, phân tử, gen) và các tương tác của chúng tạo nên một **đồ thị sinh học**. Dựa vào đó, có thể tìm hiểu quá trình trao đổi chất trong cơ thể.
* **Bài toán Tìm đường đi (Pathfinding):** Tìm đường đi ngắn nhất giữa hai điểm. Ứng dụng quan trọng trong tối ưu hóa giao hàng và logistics.
* **Web Search (PageRank):** Thuật toán PageRank được dùng trong Google Search để xếp hạng các trang web.

# HỆ QUẢN TRỊ CƠ SỞ DỮ LIỆU NEO4J

## TỔNG QUAN HỆ QUẢN TRỊ NEO4J

Neo4j là một hệ quản trị CSDL đồ thị được phát triển bởi công ty Neo4j, sử dụng ngôn ngữ Cypher để tạo, cập nhật và truy vấn dữ liệu đồ thị.

* **Phiên bản:** Neo4j có hai phiên bản: **Community** (mã nguồn mở) và **Enterprise** (thương mại).
* **Kết nối:** Cung cấp các driver để kết nối với các ứng dụng lập trình trên nhiều ngôn ngữ khác nhau (Java, .Net, Python).

## NGÔN NGỮ TRUY VẤN CYPHER

Cypher là ngôn ngữ truy vấn chính của Neo4j, được phát triển để làm cho việc truy vấn đồ thị trở nên trực quan và hiệu quả.

### Kiểu dữ liệu trong Cypher

* **Property Types:** Dùng cho thuộc tính. Bao gồm **Number** (Integer, Float), **String**, **Boolean**, và các kiểu **Temporal** (Date, Time, DateTime, Duration, v.v.).
* **Structural Types:** Dùng cho cấu trúc đồ thị. Bao gồm **Node** và **Relationship** (mỗi loại đều có Id, Type/Label(s), và Map thuộc tính).
* **Composite Types:** Các cấu trúc phức hợp. Bao gồm **List**, **Map** (tập hợp cặp Key-Value), và **Path** (một chuỗi luân phiên các nút và quan hệ).

### Quy tắc đặt tên trong Cypher

Quy cách đặt tên các đối tượng (nút, quan hệ, biến) cần tuân thủ:

* Bắt đầu bằng ký tự alphabet.
* Được chứa dấu gạch dưới (\_).
* Không được bắt đầu bằng số.
* Phân biệt chữ hoa, chữ thường.
* **Ngoại lệ:** Ký tự $ đứng đầu được sử dụng để định danh tham số (parameter).

### Cú pháp hình ảnh (ASCII-Art)

Cypher sử dụng cú pháp hình ảnh độc đáo, mô phỏng trực quan cấu trúc đồ thị:

* **Nút (Nodes):** Được biểu diễn bằng dấu ngoặc đơn (tên\_biến:Nhãn). Nút không có tên biến là nút ẩn danh.
* **Quan hệ (Relationships):**
  + **Có hướng:** -[r:TYPE]-> hoặc <-[r:TYPE]-.
  + **Vô hướng:** -[:TYPE]-.
* **Thuộc tính (Properties):** Được biểu diễn bằng dấu ngoặc nhọn {name:value}

### Các lệnh truy vấn chính (CRUD và Traversal)

Cypher sử dụng các lệnh chính sau để thao tác trên đồ thị:

* **MATCH:** Dùng để tìm kiếm một mẫu đồ thị.
* **WHERE:** Dùng để lọc kết quả.
* **RETURN:** Dùng để trả về kết quả.
* **CREATE:** Thêm nút hoặc quan hệ mới.
  + *Ví dụ:* CREATE (p:Person {name: 'Andy'}).
* **MERGE:** Tìm hoặc tạo nút/quan hệ (thao tác tối ưu, thường dùng để kiểm tra tính duy nhất hoặc cập nhật tồn kho).
* **SET/REMOVE:** Dùng để thêm/sửa (SET) hoặc xóa (REMOVE) thuộc tính của nút/quan hệ.
* **DELETE/DETACH DELETE:** Dùng để xóa quan hệ (DELETE r) hoặc xóa nút cùng với tất cả các quan hệ liên kết (DETACH DELETE n).

# CÀI ĐẶT VÀ CẤU HÌNH NEO4J

## Cài đặt Môi trường Neo4j

### Neo4j Desktop (hoặc Neo4j Aura)

Neo4j Desktop là ứng dụng quản lý đồ thị local cho phép người dùng tạo, quản lý và khởi động nhiều phiên bản CSDL Neo4j (Graph instances).

* **Tải về và Cài đặt:** Tải Neo4j Desktop từ trang chủ chính thức, sau đó tiến hành cài đặt .
* **Tạo Database mới:** Sau khi cài đặt, người dùng tạo một dự án (Project) mới và thêm một cơ sở dữ liệu đồ thị cục bộ (Local Graph) với phiên bản Community Edition.
  + Cần đặt tên cho database (Ví dụ: organic-store).
  + Thiết lập mật khẩu quản trị ban đầu (neo4j).
* **Khởi động Database:** Nhấn nút **Start** để khởi động dịch vụ Neo4j. Database sẽ chạy trên cổng mặc định 7687 (Giao thức Bolt).

### Kết nối Neo4j Browser và tạo Graph

* **Neo4j Browser** là giao diện người dùng dựa trên web, được sử dụng để tương tác trực tiếp với CSDL thông qua ngôn ngữ Cypher.
* **Truy cập Browser:** Sau khi database khởi động, nhấn nút **Open** (hoặc truy cập bolt://localhost:7687) để mở Neo4j Browser.
* **Tạo Database Logic (nếu cần):** Trong một số trường hợp, lệnh tạo database mới được thực hiện trong Browser. Tuy nhiên, nếu đã tạo qua Desktop, Browser sẽ tự động kết nối và sẵn sàng nhận lệnh Cypher.

## Các công cụ hỗ trợ truy vấn (GUI Tool)

### Giới thiệu Neo4j Browser

Neo4j Browser là công cụ chính để phát triển Cypher và quản lý CSDL.

Tính năng: Cho phép nhập các lệnh Cypher, hiển thị kết quả dưới dạng bảng hoặc đồ thị trực quan, và lưu trữ các truy vấn yêu thích. Nó tương tự như các công cụ GUI khác (ví dụ: công cụ quản lý MongoDB) nhưng được tối ưu hóa cho mô hình đồ thị.

Sử dụng: Thông tin kết nối (URI) được hiển thị rõ ràng (ví dụ: bolt://localhost:7687). Bạn có thể gõ lệnh :server status; trong Browser để kiểm tra trạng thái hoạt động của máy chủ.

### Sử dụng Bloom (nếu có) để trực quan hóa đồ thị

Neo4j Bloom là công cụ bổ sung mạnh mẽ, tập trung vào việc trực quan hóa đồ thị phức tạp. Bloom cho phép người dùng khám phá các mối quan hệ đa bước thông qua giao diện đồ họa, giúp dễ dàng nhận diện các mẫu (patterns) và kết nối ẩn trong dữ liệu mà truy vấn dạng văn bản khó thể hiện.

## Kết nối Ứng dụng với Neo4j

# MÔ HÌNH HÓA VÀ CẤU TRÚC DỮ LIỆU ĐỒ ÁN

## Phân tích Yêu cầu Nghiệp vụ

### Yêu cầu quản lý Khách hàng và Địa chỉ

Đây là quy trình xác định và duy trì thông tin khách hàng, vốn là nền tảng cho mọi giao dịch sau này.

|  |  |  |
| --- | --- | --- |
| **Bước/Quy trình** | **Mô tả nghiệp vụ thực tế** | **Yêu cầu Dữ liệu Cốt lõi** |
| **Đăng ký & Định danh** | Khách hàng (KH) cung cấp thông tin cá nhân cơ bản (HoTen, Email, SĐT) để tạo tài khoản. Khách hàng được phân hạng (LoaiKH). | Thông tin cá nhân KH, Phân hạng thành viên. |
| **Xác định Vị trí** | KH chỉ định địa chỉ giao hàng chính (DiaChiCuThe). Dữ liệu này cần được chuẩn hóa và liên kết với cấu trúc hành chính **(Phường, Quận, Thành phố)** để tối ưu hóa tuyến giao hàng và phân tích thị trường. | Cấu trúc địa lý phân cấp (TP $\rightarrow$ Quận $\rightarrow$ Phường), Mối quan hệ địa lý của KH. |
| **Phân tích Thị trường** | Quản lý cần biết Khách hàng VIP tập trung ở khu vực nào (Quận, Phường) để mở rộng dịch vụ hoặc cửa hàng. | Cần khả năng truy vấn KH theo vị trí sâu. |

### Yêu cầu quản lý Bán hàng (Hóa đơn, Sản phẩm, Tồn kho)

Quy trình này mô tả cách hàng hóa được lưu trữ, bán ra và ghi nhận trong một chuỗi cửa hàng bán lẻ.

|  |  |  |
| --- | --- | --- |
| **Bước/Quy trình** | **Mô tả nghiệp vụ thực tế** | **Yêu cầu Dữ liệu Cốt lõi** |
| **Nhập hàng (Sourcing)** | Nhà cung cấp gửi hàng đến kho, lập Phiếu nhập. Hàng hóa được gán mã và mô tả (SanPham). | NhaCungCap, PhieuNhap, SanPham, Thuộc tính DonGiaNhap (trên Quan hệ nhập hàng). |
| **Phân phối & Tồn kho** | Sản phẩm được phân phối đến các **Cửa hàng** khác nhau. Mỗi cửa hàng chịu trách nhiệm quản lý số lượng hàng hóa có sẵn. | CuaHang, SanPham. **Số lượng tồn kho (SoTon)** phải được lưu trữ theo Cửa hàng. |
| **Bán hàng (Giao dịch)** | KH đặt hàng (tạo HoaDon). Hệ thống cần ghi nhận tổng tiền, thời gian, và quan trọng là chi tiết các mặt hàng. | HoaDon, KhachHang. Chi tiết đơn hàng (mặt hàng và **SoLuong**). |
| **Quản lý Đơn hàng** | Ghi nhận trạng thái thanh toán, địa chỉ giao hàng cuối cùng, và các ghi chú liên quan đến đơn hàng. | TrangThaiThanhToan, DiaChiGiaoHang, GhiChu. |

## Thiết kế Mô hình Đồ thị (Graph Schema)

### Sơ đồ Đồ thị tổng thể.

(chèn hình vào Sơ đồ Mô hình Đồ thị)

### Đặc tả các Nút (Nodes) chính

|  |  |  |
| --- | --- | --- |
| **Nhãn (Label)** | **Thuộc tính (Properties) cốt lõi** | **Mục đích Nghiệp vụ** |
| **KhachHang** | MaKH, HoTen, TenDangNhap, Email, SoDienThoai, LoaiKH | Định danh người tiêu dùng. |
| **SanPham** | MaSP, TenSP, GiaBan, MoTa, DonVi | Quản lý danh mục hàng hóa. |
| **HoaDon** | MaHD, ThanhTien, NgayTao, TrangThaiThanhToan | Ghi nhận giao dịch mua hàng. |
| **CuaHang** | MaCH, TenCH, DiaChiCuThe | Quản lý điểm bán hàng và tồn kho phân tán. |
| **ThanhPho/Quan/Phuong** | Mã và Tên | Hỗ trợ phân tích địa lý và logistics. |

((CHÈN HÌNH Vị trí: Giữa 4.2.2. Đặc tả các Nút (Nodes) chính và 4.2.3. Đặc tả các Quan hệ (Relationships) quan trọng. Nội dung hình ảnh: Một hình ảnh đơn giản cho thấy kết quả của lệnh MATCH (n) RETURN n LIMIT 10 hoặc một sơ đồ nhỏ mô tả cấu trúc địa lý phân cấp .Gợi ý: Kết quả trực quan hóa của lệnh MATCH (kh:KhachHang {MaKH:'KH005'})-[\*1..2]-(n) RETURN \* có thể làm nổi bật các mối quan hệ đầu tiên (Hóa đơn, Địa lý) của một khách hàng cụ thể.))

### Đặc tả các Quan hệ (Relationships)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Kiểu Quan hệ (Type)** | **Nút Bắt đầu** | **Nút Kết thúc** | **Thuộc tính (Properties)** | **Mục đích Nghiệp vụ** |
| **[:CUNG\_CAP]** | CuaHang | SanPham | **SoTon** | Quản lý **Tồn kho** theo từng địa điểm Cửa hàng. |
| **[:CHUA\_SANPHAM]** | HoaDon | SanPham | **SoLuong** | Chi tiết **Sản phẩm đã mua** trong mỗi Hóa đơn. |
| **[:THUOC\_DANHMUC]** | SanPham | DanhMuc | (Không có) | Phân loại sản phẩm. |
| **[:TAO\_HOADON]** | KhachHang | HoaDon | (Không có) | Liên kết Khách hàng với lịch sử giao dịch. |

# KHỞI TẠO VÀ TRUY VẤN DỮ LIỆU (CYPHER)

## Lệnh Khởi tạo Dữ liệu

Phần này sử dụng các lệnh **CREATE** và **MATCH & CREATE** để thiết lập môi trường dữ liệu và các mối quan hệ cốt lõi đã được mô hình hóa ở Chương 4.

## Lệnh CREATE tạo các Node mẫu.

Khởi tạo các Nút thực thể chính của hệ thống.

* Nút Địa lý Phân cấp:

CREATE (:ThanhPho {MaTP:"79", TenTP:"TP. Hồ Chí Minh"}),

(:Quan {MaQuan:"037", TenQuan:"Quận 1"}),

(:Phuong {MaPhuong:"03702", TenPhuong:"Bến Nghé"});

* Nút Danh mục và Sản phẩm (Ví dụ):

CREATE (:DanhMuc {MaDM: "DM02", TenDM: "Trái Cây"});

CREATE (:SanPham {MaSP:"SP011", TenSP:"Táo Fuji hữu cơ", DonVi:"Kg", GiaBan:120000, MoTa:"Giòn ngọt"});

* Nút Khách hàng (Ví dụ):

CREATE (:KhachHang {MaKH:"KH005", HoTen:"Ngô Thị Diễm", TenDangNhap:"ngodiem", MatKhau:"xyz789", Email:"diem@mail.com", SoDienThoai:"0938000005", NgaySinh:date("1993-07-12"), LoaiKH:"VIP"});

### Lệnh MATCH & CREATE tạo các Relationship.

Liên kết các Nút lại với nhau, bao gồm việc sử dụng thuộc tính trên quan hệ.

* Liên kết Sản phẩm - Danh mục:

MATCH (sp:SanPham), (dm:DanhMuc)

WHERE sp.MaSP >= "SP011" AND sp.MaSP <= "SP020" AND dm.MaDM = "DM02"

CREATE (sp)-[:THUOC\_DANHMUC]->(dm);

* Liên kết Tồn kho (CuaHang - SanPham):

MATCH (ch:CuaHang {MaCH:"CHHCM01"}), (sp:SanPham)

WHERE sp.MaSP >= "SP011" AND sp.MaSP <= "SP030"

CREATE (ch)-[:CUNG\_CAP {SoTon: 8}]->(sp);

* Liên kết Chi tiết Hóa đơn (HoaDon - SanPham):

MATCH (hd:HoaDon {MaHD:"HD005"}), (sp:SanPham)

WHERE sp.MaSP IN ["SP015","SP035"]

CREATE (hd)-[:CHUA\_SANPHAM {SoLuong:2}]->(sp);

## Truy vấn Nghiệp vụ Cơ bản (CRUD)

### Tìm kiếm Sản phẩm theo Danh mục và Cửa hàng (Multi-hop Query Cốt lõi)

MATCH (ch:CuaHang {MaCH:"CHHCM01"})-[r:CUNG\_CAP]->(sp:SanPham)-[:THUOC\_DANHMUC]->(dm:DanhMuc {MaDM:"DM02"})

RETURN sp.TenSP, dm.TenDM, ch.TenCH, r.SoTon

***Ý nghĩa:*** Tìm tên và tồn kho (r.SoTon) của các sản phẩm thuộc Danh mục Trái Cây (DM02) **VÀ** được cung cấp bởi Cửa hàng Organic Quận 1 (CHHCM01).

### Cập nhật tồn kho (sử dụng MERGE và SET)

* Cập nhật Tồn kho đơn giản (SET):

MATCH (ch:CuaHang {MaCH:"CHHN01"})-[r:CUNG\_CAP]->(sp:SanPham)

WHERE sp.MaSP = 'SP001'

SET r.SoTon = r.SoTon - 1

RETURN r.SoTon

* + *Ý nghĩa:* Giảm số lượng tồn kho của "Cà rốt Đà Lạt" (SP001) tại cửa hàng Hà Nội (CHHN01) đi 1 đơn vị sau khi bán.
* Cập nhật Tồn kho sau khi nhập hàng (MERGE):

MATCH (pn:PhieuNhap {MaPN:"PN002"})-[nhap:NHAP\_SANPHAM]->(sp:SanPham),

(pn)-[:NHAP\_CHO\_CUAHANG]->(ch:CuaHang {MaCH:"CHHN01"})

MERGE (ch)-[r:CUNG\_CAP]->(sp)

ON MATCH SET r.SoTon = r.SoTon + nhap.SoLuong

RETURN r.SoTon

* + *Ý nghĩa:* Tự động cập nhật số lượng tồn kho (r.SoTon) cho sản phẩm được liệt kê trong Phiếu nhập (PN002) tại Cửa hàng Hà Nội.

## Truy vấn Phân tích Đồ thị (Nâng cao)

### Phân tích Khách hàng và vị trí

Truy vấn này thể hiện việc duyệt qua cấu trúc địa lý Multi-hop để phục vụ mục đích phân tích thị trường.

* Truy vấn Đa bước Địa lý (Ví dụ): Tìm Khách hàng VIP ở Phường Bến Nghé, Quận 1.

MATCH (kh:KhachHang {LoaiKH: 'VIP'})

MATCH (ph:Phuong {TenPhuong: 'Bến Nghé'})

WHERE kh.DiaChiCuThe CONTAINS ph.TenPhuong // Giả định Khách hàng được liên kết với Phường

RETURN kh.HoTen, kh.Email, ph.TenPhuong

### Đề xuất sản phẩm (Collaborative Filtering cơ bản)

Tìm kiếm các sản phẩm được Khách hàng có hành vi tương tự mua, nhưng Khách hàng mục tiêu chưa mua (Đề xuất dựa trên cộng đồng).

// Bước 1: Xác định sản phẩm mà KH005 (Ngô Thị Diễm) đã mua

MATCH (target:KhachHang {MaKH: 'KH005'})-[:TAO\_HOADON]->()-[chua1:CHUA\_SANPHAM]->(sp1:SanPham)

WITH target, collect(sp1) AS target\_bought

// Bước 2: Tìm kiếm sản phẩm mà khách hàng khác mua nhưng KH005 chưa mua

MATCH (other:KhachHang)-[:TAO\_HOADON]->()-[chua2:CHUA\_SANPHAM]->(sp2:SanPham)

WHERE other.MaKH <> target.MaKH

AND NOT sp2 IN target\_bought

// Bước 3: Đếm và xếp hạng các sản phẩm đề xuất

RETURN sp2.TenSP, count(sp2) AS buy\_count

ORDER BY buy\_count DESC

LIMIT 5

* *Ý nghĩa:* Truy vấn này duyệt qua mạng lưới giao dịch để tìm 5 sản phẩm phổ biến nhất mà những khách hàng khác đã mua nhưng Khách hàng Ngô Thị Diễm chưa từng mua, làm cơ sở cho tính năng gợi ý sản phẩm.

# ỨNG DỤNG THỰC TẾ TRÊN NỀN TẢNG CÔNG NGHỆ

26 5.1. Cơ chế kết nối Ứng dụng với Neo4j

26 5.1.1. Lựa chọn ngôn ngữ/driver (Ví dụ: C# .NET Driver).

27 5.1.2. Minh họa đoạn mã kết nối cơ bản.

28 5.2. Vai trò của Neo4j đối với Hệ thống Organic Store

28 5.2.1. Vai trò trong việc phân tích các mối quan hệ địa lý phức tạp.

28 5.2.2. Hiệu quả trong việc thực hiện các tính năng đề xuất.

# KẾT LUẬN, ĐÁNH GIÁ VÀ HƯỚNG PHÁT TRIỂN

# BẢNG PHÂN CHIA CÔNG VIỆC