**GRAPH DATABASE & NEO4J & CYPHER**

**MỤC LỤC**

[CHƯƠNG 1. Giới thiệu về graph database 4](#_Toc474767409)

[1.1. Khái niệm 4](#_Toc474767410)

[1.2. Sử dụng khi nào 5](#_Toc474767411)

[1.3. Điểm nổi bật 6](#_Toc474767412)

[1.3.1. Hiệu suất 6](#_Toc474767413)

[1.3.2. Linh động, mềm dẻo 6](#_Toc474767414)

[1.3.3. Nhanh 7](#_Toc474767415)

[CHƯƠNG 2. Giới thiệu neo4j 8](#_Toc474767416)

[2.1. Neo4j là gì? 8](#_Toc474767417)

[2.2. Tính năng 8](#_Toc474767418)

[2.3. Ưu điểm nổi bật 9](#_Toc474767419)

[2.4. Kiến trúc 10](#_Toc474767420)

[2.4.1. Lưu trữ dữ liệu 10](#_Toc474767421)

[2.5. Cài đặt 11](#_Toc474767422)

[2.5.1. Yêu cầu hệ thống 11](#_Toc474767423)

[2.5.2. Thiết lập cấu hình 11](#_Toc474767424)

[Mặc định khi cài Neo4j: 11](#_Toc474767425)

[2.5.3. Quyền 14](#_Toc474767426)

[2.5.4. Cài đặt 14](#_Toc474767427)

[2.5.5. Thiết đặt các kết nối tới Neo4j 14](#_Toc474767428)

[2.5.6. Thời hạn transaction 17](#_Toc474767429)

[2.6. Bảo mật 17](#_Toc474767430)

[2.6.1. Cấu hình cơ bản 17](#_Toc474767431)

[2.6.2. Xác thực người dùng 17](#_Toc474767432)

[2.6.3. Thiết đặt HTTPS 18](#_Toc474767433)

[2.6.4. Thiết đặt các quy tắc ủy quyền truy cập Neo4j 18](#_Toc474767434)

[2.6.5. Sử dụng Proxy 19](#_Toc474767435)

[2.7. Import 20](#_Toc474767436)

[2.8. Sao lưu, dự phòng 21](#_Toc474767437)

[2.8.1. Thiết đặt 21](#_Toc474767438)

[2.8.2. Thực thi sao lưu 21](#_Toc474767439)

[2.8.3. Khôi phục dữ liệu 21](#_Toc474767440)

[2.9. Giám sát 21](#_Toc474767441)

[2.10. Hiệu suất 21](#_Toc474767442)

[CHƯƠNG 3. Giới thiệu cypher 22](#_Toc474767443)

[3.1. Khái niệm 22](#_Toc474767444)

[3.2. Cú pháp 22](#_Toc474767445)

[3.2.1. Create 23](#_Toc474767446)

[3.2.2. Match return 25](#_Toc474767447)

[3.2.3. Set 26](#_Toc474767448)

[3.2.4. Delete 27](#_Toc474767449)

[3.2.5. Remove 27](#_Toc474767450)

[3.2.6. Cú pháp khác 27](#_Toc474767451)

[CHƯƠNG 4. Quản lý dữ liệu 30](#_Toc474767452)

[4.1. Các bước tạo dữ liệu 30](#_Toc474767453)

[4.2. Import dữ liệu 30](#_Toc474767454)

# Giới thiệu về graph database

## Khái niệm

Graph Databases: Là một dạng cơ sở dữ liệu NoSQL được thiết kế riêng cho việc lưu trữ thông tin dạng đồ thị gồm các tập điểm nối với nhau bởi các cạnh trên mỗi đỉnh, cạnh đều có những thuộc tính riêng lưu trữ dạng key - value và không bắt buộc đồng nhất. Một số sản phẩm tiêu biểu như: Neo4j.

Đồ thị là một tập các đối tượng gọi là đỉnh nối với nhau bởi các cạnh. Thông thường, đồ thị được vẽ dưới dạng một tập các điểm (đỉnh, nút) nối với nhau bởi các đoạn thẳng (cạnh). Tùy theo ứng dụng mà một số cạnh có thể có hướng.

Node là một đối tượng trong Graph database. Nó có thể tìm ra dựa vào id, nhưng nếu nó không có mối liên hệ trong Graph thì nó tồn tại không có ý nghĩa.

Label là tên gọi cho các nhóm Node. Tất cả các node cùng một nhãn sẽ trở thành một tập hợp. Nhiều của truy vấn có thể áp dụng cho cả tập hợp này thay vì toàn bộ các node trong đồ thị

Relations là mối liên hệ, là một tên đại diện cho mối quan hệ giữa các Nodes với nhau. Các mối quan hệ có hướng. Điểm đầu là node, điểm cuối cũng là 1 node.

Properties là những thuộc tính của một đối tượng hoặc một mối liên hệ, dùng để thể hiện hay làm rõ bản chất của đối tượng đó. Thể hiện dạng key – value. Các kiểu giá trị: numberic, string, boolean.

| **Kiểu** | **Mô tả** | **Miền giá trị** |
| --- | --- | --- |
| boolean |  | true/false |
| byte | 8-bit integer | -128 to 127, inclusive |
| short | 16-bit integer | -32768 to 32767, inclusive |
| int | 32-bit integer | -2147483648 to 2147483647, inclusive |
| long | 64-bit integer | -9223372036854775808 to 9223372036854775807, inclusive |
| float | 32-bit IEEE 754 floating-point number |  |
| double | 64-bit IEEE 754 floating-point number |  |
| char | 16-bit unsigned integers representing Unicode characters | u0000 to uffff (0 to 65535) |
| String | sequence of Unicode characters |  |

Một số định nghĩa liên quan trong NoSQL mà Graph cũng có:

* Relational (Rằng buộc) thuật ngữ sử dụng đến các mối quan hệ giữa các bảng trong cơ sở dữ liệu quan hệ (RDBMs) sử dụng mô hình khóa gồm 2 loại khóa: khóa chính và khóa phụ (primary key + foreign key) để ràng buộc dữ liệu nhằm thể hiện tính nhất quán dữ liệu từ các bảng khác nhau.
* Non-relational (Không rằng buộc): là khái niệm không sử dụng các ràng buộc dữ liệu cho nhất quán dữ liệu ở NoSQL database.
* High Availability (Tính sẵn sàng): Do chấp nhận sự trùng lặp trong lưu trữ nên nếu một node (commodity machine) nào đó bị chết cũng không ảnh hưởng tới toàn bộ hệ thống
* High Scalability (Khả năng mở rộng): Gần như không có một giới hạn cho dữ liệu và người dùng trên hệ thống.

## Sử dụng khi nào

 Chúng ta đã quá quen với cách biểu diễn theo từng quan hệ truyền thống nhưng với những bài toán cần nhiều quan hệ việc sử dụng Relation Database để lưu trữ không phải là một giải pháp hay vì:

* Việc biểu diễn quan hệ dưới dạng bảng không phải là một cách làm trực quan.
* Các phép kết bảng thường tốn rất nhiều chi phí, nhất là với lượng dữ liệu phức tạp, lớn hoặc cực lớn.

Graph Database liên kết trực tiếp giữa các thực thể (các đỉnh). Do đó Graph giúp trả lời rất nhiều câu hỏi liên quan đến truy vấn dữ liệu một cách hiệu quả hơn so với Relation Database, đồng thời khả năng trực quan hoá dữ liệu của Graph so với Relation Database cũng tốt hơn rất nhiều.

Đặc biệt, khác với các kiểu lưu trữ dữ liệu khác, các mối quan hệ được ưu tiên hàng đầu trong Graph Database nên rất phù hợp cho các bài toán mang tính kết nối và nhiều quan hệ giữa các thực thể, đối tượng như mạng xã hội.

## Điểm nổi bật

### Hiệu suất

Với một ứng dụng đòi hỏi làm việc với dữ liệu có kết nối với hiệu suất cao thì thì Graph Database là một lựa chọn hoàn toàn hợp lý thay vì CSDL quan hệ hay các CSDL NoSQL khác. Đối với CSDL quan hệ, khi các tập dữ liệu ngày càng lớn thì việc thực hiện các truy vấn sâu và phải kết hợp nhiều bảng sẽ rất tốn thời gian lại kém hiệu quả bởi các phép join. Nhưng với Graph Database thì đây không phải là vấn đề phải quan tâm nhiều, vì các truy vấn đã được nội bộ hóa thành một phần của đồ thị. Như vậy, thời gian thực hiện truy vấn tỷ lệ với kích thước của một phần đồ thị được duyệt để đáp ứng truy vấn chứ không phụ thuộc vào kích thước của toàn bộ đồ thị. Do vậy dù kích thước của cả tập dữ liệu có tăng lên thì thời gian thực hiện truy vấn vẫn được đảm bảo.

### Linh động, mềm dẻo

Tính linh động là một đặc trưng thấy rõ và khác biệt nhất của Graph Database so với các CSDL khác. Với Graph Database, chúng ta không cần xác định trước cấu trúc hay mô hình của CSDL. Có thể so với CSDL quan hệ, khi xây dựng một CSDL cho một bài toán, chúng ta phải xác định các bảng, các trường, kiểu của trường, … và sau đó gần như là không thay đổi gì hoặc thay đổi cũng rất khó khăn, đơn cử như việc thêm bớt trường hay thay đổi kiểu của trường, … Với Graph Database thì khác, chúng ta hoàn toàn có thể thêm vào nút hay các mối quan hệ mới mà không làm ảnh hưởng tới các nút khác, cũng không làm thay đổi kết quả truy vấn đang thực hiện.. Như vậy, ta có thể thấy Graph Database rất linh động, phù hợp cho các công việc có tính chất hay thay đổi và độ ổn định không cao. Bởi tính linh động này mà người ta không cần phải cam kết một mô hình nào đó mà sau một thời gian lại thấy không phù hợp và phải xây dựng lại. Ngoài ra thì tính linh động này còn giảm bớt chi phí bảo trì và rủi ro như các CSDL khác mang lại.

### Nhanh

Với tính chất tự nhiên trong các lược đồ tự do của mô hình dữ liệu của Graph Database cộng với việc được hỗ trợ tốt các API (Application Programming Interface) cũng như là ngôn ngữ truy vấn, Graph Database cho phép chúng ta phát triển các ứng dụng một cách có kiểm soát. Tính nhanh ở đây là chúng ta có thể vận dụng và thử các xu hướng mới một cách nhanh nhẹn và đưa ra xu hướng phù hợp nhất có thể.

# Giới thiệu neo4j

## Neo4j là gì?

Neo4j hiện đang đứng số một trong mô hình cơ sở dữ liệu dạng đồ thị được viết bằng Java. Neo4j hỗ trợ ngôn ngữ truy vấn Cypher, một ngôn ngữ rất trực quan trong việc truy vấn dữ liệu dưới dạng Graph.

Quy ước:

* Đỉnh trong Neo4j
* Được ký hiệu là (A)
* Có thể có nhiều thuộc tính dưới dạng key:value
* Có thể có một hoặc nhiều nhãn (B: Person) : B có nhãn là Person
* Cạnh trong Neo4j
* Được ký hiệu là -[rel]-
* Có thể có nhiều thuộc tính dưới dạng key:value
* Có duy nhất một kiểu của cạnh: (a)-[rel:KNOW]-(b) mô tả a và b được liên kết với nhau bởi cạnh rel có kiểu là KNOW
* có thể có hướng: (a)-[rel:KNOW]->(b)

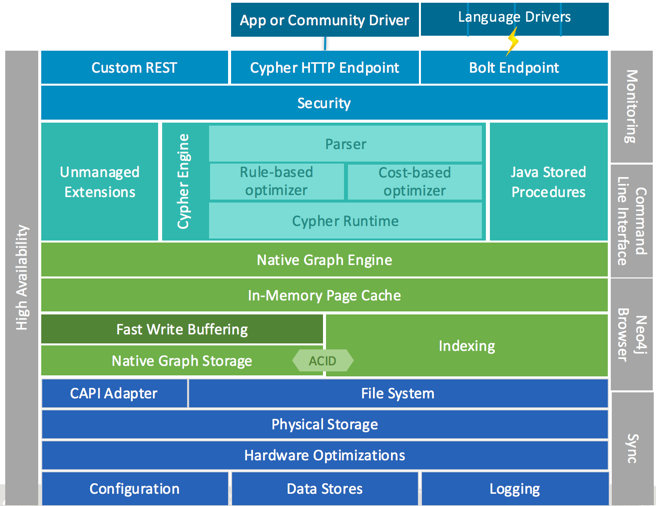
## Tính năng

* Sử dụng ngôn ngữ truy vấn CQL.
* Property Graph Data Model
* Hỗ trợ Indexes by sử dụng Apache Lucence
* Hỗ trợ UNIQUE constraints
* Có giao diện thực thi CQL Commands: Neo4j Data Browser
* Hỗ trợ đầy đủ ACID (Atomicity, Consistency, Isolation and Durability) rules. Neo4j sử dụng các giao dịch để đảm bảo dữ liệu được tồn tại trong trường hợp lỗi phần cứng hoặc hệ thống bị treo.
* Hỗ trợ truy xuất dữ liệu định dạng JSON và XLS.
* Xây dựng một cơ sở dữ liệu dạng đồ thị tự nhiên. Kiển trúc của nó được tối ưu hóa cho việc quản lý, lưu trữ.
* Neo4j có tính sẵn sàng cao thông qua sao lưu, sao chép trực tiếp.
* Neo4j có thể được sử dụng như một cơ sở dữ liệu nhúng và quản lý như là một máy chủ độc lập với khả năng tích hợp REST API để dễ dàng truy xuất dữ liệu.
* Có khả năng triển khai trên mô hình cụm máy.

## Ưu điểm nổi bật

* Nó là rất dễ dàng để diễn tả dữ liệu được kết nối.
* Nó là rất dễ dàng và nhanh hơn để lấy / nối / chuyển hướng của dữ liệu được kết nối nhiều.
* Nó trình bày dữ liệu bán cấu trúc rất dễ dàng.
* Neo4j CQL lệnh ngôn ngữ truy vấn là ở định dạng dễ tiếp cận và rất dễ dàng để tìm hiểu
* Nó sử dụng mô hình dữ liệu đơn giản và mạnh mẽ.
* Nó không đòi hỏi kết nối phức tạp để lấy quan hệ / dữ liệu liên quan vì nó là rất dễ dàng để lấy qua node tiếp giáp hoặc mối quan hệ chi tiết mà không Joins hoặc index.

## Kiến trúc



### Lưu trữ dữ liệu

Việc lưu trữ dữ liệu được thiết lập các tham số trong tệp *conf/neo4j.conf*, các tham số bao gồm: *dbms.directories.data, dbms.active\_database*. Mặc định, dữ liệu được lưu trữ tại: *data/databases/graph.db.*

| **Tệp tin** | **Mô tả** |
| --- | --- |
| **neostore.labeltokenstore\*** | Lưu trữ index related label data |
| **neostore.nodestore.\*** | Lưu trữ node related data |
| **neostore.relationship\*** | Lưu trữ data related to the relationships đượ tạo và định nghĩa |
| **neostore.propertystore.\*** | Lưu trữ thuộc tính dạng key – value. |

| **Store File** | **Record size - standard** | **Record size - high limit** | **Contents** |
| --- | --- | --- | --- |
| neostore.nodestore.db | 15B | 16B | Nodes |
| neostore.relationshipstore.db | 34B | 32B | Relationships |
| neostore.propertystore.db | 41 B | 48 B | Properties for nodes and relationships (up to 4 per record) |
| neostore.propertystore.db.strings | 128 B | 128 B | Long String properties (configurable record size) |
| neostore.propertystore.db.arrays | 128 B | 128 B | Large array properties (configurable record size) |

## Cài đặt

### Yêu cầu hệ thống

* CPU: tối thiểu Intel Core i3 (khuyến nghị Intel Core i7, IBM POWER8)
* RAM: tối thiểu 2GB (Khuyến nghị >=16GB)
* Disk 10GB SATA (khuyến nghị SSD)
* Filesystem: ext4
* Máy ảo java Java: Oracle or OpenJDK
* Neo4j 2.3.0 và cao hơn: JDK 8
* Neo4j 2.2.x và thấp hơn: JDK 7

### Thiết lập cấu hình

### Mặc định khi cài Neo4j:

| **Package** | **Configuration** | **Data** | **Logs** | **Metrics** | **Import** | **Bin** | **Lib** | **Plugins** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Linux or OS X tarball | *<neo4j-home>/conf/neo4j.conf* | *<neo4j-home>/data* | *<neo4j-home>/logs* | *<neo4j-home>/metrics* | *<neo4j-home>/import* | *<neo4j-home>/bin* | *<neo4j-home>/lib* | *<neo4j-home>/plugins* |
| Windows zip | *<neo4j-home>\conf\neo4j.conf* | *<neo4j-home>\data* | *<neo4j-home>\logs* | *<neo4j-home>\metrics* | *<neo4j-home>\import* | *<neo4j-home>\bin* | *<neo4j-home>\lib* | *<neo4j-home>\plugins* |
| Debian/Ubuntu .deb | */etc/neo4j/neo4j.conf* | */var/lib/neo4j/data* | */var/log/neo4j* | */var/lib/neo4j/metrics* | */var/lib/neo4j/import* | */usr/share/neo4j/bin* | */usr/share/neo4j/lib* | */var/lib/neo4j/plugins* |
| Windows desktop | *%APPDATA%\Neo4j Community Edition\neo4j.conf* | *%APPDATA%\Neo4j Community Edition* | *%APPDATA%\Neo4j Community Edition\logs* | *%APPDATA%\Neo4j Community Edition\metrics* | *%APPDATA%\Neo4j Community Edition\import* | *%ProgramFiles%\Neo4j CE 3.0\bin* | (in package) | *%ProgramFiles%\Neo4j CE 3.0\plugins* |
| OS X desktop | *${HOME}/Documents/Neo4j/neo4j.conf* | *${HOME}/Documents/Neo4j* | *${HOME}/Documents/Neo4j/logs* | *${HOME}/Documents/Neo4j/metrics* | *${HOME}/Documents/Neo4j/import* | (in package) | (in package) | (in package) |

Các tệp ghi chép

| **Tên tệp** | **Mô tả** |
| --- | --- |
| *neo4j.log* | Nhật ký chuẩn, các thông tin chung về Neo4j được viết. |
| *debug.log* | Các thông tin hữu ích khi gỡ các vấn đề về lỗi với Neo4j |
| *http.log* | Nhật ký các yêu cầu đến HTTP API |
| *gc.log* | Nhật ký của Garbage Collection cung cấp bởi JVM |
| *query.log* | Nhật ký truy vấn (bản tính phí) |
| *service-error.log* | Nhật ký lỗi gặp phải khi cài đặt hoặc chạy các dịch vụ Windows. (Chỉ dành cho Windows). |

Thư mục *<neo4j-home>*, *bin* và *conf*  có thể cần phải cấu hình tạo biến môi trường.

| **Vị trí** | **Mặc định** | **Biến môi trường** | **Ghi chú** |
| --- | --- | --- | --- |
| *<neo4j-home>* | Thư mục cha của *bin* | NEO4J\_HOME | Phải được thiết lập rõ rang nếu bin không phải là một thư mục con. |
| *bin* | Thư mục chứa các *neo4j* script | NEO4J\_BIN | Phải được thiết lập một cách rõ ràng nếu neo4j script được gọi như là một liên kết mềm. |
| *conf* | *<neo4j-home>/conf* | NEO4J\_CONF | Phải được thiết lập một cách rõ ràng nếu nó không phải là một thư mục con của *<neo4j-home>*. |

### Quyền

Chỉ đọc

* conf
* import
* bin
* lib
* plugins

Đọc và viết

* data
* logs
* metrics

Thực thi

* tất cả tài liệu trong *bin*

### Cài đặt

Trên Linux

1. Tải file tar.gz tại: <http://neo4j.com/download/>
2. Giải nén: tar -xf <filename> ra thư mục NEO4J\_HOME
3. Di chuyển đến thư mục NEO4J\_HOME và thực thi *./bin/neo4j console*.

Console bao gồm: start, stop, restart, status.

1. Dừng server gõ Ctrl-C.

Trên OS X

1. Tải file .dmg tại: <http://neo4j.com/download/>
2. Bấm vào file đã tải
3. Kéo Neo4j icon vào thư mục Applications.

Trên Windows

1. Tải file .exe hoặc .msi
2. Bấm chạy tệp cài đặt.

### Thiết đặt các kết nối tới Neo4j

Có 3 cách kết nối: bolt, http, https

# Bolt connector

dbms.connector.bolt.type=BOLT

dbms.connector.bolt.enabled=true

dbms.connector.bolt.tls\_level=OPTIONAL

# To have Bolt accept non-local connections, uncomment this line

# dbms.connector.bolt.address=0.0.0.0:7687

# HTTP Connector

dbms.connector.http.type=HTTP

dbms.connector.http.enabled=true

#dbms.connector.http.encryption=NONE

# To have HTTP accept non-local connections, uncomment this line

#dbms.connector.http.address=0.0.0.0:7474

# HTTPS Connector

dbms.connector.https.type=HTTP

dbms.connector.https.enabled=true

dbms.connector.https.encryption=TLS

dbms.connector.https.address=localhost:7473

Trình kết nối bolt là cổng mà chấp nhận kết nối qua giao thức cơ sở dữ liệu Bolt. Đây là giao thức được sử dụng bởi các trình điều khiển chính thức Neo4j. Phải có ít nhất một trình điều khiển Bolt.

Neo4j cũng có thể được cấu hình cho nhiều kết nối bolt cho phép cho các kết nối từ xa và địa phương riêng biệt mà có thể có yêu cầu mã hóa khác nhau. Mỗi connector có một tên duy nhất để xác định nó, ký hiệu <bolt-connector-name> bên dưới. Ví dụ, một kết nối dành cho sử dụng bên ngoài được đặt tên là "bolt-public". Tên của trình cắm bolt trong cấu hình mặc định là bolt.

| **Tên** | **Mô tả** | **Giá trị** | **Giá trị mặc định** |
| --- | --- | --- | --- |
| dbms.connector.<bolt-connector-name>.address | Địa chỉ kết nối được gắn vào | <host>:<port> | localhost:7687 |
| dbms.connector.<bolt-connector-name>.enabled | Kích hoạt trình kết nối | true hoặc false | false |
| dbms.connector.<bolt-connector-name>.tls\_level | Cấp độ mã hóa yêu cầu | REQUIRED, OPTIONAL, hoặc DISABLED | OPTIONAL |
| dbms.connector.<bolt-connector-name>.type | Loại kết nối | BOLT hoặc HTTP | Giá trị Bolt là bắt buộc cho trình kết nối bolt. |

Trình kết nối HTTP trình bày điểm cuối HTTP của Neo4j. Nó được yêu cầu kết nối mã hóa. Chắc chắn phải có chính xác một kết nối HTTP và có thể có kết nối HTTPS.

Mỗi connector có một tên duy nhất để xác định nó, ký hiệu <http-connector-name>. Ví dụ, một kết nối dành cho sử dụng bên ngoài được đặt tên là "http-public". Tên của trình điều khiển HTTP trong cấu hình mặc định là http, và tên của trình điều khiển HTTPS trong cấu hình mặc định là https.

| **Tên** | **Mô tả** | **Giá trị** | **Giá trị mặc định** |
| --- | --- | --- | --- |
| dbms.connector.<http-connector-name>.address | Địa chỉ kết nối được gắn vào | <host>:<port> | localhost:7474 |
| dbms.connector.<http-connector-name>.enabled | Kích hoạt trình kết nối | true hoặc false | false |
| dbms.connector.<http-connector-name>.encryption | Kích hoạt TLS cho trình kết nối | NONE hoặc TLS | NONE |
| dbms.connector.<http-connector-name>.type | Loại kết nối | BOLT hoặc HTTP | Giá trị HTTP là bắt buộc cho trình kết nối HTTP. |

### Thời hạn transaction

## Bảo mật

### Cấu hình cơ bản

Mặc định, Neo4j được cài đặt chỉ để phục vụ cho các dịch vụ web, các ứng dụng cùng máy chủ có thể truy cập được. Thiết đặt này được cấu hình tronh tệp *neo4j.conf*:

# Let the webserver only listen on the specified IP. Default is localhost (only

# accept local connections). Uncomment to allow any connection.

dbms.connector.http.type=HTTP

dbms.connector.http.enabled=true

#dbms.connector.http.address=0.0.0.0:7474

Nếu muốn thiết lập cho phép các kết nối từ bên ngoài, bạn chỉ cần bỏ dấu # ở dòng

#dbms.connector.http.address=0.0.0.0:7474

Lưu ý rằng tường lửa phải được cấu hình phù hợp.

### Xác thực người dùng

Về vấn đề xác thực người dùng, Neo4j yêu cầu phía client cung cấp thông tin xác thực khi truy cập vào API REST. Thông tin xác thực và ủy quyền được lưu trữ trong tệp: *data/dbms/auth.* Bạn có thể sao chép tập tin xác thực này sang instance khác để sử dụng cùng một người dùng.

Khi truy cập Neo4j qua mạng không an toàn, hãy đảm bảo HTTPS được cấu hình và sử dụng để truy cập.

Nếu cần thiết, chứng thực có thể vô hiệu hóa. Điều này sẽ cho phép bất kỳ máy khách có thể truy cập cơ sở dữ liệu mà không cần cung cấp thông tin xác thực.

# Disable authorization

dbms.security.auth\_enabled=false

### Thiết đặt HTTPS

Ngay từ lần chạy đầu tiên, Neo4j tự động tạo một chứng chỉ SSL và một private key. Do đó chứng nhận này tự ký và không an toàn, bạn nên cung cấp private key của mình và chứng nhận cho máy chủ để sử dụng.

Bạn cung cấp key và certificate của mình bằng cách tạo *neo4j.key* và *neo4j.cert* trong thư mục *certificates*. Vị trí thư mục đươc thiết đặt trong biến "*dbms.directories.certificates*" của tệp *neo4j.conf*.

# Certificates location (auto generated if the file does not exist)

dbms.directories.certificates=certificates

### Thiết đặt các quy tắc ủy quyền truy cập Neo4j

Quản trị Neo4j có thể thiết đặt quyền hạn chi tiết hơn ngoài việc ủy quyền và giới hạn địa chỉ IP. Nó có thể quản lý việc cho phép truy cập các phương diện cụ thể của dữ liệu.

* Enforcing server authorization rules

In this example, a (dummy) failing security rule is registered to deny access to all URIs to the server by listing the rules class in neo4j.conf:

org.neo4j.server.rest.security\_rules=my.rules.PermanentlyFailingSecurityRule

with the rule source code of:

public class PermanentlyFailingSecurityRule implements SecurityRule

{

public static final String REALM = "WallyWorld"; // as per RFC2617 :-)

@Override

public boolean isAuthorized( HttpServletRequest request )

{

return false; // always fails - a production implementation performs

// deployment-specific authorization logic here

}

@Override

public String forUriPath()

{

return "/\*";

}

@Override

public String wwwAuthenticateHeader()

{

return SecurityFilter.basicAuthenticationResponse(REALM);

}

}

With this rule registered, any access to the server will be denied. In a production-quality implementation the rule will likely lookup credentials/claims in a 3rd-party directory service (e.g. LDAP) or in a local database of authorized users.

Example request

* **POST** http://localhost:7474/db/data/node
* **Accept:** application/json; charset=UTF-8

Example response

* **401:** Unauthorized
* **WWW-Authenticate:** Basic realm="WallyWorld"

**…**……………………………

……………………………….

### Sử dụng Proxy

Mặc dù Neo4j đã tích hợp sẵn những tính năng bảo mật nhưng họ vẫn có thể bổ sung bảo mật bằng cách sử dụng một proxy để kết nối phía bên ngoài.

* Kiểm soát truy cập tới Neo4j

<Proxy \*>

Order Deny,Allow

Deny from all

Allow from 192.168.0

</Proxy>

* Chạy Neo4j với vai trò không phải người dùng root trên port<1000

ProxyPass /neo4jdb/data http://localhost:7474/db/data

ProxyPassReverse /neo4jdb/data http://localhost:7474/db/data

* Cân bằng tải trong môi trường làm việc cụm máy chủ:

<Proxy balancer://mycluster>

BalancerMember http://192.168.1.50:80

BalancerMember http://192.168.1.51:80

</Proxy>

ProxyPass /test balancer://mycluster

## Import

Trên môi trường câu lệnh:

*Ta thực thi ./*bin*/*neo4j-import

Các lựa chọn bao gồm:

**--into <store-dir>**

**--nodes[:Label1:Label2] "<file1>,<file2>,…​"**

**--relationships[:RELATIONSHIP\_TYPE] "<file1>,<file2>,…​"**

**--delimiter <delimiter-character>**

**--array-delimiter <array-delimiter-character>**

**--quote <quotation-character>**

**--multiline-fields <true/false>**

**--input-encoding <character set>**

**--ignore-empty-strings <true/false>**

**--id-type <id-type>**

**--processors <max processor count>**

**--stacktrace <true/false>**

**--bad-tolerance <max number of bad entries>**

**--skip-bad-relationships <true/false>**

**--skip-duplicate-nodes <true/false>**

**--ignore-extra-columns <true/false>**

**--db-config <path/to/neo4j.conf>**

## Sao lưu, dự phòng

### Thiết đặt

* *dbms.backup.enabled=true* Cho phép backup.
* *dbms.backup.address=<hostname or IP address>:6362* cấu hình địa chỉ lắng nghe. Mặc định là 127.0.0.1:6362. Bạn Cũng có thể thiết đặt cho bên ngoài *dbms.backup.address=0.0.0.0:6362*.

### Thực thi sao lưu

Sử dung công cụ neo4j-backup

# Performing a full backup: create a blank directory and run the backup tool

mkdir /mnt/backup/neo4j-backup

./bin/neo4j-backup -host 192.168.1.34 -to /mnt/backup/neo4j-backup

# Performing an incremental backup: just specify the location of your previous backup

./bin/neo4j-backup -host 192.168 -to /mnt/backup/neo4j-backup

# Performing an incremental backup where the service is listening on a non-default port

./bin/neo4j-backup -host 192.168.1.34 -port 9999 -to /mnt/backup/neo4j-backup

Các tệp backup được lưu trữ tại /mnt/backup/neo4j-backup .

Không thể sao lưu thành công nếu:

* Nếu thư mục hiện tại không chứa một bản sao lưu hợp lệ.
* Nếu thư mục hiện có chứa một bản sao lưu của cơ sở dữ liệu khác.
* Nếu thư mục hiện có chứa một bản sao lưu từ một phiên bản cơ sở dữ liệu trước đó.

### Khôi phục dữ liệu

1. Dừng Neo4j.
2. Thay thế file cơ sở dữ liệu trong thư mục data bằng file trong thư mục backup.
3. Khởi động Neo4j.

## Giám sát

## Hiệu suất

# Giới thiệu cypher

## Khái niệm

CYPHER là một ngôn ngữ truy vấn dữ liệu dành riêng cho Graph BD, cho phép truy vấn hiệu quả nhất. Đây là một ngôn ngữ đơn giản nhưng rất mạnh mẽ, truy vấn dữ liệu phức tạp có thể dễ dàng thể hiện qua ngôn ngữ Cypher. Nó cho phép tập trung vào miền của họ hay vì phải lần mò trong cơ sở dữ liệu.

Cypher được thiết kế để phù hợp với người lập trình. Cấu trúc của nó được dựa trên văn xuôi tiếng Anh và hình tượng gọn gàng giúp để thực hiện truy vấn sáng sủa hơn.

Kiểu dữ liệu:

|  |  |  |
| --- | --- | --- |
| **STT** | **Kiểu** | **Mô tả** |
| 1. | boolean | true, false. |
| 2. | byte | 8-bit integers. |
| 3. | short | 16-bit integers. |
| 4. | int | 32-bit integers. |
| 5. | long | 64-bit integers. |
| 6. | float | 32-bit floating-point numbers. |
| 7. | double | 64-bit floating-point numbers. |
| 8. | char | 16-bit characters. |
| 9. | String | Strings. |

## Cú pháp

Cypher vay mượn cấu trúc của SQL, truy vấn được xây dựng bằng cách sử dụng mệnh đề khác nhau:

|  |  |  |
| --- | --- | --- |
| **STT** | **Từ khóa** | **Chức năng** |
| 1. | CREATE | Tạo nốt, quan hệ, thuộc tính |
| 2. | MATCH | Lấy dữ liệu |
| 3. | RETURN | Xuất kết quả truy vấn |
| 4. | WHERE | Nêu điều kiện lọc dữ liệu truy vấn |
| 5. | DELETE | Xóa nốt và quan hệ |
| 6. | REMOVE | Xóa thuộc tính của nốt, thuộc tính của quan hệ |
| 7. | ORDER BY | Sắp xếp dữ liệu trả về |
| 8. | SET | Thêm, cập nhật nhãn |

### Create

* Tạo node không thuộc tính:

CREATE (<node-name>:<label-name>)

Ví dụ: CREATE (emp:Employee)

* Tạo node có thuộc tính

CREATE (

<node-name>:<label-name>

{

<Property1-name>:<Property1-Value>

........

<Propertyn-name>:<Propertyn-Value>

}

)

Ví dụ: CREATE (dept:Dept { deptno:10,dname:"Accounting",location:"Hyderabad" })

* Tạo quan hệ không thuộc tính giữa các node có sẵn

MATCH (<node1-label-name>:<node1-name>),(<node2-label-name>:<node2-name>)

CREATE

(<node1-label-name>)-[<relationship-label-name>:<relationship-name>]->(<node2-label-name>)

RETURN <relationship-label-name>

Ví dụ:

MATCH (e:Customer)

RETURN e

MATCH (cc:CreditCard)

RETURN cc

Sau đó:

MATCH (e:Customer),(cc:CreditCard)

CREATE (e)-[r:DO\_SHOPPING\_WITH ]->(cc)

* Tạo quan hệ có thuộc tính giữa các node có sẵn

MATCH (<node1-label-name>:<node1-name>),(<node2-label-name>:<node2-name>)

CREATE

(<node1-label-name>)-[<relationship-label-name>:<relationship-name>

{<define-properties-list>}]->(<node2-label-name>)

RETURN <relationship-label-name>

Ví dụ:

MATCH (e:Customer)

RETURN e

MATCH (cc:CreditCard)

RETURN cc

Sau đó:

MATCH (cust:Customer),(cc:CreditCard)

CREATE (cust)-[r:DO\_SHOPPING\_WITH{shopdate:"12/12/2014",price:55000}]->(cc)

RETURN r

* Tạo một nhãn cho một node

CREATE (<node-name>:<label-name>)

Ví dụ: CREATE (google1:GooglePlusProfile)

* Tạo nhiều nhãn cho 1 node

CREATE (<node-name>:<label-name1>:<label-name2>.....:<label-namen>)

Ví dụ: CREATE (m:Movie:Cinema:Film:Picture)

* Tạo một nhãn cho 1 quan hệ

CREATE (<node1-name>:<label1-name>)-

[(<relationship-name>:<relationship-label-name>)]

->(<node2-name>:<label2-name>)

Ví dụ: CREATE (p1:Profile1)-[r1:LIKES]->(p2:Profile2)

### Match return

MATCH Command

RETURN Command

Dùng để trích xuất dữ liệu:

* Node:

MATCH (e:Customer)

RETURN e

* Thuộc tính của node

MATCH

(

<node-name>:<label-name>

)

RETURN

<node-name>.<property1-name>,

...

<node-name>.<propertyn-name>

Ví dụ:

MATCH (dept: Dept)

RETURN dept.deptno,dept.dname

* Quan hệ các node

MATCH

(<node1-label-name>)-[<relationship-label-name>:<relationship-name>]->(<node2-label-name>)

RETURN <relationship-label-name>

Thêm cú pháp WHERE để tạo bộ lọc cho việc trích xuất dữ liệu

WHERE <condition>

Hoặc

WHERE <condition> <boolean-operator> <condition>

Các Boolean-operator bao gồm: AND, OR, NOT, XOR.

Condition là các phép toán so sánh: =, <>, <, >, <=, >=.

Ví dụ:

* Lấy danh sách node có điều kiện:

MATCH (emp:Employee)

WHERE emp.name = 'Abc'

RETURN emp

Hoặc

MATCH (emp:Employee)

WHERE emp.name = 'Abc' OR emp.name = 'Xyz'

RETURN emp

* Tạo quan hệ giữa các node với điều kiện:

MATCH (<node1-label-name>:<node1-name>),(<node2-label-name>:<node2-name>)

WHERE <condition>

CREATE (<node1-label-name>)-[<relationship-label-name>:<relationship-name>

{<relationship-properties>}]->(<node2-label-name>)

Ví dụ:

MATCH (cust:Customer),(cc:CreditCard)

WHERE cust.id = "1001" AND cc.id= "5001"

CREATE (cust)-[r:DO\_SHOPPING\_WITH{shopdate:"12/12/2014",price:55000}]->(cc)

RETURN r

### Set

* Tạo thuộc tính mới hoặc cập nhật thuộc tính cho node hoặc quan hệ đã tồn tại.

SET <property-name-list>

Ví dụ:

MATCH (dc:DebitCard)

SET dc.atm\_pin = 3456

RETURN dc

### Delete

* Để xóa node

MATCH (<node1-label-name>:<node1-name>) DELETE e

Ví dụ:

MATCH (e: Employee) DELETE e

* Xóa node và quan hệ

DELETE <node1-name>,<node2-name>,<relationship-name>

Ví dụ:

MATCH (cc: CreditCard)-[rel]-(c:Customer)

DELETE cc,c,rel

### Remove

* Loại bỏ nhãn của một node hoặc một quan hệ

REMOVE <label-name-list>

Ví dụ:

MATCH (m:Movie)

REMOVE m:Picture

* Loại bỏ thuộc tính của một node hoặc một quan hệ

REMOVE <property-name-list>

Ví dụ:

MATCH (book { id:122 })

REMOVE book.price

RETURN book

Hoặc

MATCH (dc:DebitCard)

REMOVE dc.cvv

RETURN dc

### Cú pháp khác

* Sắp xếp khi trích xuất dữ liệu: ORDER BY

MATCH (emp:Employee)

RETURN emp.empid,emp.name,emp.salary,emp.deptno

ORDER BY emp.name DESC

* Hợp UNION nhưng khi return phải dùng kết hợp “AS” cho các trường dữ liệu cùng tên. UNION ALL tương tự UNION nhưng nó có thể xuất dữ liệu trùn lặp.

<MATCH Command1>

UNION

<MATCH Command2>

Ví dụ:

MATCH (cc:CreditCard)

RETURN cc.id as id,cc.number as number,cc.name as name,

cc.valid\_from as valid\_from,cc.valid\_to as valid\_to

UNION

MATCH (dc:DebitCard)

RETURN dc.id as id,dc.number as number,dc.name as name,

dc.valid\_from as valid\_from,dc.valid\_to as valid\_to

* Tạo giới hạn, bộ lọc cho trích xuất dữ liệu:

LIMIT <number>

Ví dụ:

MATCH (emp:Employee)

RETURN emp

LIMIT 2

* Câu lệnh MERGE có nhiệm vụ tạo node, quan hệ, thuộc tính và trích xuất dữ liệu

MERGE = CREATE + MATCH

MERGE (<node-name>:<label-name>

{

<Property1-name>:<Pro<rty1-Value>

.....

<Propertyn-name>:<Propertyn-Value>

})

Ví dụ:

MERGE (gp2:GoogleProfile2{ Id: 201402,Name:"Nokia"})

MATCH (gp1:GoogleProfile1)

RETURN gp1.Id,gp1.Name

* Thêm điều kiện trong tập giá trị: IN

IN[<Collection-of-values>]

Ví dụ:

MATCH (e:Employee)

WHERE e.id IN [123,124]

RETURN e.id,e.name,e.sal,e.deptno

# Quản lý dữ liệu

## Các bước tạo dữ liệu

Gồm 3 bước:

1. Tạo các nốt
2. Tạo index cho các đối tượng
3. Tạo các quan hệ

## Import dữ liệu

Khi import dữ liệu cũng phải tuân thủ theo 3 bước trên

Cú pháp như sau: **movies2.csv.**

movieId:ID;title;year:int;:LABEL

tt0133093;'The Matrix';1999;Movie

tt0234215;'The Matrix Reloaded';2003;Movie|Sequel

tt0242653;'The Matrix Revolutions';2003;Movie|Sequel

**actors2.csv.**

personId:ID;name;:LABEL

keanu;'Keanu Reeves';Actor

laurence;'Laurence Fishburne';Actor

carrieanne;'Carrie-Anne Moss';Actor

**roles2.csv.**

:START\_ID;role;:END\_ID;:TYPE

keanu;'Neo';tt0133093;ACTED\_IN

keanu;'Neo';tt0234215;ACTED\_IN

keanu;'Neo';tt0242653;ACTED\_IN

laurence;'Morpheus';tt0133093;ACTED\_IN

laurence;'Morpheus';tt0234215;ACTED\_IN

laurence;'Morpheus';tt0242653;ACTED\_IN

carrieanne;'Trinity';tt0133093;ACTED\_IN

carrieanne;'Trinity';tt0234215;ACTED\_IN

carrieanne;'Trinity';tt0242653;ACTED\_IN

The call to neo4j-import would look like this:

./bin/neo4j-import --into path\_to\_target\_directory --nodes movies2.csv --nodes actors2.csv --relationships roles2.csv --delimiter ";" --array-delimiter "|" --quote "'"