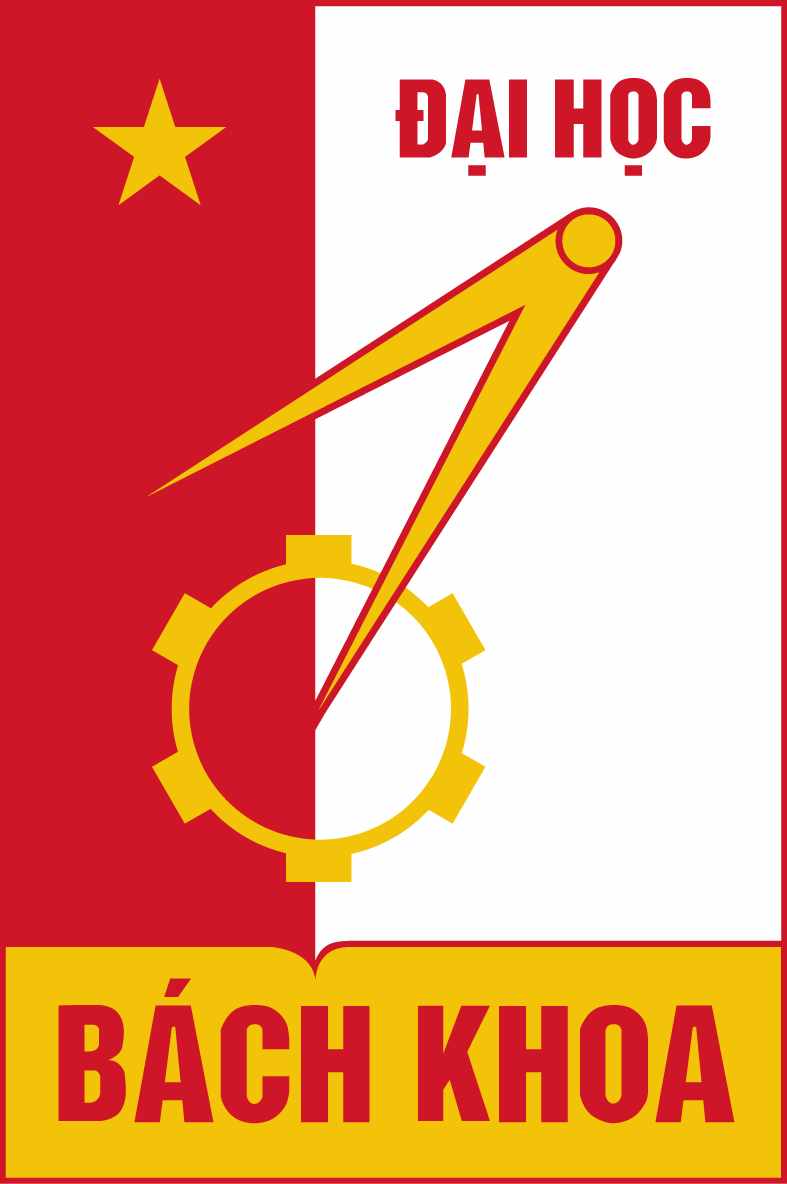
TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

****

**Đề tài:**

**Tìm hiểu về công nghệ lưu trữ GraphDB**

Giảng viên hướng dẫn: TS. Trịnh Tuấn Đạt

Bộ môn công nghệ phần mềm

Lớp 104404 - nhóm 5:

Trần Trung Huỳnh 20161917 40%

Dương Thị Thoa 20163902 15%

Đinh Mạnh Hùng 20161925 15%

Lâm Phúc Anh Tú 20167974 15%

Lê Xuân Long 20162521 15%

MỤC LỤC

[I. Nhiệm vụ của từng thành viên 4](#_Toc533380264)

[II. Tìm hiểu về công nghệ 5](#_Toc533380265)

[1. Giới thiệu về công nghệ 5](#_Toc533380266)

[2. Các thuật ngữ 5](#_Toc533380267)

[2.1. Resource Description Framework – RDF (Bộ khung mô tả tài nguyên) 5](#_Toc533380268)

[2.2. Bộ ba Triples: Subject-Predicate-Object 5](#_Toc533380269)

[2.3. Uniform Resource Identifiers – URIs (Mã định danh tài nguyên) 6](#_Toc533380270)

[2.4. Literal 6](#_Toc533380271)

[2.5. Resource description 6](#_Toc533380272)

[3. Hướng dẫn cài đặt 7](#_Toc533380273)

[3.1. Trên Windows 7](#_Toc533380274)

[3.2. Trên MacOS 7](#_Toc533380275)

[3.3. Trên Linux 7](#_Toc533380276)

[3.4. Chạy như một server riêng lẻ. 8](#_Toc533380277)

[4. Các kỹ thuật sử dụng 8](#_Toc533380278)

[4.1. Tạo một repository mới 8](#_Toc533380279)

[4.2. Thêm dữ liệu vào repository và chuẩn bị API 9](#_Toc533380280)

[5. Cách truy vấn 11](#_Toc533380281)

[5.1. Truy vấn trực tiếp trên server 11](#_Toc533380282)

[5.2. Truy vấn thông qua API 12](#_Toc533380283)

[III. Mô hình dữ liệu 14](#_Toc533380284)

[1. Mô hình dữ liệu 14](#_Toc533380285)

[2. Giải thuật sinh và upload dữ liệu 15](#_Toc533380286)

[IV. Chương trình 16](#_Toc533380287)

[1. Biểu đồ thể hiện sự phụ giữa các package 16](#_Toc533380288)

[1.1. Package model 16](#_Toc533380289)

[1.2. Package main 17](#_Toc533380290)

[1.3. Package graphdb 17](#_Toc533380291)

[1.4. Package file.io 18](#_Toc533380292)

[1.5. Package create.entity 18](#_Toc533380293)

[2. Biểu đồ lớp 19](#_Toc533380294)

[2.1. Biểu đồ lớp các thực thể 19](#_Toc533380295)

[2.2. Biểu đồ các lớp sinh thực thể 20](#_Toc533380296)

[2.3. Biểu đồ upload thực thể 21](#_Toc533380297)

[2.4. Biểu đồ chạy hàm main 22](#_Toc533380298)

[V. Kiểm thử hiệu năng 23](#_Toc533380299)

[1. Các câu truy vấn 23](#_Toc533380300)

[1.1. Truy vấn cơ bản 23](#_Toc533380301)

[1.2. Các câu truy vấn nâng cao 25](#_Toc533380302)

[2. Đo thời gian truy vấn 28](#_Toc533380303)

[3. Kết luận 28](#_Toc533380304)

[3.1. Ưu điểm 28](#_Toc533380305)

[3.2. Nhược điểm 28](#_Toc533380306)

# I. Nhiệm vụ của từng thành viên

|  |  |  |
| --- | --- | --- |
| Tên | Phần trăm | Nhiệm vụ |
| Trần Trung Huỳnh | 40% | Thiết kế phân chia gói, lớp, code chương trình, tìm hiểu về công nghệ lưu trữ, nghĩ truy vấn, kết nối server, làm báo cáo phần mô hình, kiểm thử hiệu năng, chương trình, mô hình dữ liệu, cách truy vấn. |
| Dương Thị Thoa | 20% | Tìm hiểu về cách kết nối, nghĩ truy vấn, tạo dữ liệu quan hệ, tìm hiểu công nghệ, làm báo cáo phần, cách cài đặt, giới thiệu. |
| Đinh Mạnh Hùng | 20% | Tìm hiểu cách kết nối, nghĩ truy vấn, tạo dữ liệu quan hệ, tìm hiểu công nghệ, làm báo cáo phần kỹ thuật sử dụng. |
| Lâm Phúc Anh Tú | 10% | Tìm hiểu công nghệ, nghĩ truy vấn, báo cáo các thuật ngữ. |
| Lê Xuân Long | 10% | Tìm hiểu công nghệ, nghĩ truy vấn, báo cáo các thuật ngữ. |

# II. Tìm hiểu về công nghệ

## Giới thiệu về công nghệ

GraphDB là một cơ sở dữ liệu đồ thị mạnh mẽ và hiệu quả cao với sự hỗ trợ RDF và SPARQL. Nó tối ưu hoá việc tải và sử dụng các bộ dữ liệu điện toán đám mây được liên kết với nhau cũng như những tài nguyên của riêng bạn.GraphDB sử dụng RDF4J như là một thư viện, nhờ đó tận dụng được các API của RDF4J cho việc lưu trữ và truy vấn, cũng như hỗ trợ nhiều loại ngôn ngữ truy vấn như SPARQL, SeRQL, … và các cú pháp của RDF như RDF/XML, N3, Turtle, …

## Các thuật ngữ

### 2.1. Resource Description Framework – RDF (Bộ khung mô tả tài nguyên)

RDF là một tiêu chuẩn trao đổi dữ liệu trên Web. RDF là một tập hợp các nguyên tắc dành cho ngôn ngữ đánh dấu, cung cấp mô hình dữ liệu và cú pháp đơn giản sao cho các hệ thống độc lập có thể trao đổi và sử dụng. Đồng thời, nó được thiết kế sao cho hệ thống máy tính có thể hiểu được và có thể đọc được thông tin, chứ không chỉ đơn giản là để trình bày dữ liệu cho người dùng. Cú pháp của RDF được xây dựng dựa trên mô hình dữ liệu và mô hình này ảnh hưởng đến cách thức mà những thuộc tính được mô tả và nó cũng làm cho cấu trúc của những mô tả đó trở nên rõ ràng.

### 2.2. Bộ ba Triples: Subject-Predicate-Object

RDF sử dụng một mô hình trừu tượng để phân rã thông tin/kiến thức thành những mảnh con, với 1 số luật cơ bản về ngữ nghĩa cho các mảnh này. Mục tiêu là cung cấp 1 phương thức chung mà đủ đơn giản và linh hoạt để diễn giải bất kỳ sự thật (fact) nào, nhưng có cấu trúc để các ứng dụng máy tính có thể hiểu và diễn giải cấu trúc đó.

Mô hình trừu tượng gồm các thành phần chính sau:

* Statement (phát biểu hay mệnh đề).
* Predicate.
* Các nguồn tài nguyên subject và object.

Với các thành phần trên, chúng ta có 1 cấu trúc đồ thị của 1 phát biểu RDF như sau:



Mỗi mảnh kiến thức được phân ra từ mô hình RDF gọi là một statement, một statement RDF (phát biểu RDF) phải có định dạng sau: “subject-predicate-object “, thứ tự này không bao giờ được thay đổi (cố định). với subject và object là tên của 2 thứ bất kỳ và predicate là tên của 1 quan hệ kết nối giữa 2 thứ đó. Theo sơ đồ trên, chúng ta có subject chứa trong hình oval bên trái, subject chứa trong hình oval bên phải, và predicate là tên nhãn của mũi tên chỉ từ subject sang object.

Các statement luôn có 3 thành phần subject-predicate-object, vì vậy người ta thường gọi statement là 1 bộ ba (triple). Mỗi triple hay statement thể hiện 1 sự thật đơn (single fact), một tập các statement hay triple thể hiện 1 vài mảnh thông tin/kiến thức và hình thành nên mô hình RDF.

### 2.3. Uniform Resource Identifiers – URIs (Mã định danh tài nguyên)

"URI" là Uniform Resource Identifier. Nó là nhãn được sử dụng để xác định duy nhất các loại thực thể khác nhau trong biểu đồ RDF. Một URI điển hình trông rất giống địa chỉ web: http://example.org/project/ class#number. Mặc dù giống nhau, URI có thể là địa chỉ web hoặc không tùy người dùng định nghĩa. Nó chỉ đơn giản là một nhãn. URI được dùng với bộ mã ASCII còn IRI (được coi là bản nâng cấp của URI được dùng với mã UNICODE. IRI có 2 phần cơ bản, prefix và local. Một IRI có dạng như sau:

http://abc.com/xyz

* Trong đó http://abc.com/ là prefix (namespace), đây là 1 đường dẫn bắt kì, nhưng nên bắt đầu bằng http://
* xyz là phần local.

### 2.4. Literal

Literal cũng là 1 dạng RDF, nó thường được dùng để lưu trữ các dữ liệu có dạng String là chủ yếu, Literal chỉ có 1 thành phần, chính là thành phần mang ý nghĩa của literal.

### 2.5. Resource description

"Resource description" được định nghĩa là một tập hợp các triples mà tất cả đều có cùng một URI trong trường chủ đề. Nói cách khác, triples đều mô tả các thuộc tính của cùng một thứ.

## Hướng dẫn cài đặt

### 3.1. Trên Windows

* Bước 1: Tải tệp tin theo đường dẫn dưới đây:

<http://go.pardot.com/e/45622/b1b6b38-GraphDB-Free-8-8-0-exe/6l8mpj/1405360163?h=sBC4-S5zLBzNNozFnC77ZfCIolQig7oEDEAQb9nHjjU>

* Bước 2: Chạy tệp tin và làm theo chỉ dẫn trên màn hình
* Bước 3: Khởi chạy ứng dụng GraphDB trên màn hình Desktop. Cơ sở dữ liệu và công cụ quản trị được mở tại <http://localhost:7200/>

### 3.2. Trên MacOS

* Bước 1: Tải tệp tin .dmg

<http://go.pardot.com/e/45622/b1b6b38-GraphDB-Free-8-8-0-dmg/6l8mpl/1405360163?h=sBC4-S5zLBzNNozFnC77ZfCIolQig7oEDEAQb9nHjjU>

* Bước 2: Chạy tệp tin để mở ổ đĩa ảo và sao chép nội dung trong ổ ảo vào thư mục ứng dụng trong ổ đĩa cứng
* Bước 3: Khởi chạy ứng dụng GraphDB bằng cách nhấp chuột vào biểu tượng. Cử sở dữ liệu và công cụ quản trị được mở tại <http://localhost:7200/>

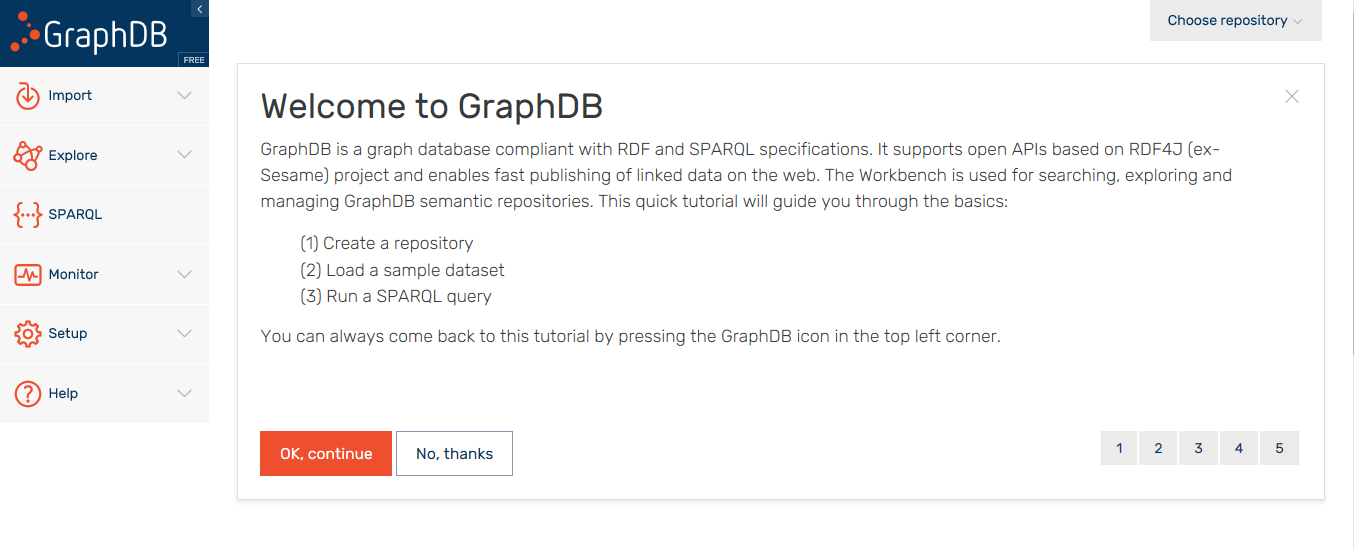
### 3.3. Trên Linux

* Bước 1: Tải tệp tin .rpm hoặc .deb

<http://go.pardot.com/e/45622/b1b6b38-graphdb-free-8-8-0-deb/6l8mpn/1405360163?h=sBC4-S5zLBzNNozFnC77ZfCIolQig7oEDEAQb9nHjjU>

<http://go.pardot.com/e/45622/raphdb-free-8-8-0-1-x86-64-rpm/6l8mpq/1405360163?h=sBC4-S5zLBzNNozFnC77ZfCIolQig7oEDEAQb9nHjjU>

* Bước 2: Cài đặt tệp tin với lệnh sudo rpm -i hoặc sudo dpkg -i cộng với tên của tệp đã tải
* Bước 3: Khởi chạy ứng dụng GraphDB bằng cách nhấp chuột vào biểu tượng. Cơ sở dữ liệu và công cụ quản trị được mở tại <http://localhost:7200/>

****

### 3.4. Chạy như một server riêng lẻ.

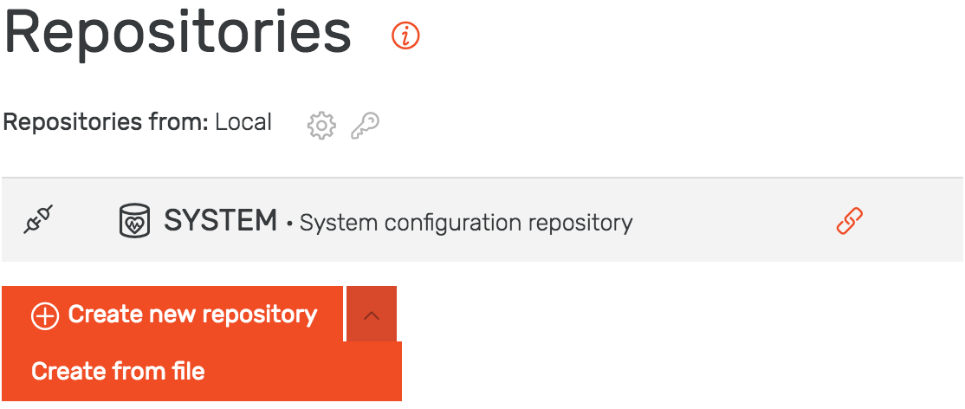
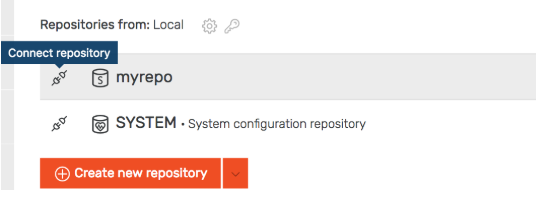
- Bước 1: Tải tài liệu theo link sau:

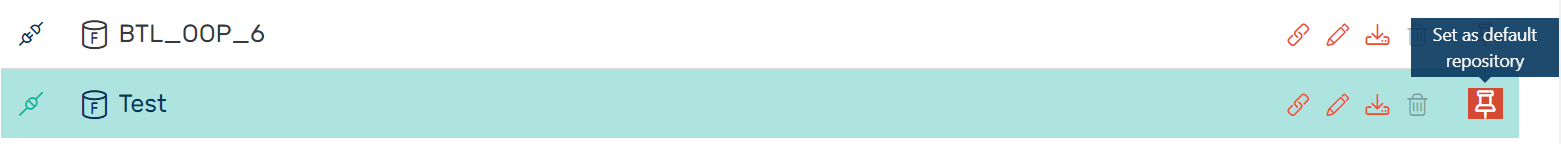
<http://go.pardot.com/e/45622/38-graphdb-free-8-8-0-dist-zip/6l8mps/1405360163?h=sBC4-S5zLBzNNozFnC77ZfCIolQig7oEDEAQb9nHjjU>

* Bước 2: Giải nén được thư mục graphdb-free-8.7.1 và vào trong graphdb-free-8.7.1\bin khởi chạy graphdb.cmd
* Bước 3: server đã bắt đầu chạy ở <http://localhost:7200/>

## Các kỹ thuật sử dụng

### 4.1. Tạo một repository mới

* Bước 1: Chọn *Setup -> Repositories*
* Bước 2: Chọn *Creat new repository*
* Bước 3: Điền *Repository ID* (ví dụ: myrepo), cách tuỳ chọn khác để giá trị mặc định
* Bước 4: Chọn nút *Connect* để đặt repository muốn sử dụng
* Bước 5: Chọn nút *Set as default repository* để đặt repository mặc định

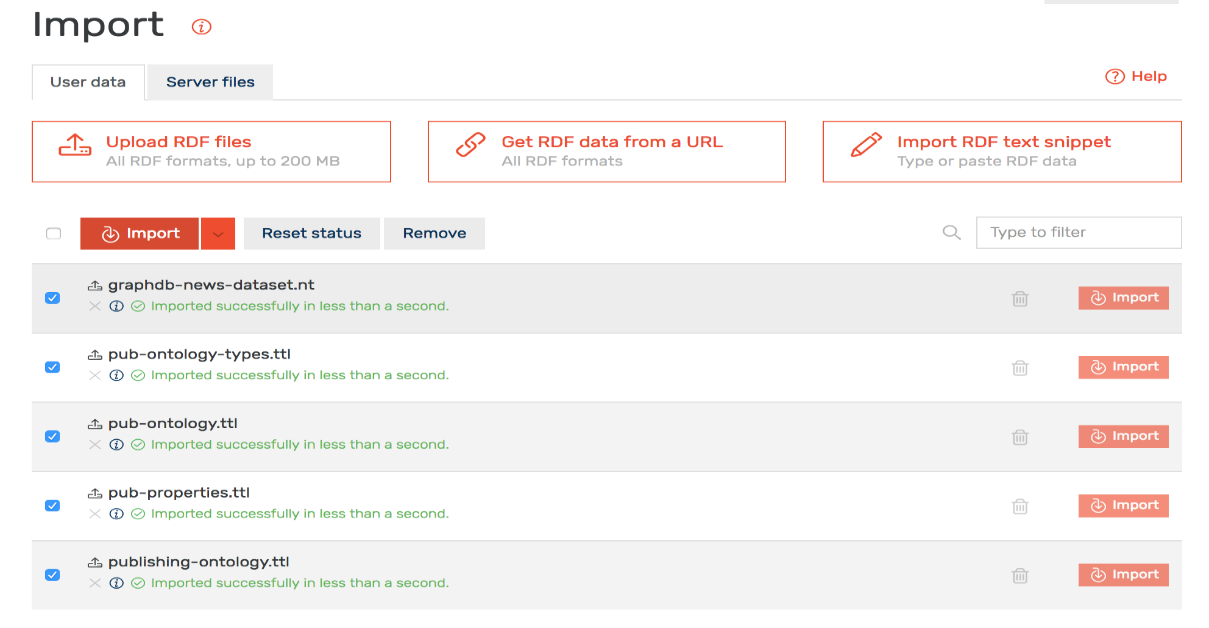


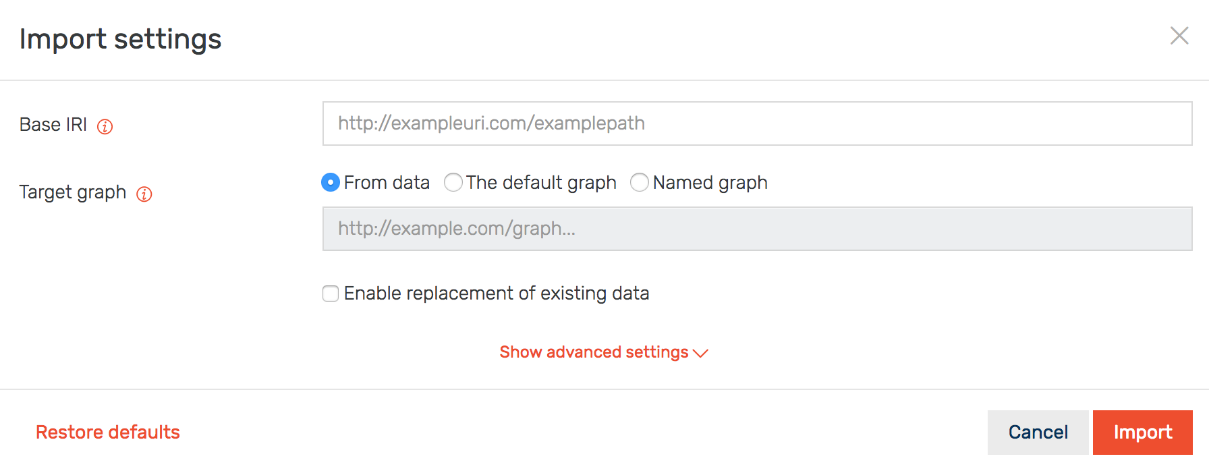
### 4.2. Thêm dữ liệu vào repository và chuẩn bị API

GraphDB hỗ trợ tải dữ liệu được lưu trữ sẵn trong máy thông qua công cụ quản trị GraphDB Workbench hoặc dùng API của RDF4J để thêm dữ liệu.

**4.2.1. Thông qua công cụ quản trị GraphDB Workbench**

* Bước 1: Chọn *Import -> RDF*

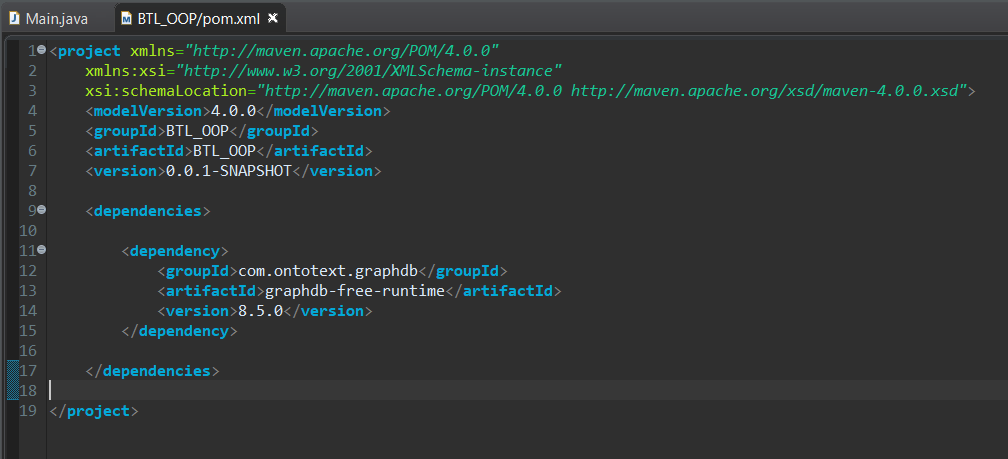
- Bước 2: Chọn tab *User Data* và chọn *Upload RDF files* để tải file lên repository

**-** Bước 3: Chọn nút *Import*, xuất hiện của sổ *Import settings*

**-** Bước 4: Nhập các thông số và ấn nút *Import* để tiến hành tải lên

**4.2.2. Thông qua API của RDF4J**

* Do GraphDB xây dựng dựa trên RDF4J nên ta có thể sử dụng các API của RDF4J để thêm dữ liệu từ máy khách bằng cách kết nối đến repository thông qua giao thức http.
* Chuẩn bị API:
* Đầu tiên cấu hình file pom.xml trong project maven như sau:



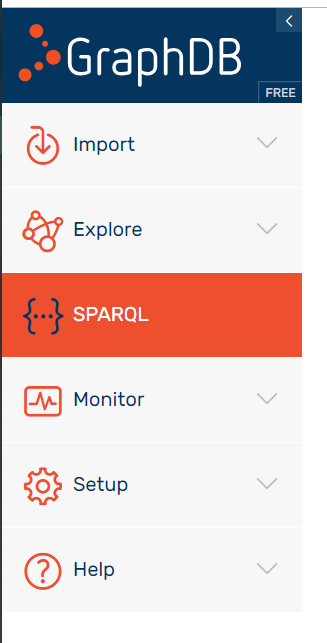
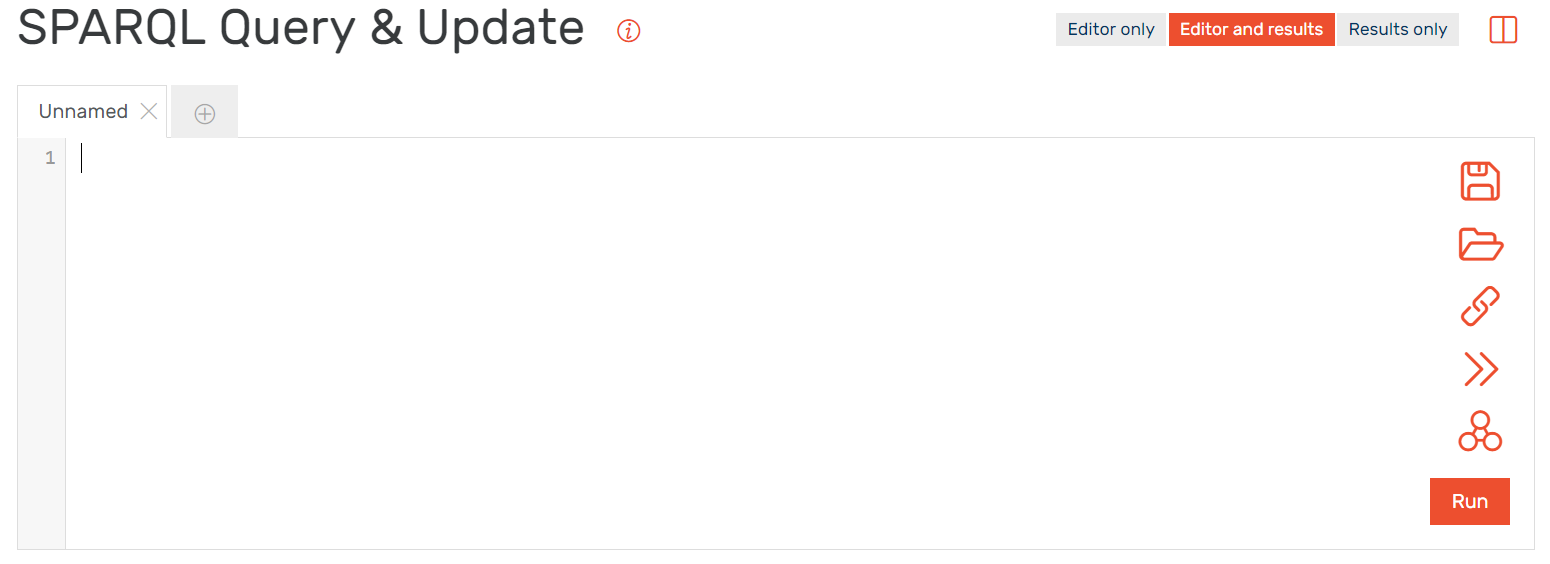
* Thứ hai tải file như hình dưới về máy tạo một thư mục trong project để nó trong đó.
* Thứ ba, kích chuột phải vào file.jar rồi chọn *Build Path* rồi chọn *Buil.*
* Thứ tư, kích chuột phải vào project chọn *Maven -> Update project.* Vậy là đã xog phần chuẩn bị API.

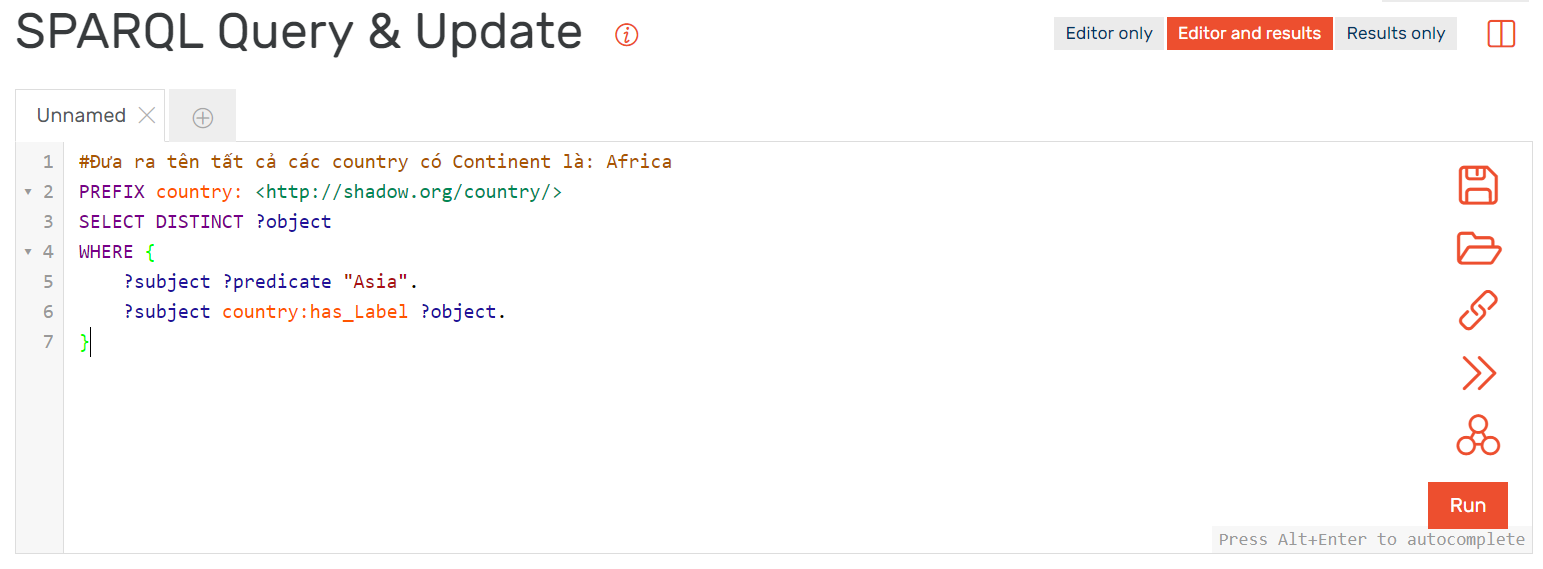
## Cách truy vấn

Graphdb lưu trữ dưới dạng các triples (subject – predicate – object). Để truy vấn thì ta dùng ngôn ngữ SPARQL.

Ví dụ về 1 câu lệnh SparQL: “SELECT ?s ?p ?o WHERE {?s ?p ?o. }” nó sẽ trả về kết quả 3 biến s, p, o chứa các giá trị tương ứng với dạng của nó thỏa mãn mệnh đề trong WHERE. Thông thường thì s, p đều là các IRI còn o thì có thể là literal hoặc IRI. Dấu “.” Biểu thị cho kết thúc rằng buộc.

### 5.1. Truy vấn trực tiếp trên server

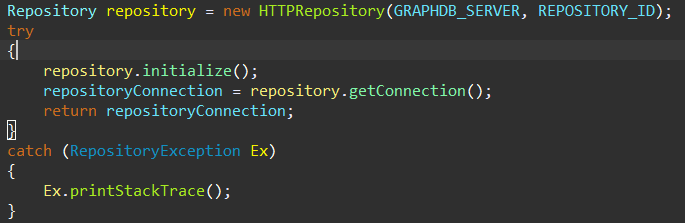
* Bước 1: Mở server lên tại: <http://localhost:7200/>
* Bước 2: Chọn mục SPARQL.
* Bước 3: Nhập truy vấn trong ô bên phải như hình dưới rồi ấn *Run*:

Ví dụ: Câu lệnh truy vấn trên server.

### 5.2. Truy vấn thông qua API

Trước tiên cần khởi tạo và kết nối đến server:

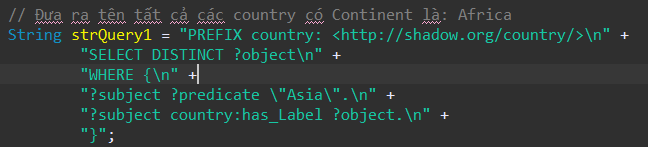
* Đầu tiên cần tạo 2 String và khai báo 1 đối tượng RepositoryConnection như hình dưới: Cái đầu tiên là chứa địa chỉ truy cập server, cái thứ 2 là chứa tên của Repository.
* Tiếp theo tạo đối tượng Repository thông qua đối tượng con là HTTPRepository để chuẩn bị cho khởi tạo ở câu lênh *repository.initialize().*

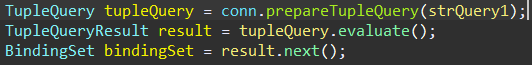


* Rồi dùng lệnh *repository.getConnection()* để bắt đầu kết nối.

Sau đó ta có thể truy vấn như sau:

Ví dụ với câu truy vấn như dưới đây trên server:

Sẽ được viết lại dưới dạng String như sau:

Rồi truyền vào như sau:

Thực thi 2 câu lệnh đầu nó sẽ trả về một đối tượng TupleQueryResult bao gồm một chuỗi các đối tượng BindingSet. Mỗi BindingSet chứa một tập các Binding. Một Binding là một cặp giữa tên biến và giá trị của nó. Ví dụ như subject (tên biến) – IRI (giá trị của nó).

Để lấy giá trị của tên biến thì ta dùng câu lệnh như dưới đây:

Với biến “line” có thể là Value hoặc có thể là String, ở đây thì “line” là String.

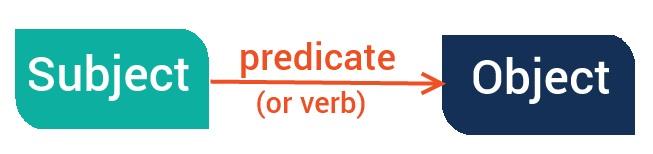
**Lưu ý:** Tên biến truyền vào trong bindingSet.getValue() phải trùng tên biến trong mệnh đề SELECT.

# III. Mô hình dữ liệu

## Mô hình dữ liệu

Các tài nguyên có thể được mô tả thông qua các phát biểu cho phép xác định thuộc tính và giá trị. RDF sử dụng bộ ba Triples để nói về những thành phần khác nhau của phát biểu. Một bộ ba được mô tả dưới dạng nút-cung-nút, cung có hướng bắt đầu từ subject đến object

* Subject (chủ ngữ): Là thành phần xác định đối tượng mà phát biểu đề cập
* Predicate (vị ngữ): Là thành phần xác định thuộc tính của chủ ngữ trong phát biểu
* Object (tân ngữ): Là thành phần xác định giá trị của thuộc tính, có thể là một tài nguyên (URI), hoặc một giá trị (Literal)



Tập hợp các bộ ba hợp lại tạo thành đồ thị RDF, các nút trong đồ thị có thể là subject hoặc object, các cung trong đồ thị là predicate.

Áp dụng vào vào dữ liệu với đề tài mà nhóm đã được phân công thì mô hình dữ liệu có dạng cụ thể như sau:

* Đối với thực thể thì sẽ có dạng: subject – predicate – object với mỗi thuộc tính là:
* Subject: Id được đổi thành dạng IRI (<http://shadow.org/country/country1>)
* Predicate: Ở dạng IRI (<http://shadow.org/country/has_Label>)
* Object: Ở dạng Literal (Ba Lan)
* Mỗi thực thể có 5-6 triple tùy vào thực thể đó có bao nhiêu thuộc tính.
* Đối với quan hệ thì sẽ có dạng: subject – predicate – object là:
* Subject: Id của thực thể 1 có dạng IRI (<http://shadow.org/country/country1>)
* Predicate: Quan hệ có dạng IRI (<http://shadow.org/relationship/cooperate>)
* Object: Id của thực thể 2 có dạng IRI (<http://shadow.org/country/country670>)

## Giải thuật sinh và upload dữ liệu

* Do cần phải sinh ngẫu nhiên các thực thể để có thể đẩy lên lưu trữ trên server cho nên ta cần có dữ liệu lưu trữ dưới dạng file, các file này lưu trữ các thuộc tính của các thực thể và các quan hệ. Từ đó ta đọc file và lưu trữ vào ArrayList<String>.
* Sau khi đọc các file xog ta tiến hành tạo các thực thể một cách ngẫu nhiên rồi đưa tất cả vào 1 ArrayList<Entity> (ArrayList<String> này chứa được tất cả các thực thể khác vì nó là lớp cha của các lớp còn lại).
* Sau khi tạo random xong các thực thể ta tiến hành tạo List các quan hệ giữa 2 thực thể bất kỳ nhặt random giữa các thực thể đã tạo.
* Sau khi tạo các thực thể và quan hệ xong ta tiến hành tạo các triples (subject-predicate-object) để có thể upload lên server.
* Sau khi tạo các triples xong ta tiến hành upload dữ liệu đã tạo lên server ở dạng triples.

Ví dụ cụ thể: Tạo triple cho 1 person bất kỳ.

* Câu lệnh dưới đây là câu lệnh tạo IRI thực thể person kèm theo Id của thực thể đó.

IRI subject = valueFactory.createIRI(PERSON, "/person" + person.getId());

* Câu lệnh dưới đây tạo biến IRI (Predicate) của thực thể person trên.

IRI predicate = valueFactory.createIRI(PERSON, LABEL);

* Câu lệnh dưới đây là tạo Literal (Object) cho tên của person.

Literal object = valueFactory.createLiteral(person.getName());

* Câu lệnh tiếp là add vào model (nơi lưu trữ các triples trước khi upload lên server).

model.add(subject, predicate, object);

* Để upload lên server ta dùng lệnh.

conn.add(model)

Trong đó:

PERSON: Là IRI của thực thể person có dạng: “<http://shadow.org/person>”.

LABEL: Là IRI của các label có dạng: "/has\_Label".

Sau khi tạo xog IRI predicate nó sẽ có dạng: “<http://shadow.org/person>/has\_Label”.

conn: Là đối tượng *RepositoryConnection* đã được tạo.

Tương tự với các thuộc tính khác của các thực thể cũng như quan hệ, ta cũng thêm nó vào 1 model rồi sau đó sẽ upload lên server. Việc này sẽ giảm thời gian sinh và upload thực thể, quan hệ lên server so với việc upload thẳng từng triple lên.

# IV. Chương trình

## Biểu đồ thể hiện sự phụ giữa các package

### Package model

Package có chức năng chứa các class thực thể như:

* Entity: Chứa các thuộc tính và phương thức chung của các class khác như person, time, … cho nên class này sẽ được các class thực thể khác kế thừa. Class này chứa các thuộc tính chung như: name, desciption, … và các phương thức get(), set() của nó.
* Person: Class này chứa các thông tin của person ngoài các thông tin đã kế thừa từ lớp cha là lớp Entity, các thuộc tính riêng như id, job và các phương thức get(), set() của từng thuộc tính, còn có cả constructor gọi đến constructor của lớp cha là lớp Entity và khởi tạo một đối tượng person. Ngoài ra nó còn có phương thức toString().
* Country: Class này chứa các thông tin của country ngoài các thông tin đã kế thừa từ lớp cha là lớp Entity. Nó có một số thuộc tính như: id, continent và các phương thức get(), set() của từng thuộc tính đó, nó cũng có constructor gọi đến constructor của lớp cha là lớp Entity để khởi tạo đối tượng country.. Ngoài ra nó còn có phương thức toString().
* Time: Class này chứa thuộc tính id và các phương thức get(), set() của nó, nó cũng chứa constructor gọi đến constructor của lớp cha là lớp Entity để khởi tạo một đối tượng time. Ngoài ra nó còn có phương thức toString().
* Location: Class này chứa thuộc tính id và phương thức get(), set(). Nó cũng chứa constructor gọi đến constructor của lớp cha để khởi tạo thực thể location. Ngoài ra nó cũng có phương thức toString().
* Organization: Class này cũng chứa các thuộc tính của riêng nó như id, headquarter cùng các phương thức get(), set() của từng thuộc tính. Nó cũng chứa constructor gọi đến constructor của lớp cha để tạo ra thực thể organization. Ngoài ra thì nó cũng có phương thức toString().
* Event: Class này chứa thuộc tính id và phương thức get(), set() của nó, nó cũng có constructor để gọi đến constructor của lớp cha mỗi khi tạo ra thực thể event. Nó cũng có phương thức toString().

### Package main

Package này chứa class main chứa phương thức main, tạo ra các đối tượng ở trong phương thức main và gọi đến các phương thức của từng đối tượng để thực thi chương trình.

### Package graphdb

Package này chứa các class liên quan đến graphdb, các class như sau:

* Connection: Class này chứa thuộc tính về đường link server và tên repository để dùng cho khởi tạo kết nối, chứa phương thức getRepositoryConnection() dùng để khởi tạo kết nối và dùng phương thức closeConnection() để đóng kết nối khi cần thiết.
* Query: Class này có thuộc tính là đối tượng của lớp Connection, thuộc tính start, end để đo thời gian truy vấn, chứa các phương thức truy vấn cơ bản và nâng cao, nó còn có thêm phương thức đóng kết nối.
* CreateIRIRelation: Class này chứa các thuộc tính là đối tượng của lớp Connection, RepositoryConnection, ValueFactory và các thuộc tính IRI cũng như các thuộc tính String lưu tên các namspace và các ArrayList<IRI> để lưu các quan hệ giữa các thực thể. Lớp này chứa các phương thức tạo IRI cho relation và constructor khởi tạo. Lớp này kế thừa từ lớp ReadFile ở package file.io để lấy các ArrayList<String> quan hệ để biến đổi thành ArrayList<IRI>.
* UploadData: Class này chứa các thuộc tính là đối tượng của lớp Connection phục vụ cho kết nối đưa data lên, nó cũng chứa các thuộc tính là các ArrayList của các thực thể và IRI phục vụ cho việc tạo IRI là upload lên server. Nó cũng có phương thức khởi tạo constructor để khởi tạo các thuộc tính ở trên. Nó có các phương thức upLoadEntity() để upload thực thể lên server, nó có phương thức upLoadRelationData() để upload các relation lên server, có phương thức khởi tạo và đóng kết nối tới server khi mà upload xong. Lớp này kế thừa từ lớp CreateIRIRelation để lấy cái list IRI quan hệ từ bên lớp đó.

### Package file.io

Package này chứa các lớp liên quan đến thao tác với file:

* FileNameConstant: Class này chứa các thuộc tính là tên đường dẫn tương đối đến các file trong project để cho lớp ReadFile có thể kế thừa và sử dụng.
* ReadFile: Class này có thuộc tính là các ArrayList<String> dùng để chứa thông tin của các thực thể cũng như quan hệ, nó chứa constructor để khởi tạo các ArrayList và gọi đến các phương thức đọc file.
* WriteFile: Class này được sử dụng để ghi kết quả truy vấn, nó chứa phương thức ghi file có tham số là ArrayList<String> chứa kết quả truy vấn.

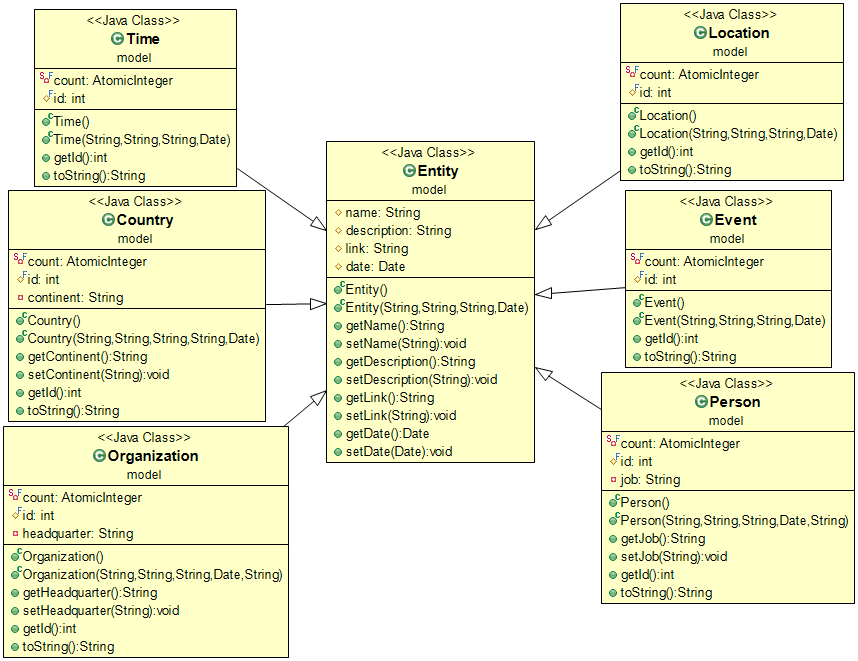
### Package create.entity

Package này có chức năng là chứa các lớp phục vụ cho việc tạo thực thể, các lớp như sau:

* CreateCountry: Class này chứa phương thức để tạo ra một thực thể country với các thông tin name, description, continent, link, date và trả về một thực thể country. Class này nhận tham số là các ArrayList<String> chứa thông tin về thực thể country. Class này kế thừa từ class RandomDate để sinh ngẫu nhiên date.
* CreateEvent: Class này chứa phương thức để tạo ra một thực thể event với các thông tin name, description, link, date và trả về một thực thể event. Class này nhận tham số là các ArrayList<String> chứa thông tin về thực thể event. Class này kế thừa từ class RandomDate để sinh ngẫu nhiên date.
* CreateLocation: Class này chứa phương thức để tạo ra một thực thể location với các thông tin name, description, link, date và trả về một thực thể location. Class này nhận tham số là các ArrayList<String> chứa thông tin về thực thể location. Class này kế thừa từ class RandomDate để sinh ngẫu nhiên date.
* CreateOrganization: Class này chứa phương thức để tạo ra một thực thể organization với các thông tin name, description, headquarter, link, date và trả về một thực thể organization. Class này nhận tham số là các ArrayList<String> chứa thông tin về thực thể organization. Class này kế thừa từ class RandomDate để sinh ngẫu nhiên date.
* CreatePerson: Class này chứa phương thức để tạo ra một thực thể person với các thông tin name, description, job, link, date và trả về một thực thể person. Class này nhận tham số là các ArrayList<String> chứa thông tin về thực thể person. Class này kế thừa từ class RandomDate để sinh ngẫu nhiên date.
* CreateTime: Class này chứa phương thức để tạo ra một thực thể time với các thông tin name, description, link, date và trả về một thực thể time. Class này nhận tham số là các ArrayList<String> chứa thông tin về thực thể time. Class này kế thừa từ class RandomDate để sinh ngẫu nhiên date.
* RandomDate: Class này chứa phương thức getDateRandom() trả về một đối tượng của lớp Date dùng cho việc sinh ngẫu nhiên date của các thực thể ở trên, class còn có phương thức randBetween() để giúp tạo random ra một số nằm giữa 2 số là tham số truyền vào.
* CreateRandomEntity: Class này kế thừa từ lớp ReadFile để lấy các ArrayList<String> thông tin về các thực thể để có thể tạo ra random một thực thể, class này chứa thuộc tính là các đối tượng của các lớp cùng gói như CreatePerson, CreateTime, … và chứa thuộc tính là đối tượng của lớp Entity trong package model để tạo ra 1 list thực thể Entity chứa đủ các thực thể được random khác như person, time, … nó cũng có constructor để khởi tạo đối tượng và gọi đến constructor của lớp cha,class có phương thức getListRandomEntity() nhận tham số là số lượng thực thể cần sinh random, từ đó nó gọi đến phương thức randomEntity() để lựa chọn cũng như sinh ngẫu nhiên một thực thể trong 6 thực thể rồi add vào list Entity. Từ đó ta có được list Entity chứa các đối tượng person, time, … được sinh ngẫu nhiên.

## Biểu đồ lớp

### Biểu đồ lớp các thực thể



Các class như: Person, Orgranization, Location, Country, Event, Time đều kế thừa từ class Entity bởi vì chúng sẽ dùng chung các thuộc tính như name, description, link, date và các phương thức get(), set() của các thuộc tính đó giúp ta giảm thiểu được code và dễ dàng quản lý, bảo trì dễ hơn.

### Biểu đồ các lớp sinh thực thể

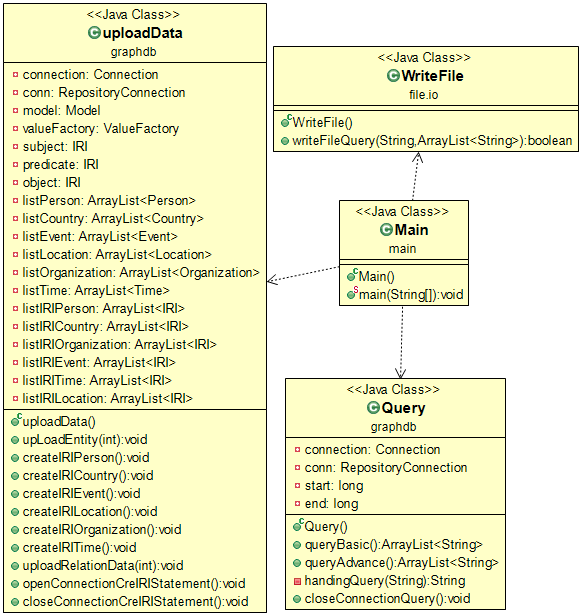
* Ta có các lớp CreateCountry, CreateTime, … kế thừa từ lớp RandomDate.
* CreateRandomEntity có quan hệ Composition với các lớp khác như trong hình vẽ.

### Biểu đồ upload thực thể

* Lớp UploadData kế thừa từ CreateIRIRelation. Giúp ta giảm được lượng code dễ bảo trì hơn.
* Lớp UploadData có quan hệ Dependency với lớp CreateRandomEntity.
* Lớp UploadData có quan hệ Composition với lớp Connection.
* Lớp CreateRandomEntity có quan hệ Composition với lớp Connection.

### Biểu đồ chạy hàm main

* Lớp Main phụ thuộc vào các lớp còn lại.



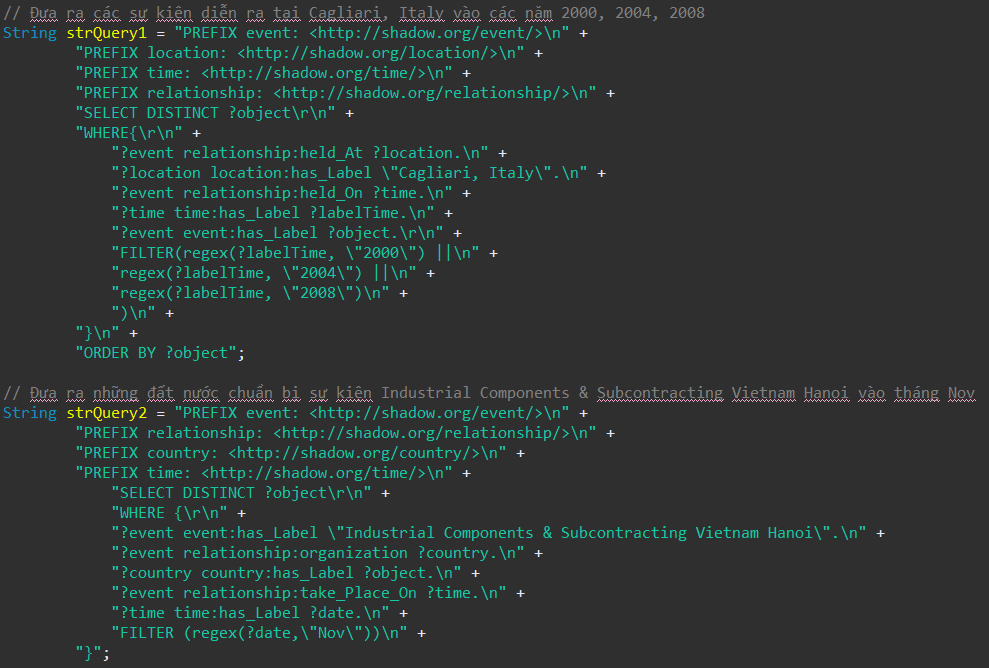
# V. Kiểm thử hiệu năng

## 1. Các câu truy vấn

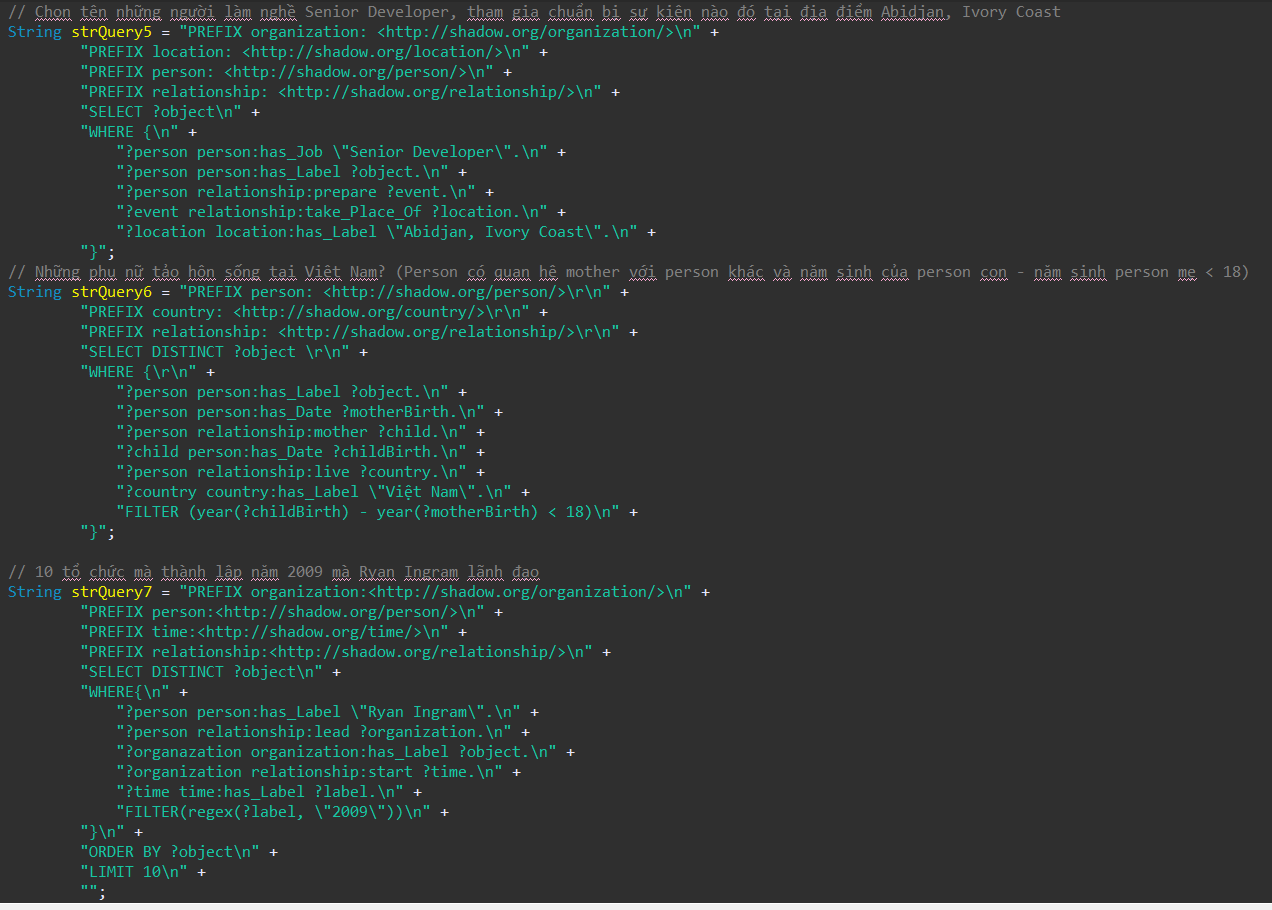
### 1.1. Truy vấn cơ bản

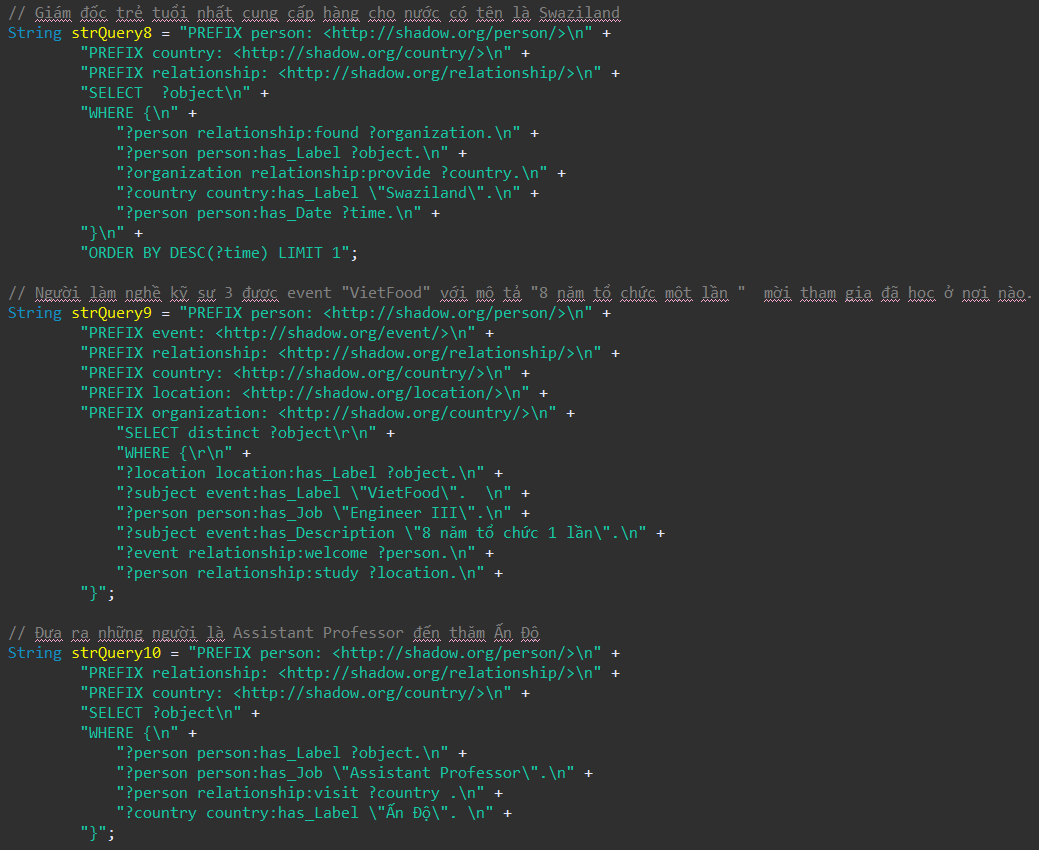


### 1.2. Các câu truy vấn nâng cao









## 2. Đo thời gian truy vấn

Vì thời gian được tính bằng code Java nên kết quả có thể không được chính xác hoàn toàn và lâu hơn thời gian chính xác của nó (phụ thuộc cả vào máy). Thời gian được tính theo (ms).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (N, M) | TVCB1 | TVCB2 | TVCB3 | TVCB4 | TVCB5 | TVCB6 | TVCB7 | TVCB8 | TVCB9 | TVCB10 |
| 10000, 20000 | 67 | 20 | 11 | 6 | 4 | 6 | 5 | 10 | 22 | 9 |
| 40000, 56000 | 68 | 25 | 9 | 9 | 6 | 12 | 6 | 147 | 15 | 17 |
| 300000, 400000 | 2516 | 213 | 525 | 74 | 31 | 617 | 208 | 1229 | 34 | 75 |
| 1000000, 2000000 | 5912 | 804 | 634 | 83 | 36 | 701 | 668 | 2118 | 86 | 719 |
| 4000000, 8000000 | 6271 | 1239 | 2555 | 1155 | 93 | 8192 | 1146 | 13318 | 356 | 1051 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| (N, M) | TVNC1 | TVNC2 | TVNC3 | TVNC4 | TVNC5 | TVNC6 | TVNC7 | TVNC8 | TVNC9 | TVNC10 |
| 10000, 20000 | 8 | 5 | 4 | 5 | 4 | 1 | 7 | 0 | 16 | 4 |
| 40000, 56000 | 9 | 6 | 5 | 4 | 4 | 7 | 15 | 2 | 16 | 4 |
| 300000, 400000 | 308 | 84 | 147 | 254 | 80 | 63 | 198 | 64 | 240 | 82 |
| 1000000, 2000000 | 564 | 411 | 425 | 449 | 138 | 128 | 1242 | 148 | 955 | 130 |
| 4000000, 8000000 | 2712 | 2520 | 2416 | 2604 | 1028 | 586 | 4933 | 849 | 3732 | 402 |

## Kết luận

### Ưu điểm

* Server cài đặt dễ dàng, tương thích trên nhiều nền tảng.
* Giao diện đẹp dễ sử dụng, tạo được nhiều repository, quan sát được biểu đồ, …
* Tuy mới thử nghiệm trên bản free mà cũng được lượng lưu trữ khá cao cho thấy server hỗ trợ lưu trữ lượng lớn tốt.
* Kết nối API dễ dàng.

### Nhược điểm

* Do là bản free nên chưa thực hiện được nhiều câu query trên server cùng lúc.
* Khi chạy chương trình server ngốn ram, vì dữ liệu được lưu tạm trên heap trên ram.
* Tạo kết nối trên java hơi phức tạp.
* Khi lưu trữ với một lượng lớn (>= 20tr) thì việc tiếp tục lưu trữ chậm lại rõ rệt.
* Document hướng dẫn còn khó hiểu, chưa rõ ràng.