TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

──────── \* ───────



BÀI TẬP LỚN OOP

Đề tài: GraphDB

NGÀNH CÔNG NGHỆ THÔNG TIN

Giảng viên hướng dẫn: TS.Trịnh Tuấn Đạt

Mã lớp: 104404

Nhóm: 01

Sinh viên thực hiện:

Nguyễn Văn Hùng - 20161972 20%

Nguyễn Đức Danh - 20160600 25%

Hoàng Mạnh Hiệp - 20161438 25%

Tạ Trung Kiên - 20162255 20%

Nguyễn Tuấn Dương - 20155304 10%

# Đóng góp của các thành viên trong nhóm

**1. Nguyễn Văn Hùng - 20161972**

- Đóng góp : 20%

- Nhiệm vụ đóng góp: sinh dữ liệu

**2. Nguyễn Đức Danh - 20160600**

- Đóng góp: 25%

- Nhiệm vụ đóng góp: kết nối graphdb và truy vấn

**3. Hoàng Mạnh Hiệp**

- Đóng góp:25%

- Nhiệm vụ đóng góp: kết nối graphdb và truy vấn

**4. Tạ Trung Kiên**

- Đóng góp:20%

- Nhiệm vụ đóng góp: Làm báo cáo và bài tập nhóm

**5. Nguyễn Tuấn Dương**

- Đóng góp:10%

- Nhiệm vụ đóng góp: làm bài tập nhóm

Nội dung

[Đóng góp của các thành viên trong nhóm 2](#_Toc533405301)

[PHẦN I: GIỚI THIỆU VỀ CÔNG NGHỆ GRAPHDB 4](#_Toc533405302)

[1. GIỚI THIỆU 4](#_Toc533405303)

[2. KIẾN TRÚC VÀ THÀNH PHẦN 4](#_Toc533405304)

[3. HƯỚNG DẪN CÀI ĐẶT 6](#_Toc533405305)

[4. SỬ DỤNG 6](#_Toc533405306)

[PHẦN II: DỮ LIỆU 7](#_Toc533405307)

[1. Cấu trúc mô hình dữ liệu 7](#_Toc533405308)

[2. Giải thuật sinh dữ liệu 7](#_Toc533405309)

[} 9](#_Toc533405310)

[PHẦN III:CHƯƠNG TRÌNH 10](#_Toc533405311)

[1. Các package 10](#_Toc533405312)

[2. Biểu đồ sự phuộc giữa các package 10](#_Toc533405313)

[3. Các lớp trong chương trình 10](#_Toc533405314)

[1. Danh sách các lớp và chức năng của từng lớp 10](#_Toc533405315)

[2. Các phương thức quan trọng trong từng lớp 11](#_Toc533405316)

[3. Biểu đồ quan hệ giữa các package 12](#_Toc533405317)

[4. Biểu đồ quan hệ giữa các lớp 12](#_Toc533405318)

[PHẦN IV: KIỂM THỬ HIỆU NĂNG CỦA HỆ THỐNG 13](#_Toc533405319)

[1. Các câu truy vấn cơ bản 13](#_Toc533405320)

[2. Các câu truy vấn phức tạp 13](#_Toc533405321)

[3. Độ lớn dữ liệu sinh và kết quả thu được 14](#_Toc533405322)

[4. Ưu nhược điểm của công nghệ sử dụng 14](#_Toc533405323)

[PHẦN IV: KẾT LUẬN 15](#_Toc533405324)

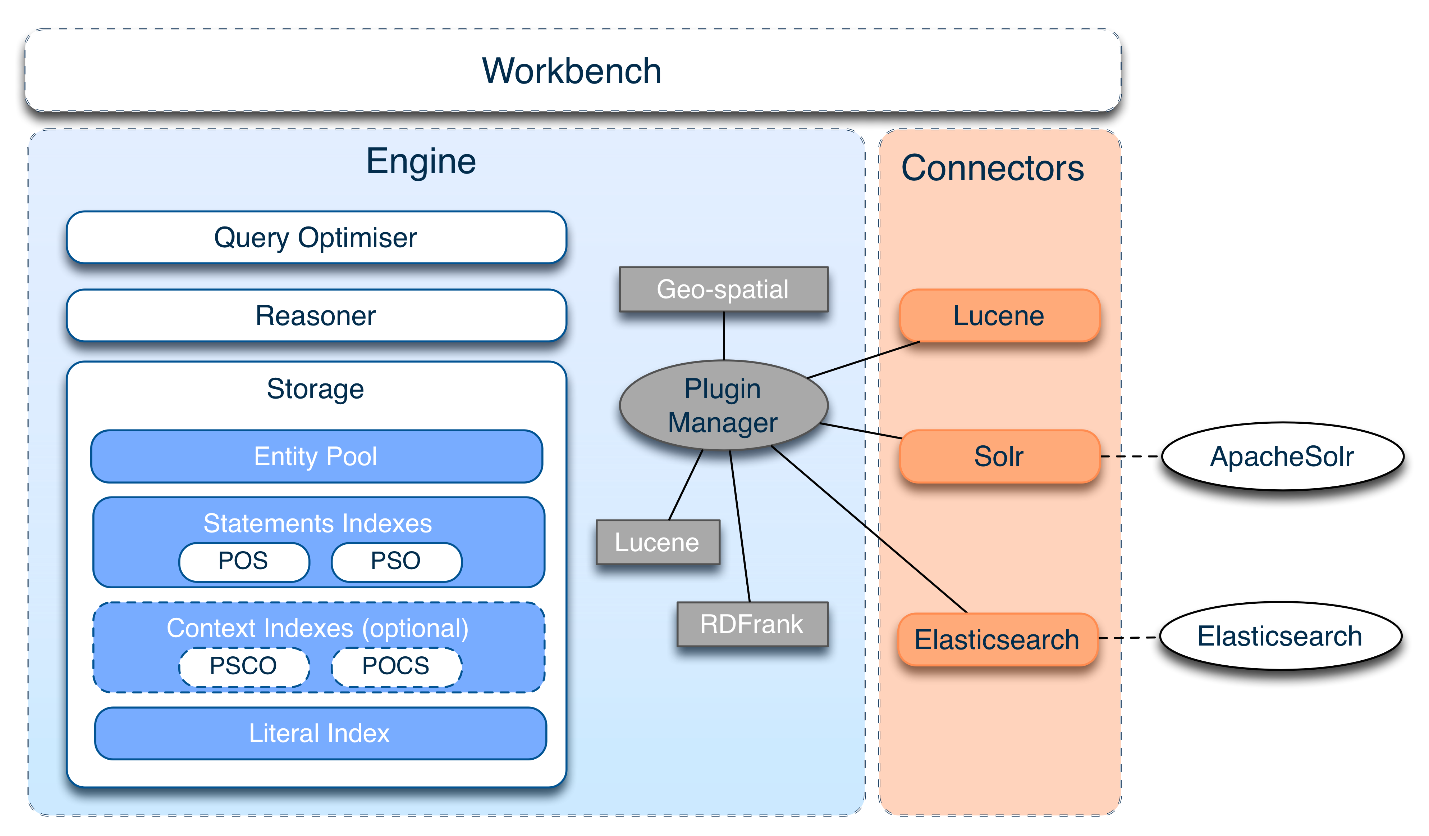
# 

# PHẦN I: GIỚI THIỆU VỀ CÔNG NGHỆ GRAPHDB

## GIỚI THIỆU

GraphDB là một nhóm các cơ sở dữ liệu RDF hiệu quả cao, mạnh mẽ và có thể mở rộng. Nó hợp lý hóa tải và sử dụng các bộ dữ liệu đám mây dữ liệu được liên kết cũng như các tài nguyên của riêng bạn. Để dễ sử dụng và tương thích với các tiêu chuẩn công nghiệp, GraphDB triển khai các giao diện khung RDF4J, đặc tả Giao thức SPARQL của W3C và hỗ trợ tất cả các định dạng tuần tự hóa RDF. Cơ sở dữ liệu là lựa chọn ưa thích của cả nhà phát triển độc lập nhỏ và tổ chức doanh nghiệp lớn, vì hỗ trợ cộng đồng và thương mại, các tính năng doanh nghiệp tuyệt vời, như hỗ trợ cụm và tích hợp với các ứng dụng tìm kiếm hiệu suất cao bên ngoài - Lucene, SOLR và ElSTERearch.

## KIẾN TRÚC VÀ THÀNH PHẦN



* Kiến trúc

GraphDB được đóng gói dưới dạng Storage and Inference Layer (SAIL) cho RDF4J và sử dụng rộng rãi các tính năng và cơ sở hạ tầng của RDF4J, đặc biệt là mô hình RDF, trình phân tích cú pháp RDF và công cụ truy vấn.

GraphDB triển khai giao diện sail API để có thể tích hợp nó với phần còn lại của khung RDF4J, ví dụ: các công cụ truy vấn và giao diện người dùng web. Một ứng dụng người dùng có thể được thiết kế để sử dụng GraphDB trực tiếp thông qua API RDF4J SAIL hoặc thông qua các giao diện chức năng cấp cao hơn. Khi kho lưu trữ GraphDB được hiển thị bằng Máy chủ HTTP RDF4J, người dùng có thể quản lý kho lưu trữ thông qua Workbench hoặc Workbench RDF4J hoặc các công cụ khác được tích hợp với RDF4J.

- RDF4J là gì?

RDF4J là một framework để lưu trữ, truy vấn và lập luận với dữ liệu RDF. Nó được Aduna triển khai trong Java như một dự án nguồn mở và bao gồm các phần lưu trữ khác nhau (bộ nhớ, tệp, cơ sở dữ liệu), ngôn ngữ truy vấn, lý do và giao thức máy chủ-máy khách.

Có 2 cách sử dụng RDF4J

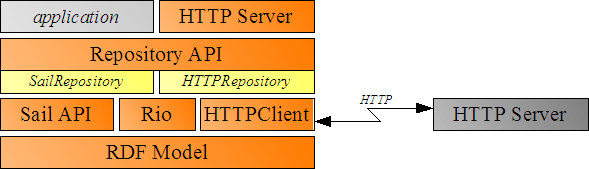
- Xem như một máy chủ độc lập

- Nhúng trong ứng dụng như một thư viện Java

RDF4J hỗ trợ ngôn ngữ truy vấn SP3QL của W3C. Nó cũng hỗ trợ các định dạng tệp RDF phổ biến nhất và các định dạng kết quả truy vấn.

RDF4J cung cấp API người dùng giống như JDBC, API hệ thống được sắp xếp hợp lý và giao diện RESTful HTTP

Kiến trúc của RDF4J



- Sail API là một bộ giao diện Java hỗ trợ lưu trữ, truy xuất, xóa và suy luận RDF. Nó được sử dụng để trừu tượng hóa từ cơ chế lưu trữ thực tế, ví dụ: việc triển khai có thể sử dụng cơ sở dữ liệu quan hệ, hệ thống tệp, lưu trữ trong bộ nhớ, v.v.

+ Tính linh hoạt và tự do để tối ưu hóa để có thể xử lý hiệu quả lượng dữ liệu khổng lồ trên các máy cấp doanh nghiệp;

+ khả năng mở rộng sang các ngôn ngữ dựa trên RDF khác

+ kiểm soát đồng thời cho bất kỳ loại kho lưu trữ

* Thành phần

- Engine

- Conector

- Workbench

Workbench là công cụ quản trị dựa trên web GraphDB. Giao diện người dùng tương tự như Ứng dụng Web RDF4J Workbench, nhưng có nhiều chức năng hơn.:

+ Quản lí kho lưu trữ

+ Tải và xuất dữ liệu

+ Quản lí namespace, context

+ Xem và chỉnh sửa mã nguồnRDF

+ Giám sát truy vấn, mã nguồn.

+ Quản lí người dùng, quyền truy cập

+ Quản lí kết nối

+ Cung cấp API REST để tự động hóa các tác vụ khác nhau để quản lý và quản lý kho lưu trữ

## HƯỚNG DẪN CÀI ĐẶT

* Yêu cầu

- 2GB Memory

- 2GB Disk Space

- Java SE Development 8 trở lên

- Bản quyền : GraphDB Free có sẵn theo giấy phép miễn phí giống như RDBMS. Có thể sử dụng nhưng không phải là nguồn mở. Trước khi phân phối lại GraphDB Free, cần liên hệ graphdb-info@ontotext.com để nhận được sự cho phép.

* Quá trình cài đặt

- Tải và giải nén

- Triển khải thư mục “.war” lên ứng dụng

- Sử dụng GraphDB Workbench

- Tạo Cơ sở dữ liệu và kho lưu trữ

- Thực thi truy vấn SPARQL

## SỬ DỤNG

- Nhập dữ liệu sử dụng Workbench

- Khai phá dữ liệu

- Truy vấn dữ liệu

- Xuất dữ liệu

- Sử dụng với RDF4J API

# PHẦN II: DỮ LIỆU

## Cấu trúc mô hình dữ liệu

* Dữ liệu sinh ra được tổ chức theo mô hình RDF và thêm vào cơ sở dữ liệu.
* Trong mô hình RDF, mỗi thực thể được coi là một Statement. Mỗi câu lệnh bao gồm ba phần ( Nên nó cũng hay được gọi là bộ ba S-P-O):
* Subject (S): Nút bắt đầu của Statement, đại diện cho tài nguyên đó về cái gì.
* Predicate (P): Là thuộc tính thể hiện mỗi quan hệ giữa hai nút.
* Object (O): Là nút kết thúc một Statement, thường là các value (giá trị).



## Giải thuật sinh dữ liệu

* Trong package Manager: Dữ liệu từ các file text sẽ được nạp vào chương trình và lưu dữ trong những Arraylist của từng đối tượng riêng biệt.

public ArrayList<Person> personList;

public ArrayList<Country> countryList;

public ArrayList<Organization> organizationList;

public ArrayList<Event> eventList;

public ArrayList<Location> locationList;

public ArrayList<NguonGoc> nguonGocList;

public ArrayList<String> quanHeList;

public ArrayList<String> timeList;

Dữ liệu được đọc theo từng dòng trong File text đã được sinh sẵn:

Ví dụ khi nạp nguồn gốc vào nguonGocList

// nap danh sach NguonGoc vao nguonGocList

// Doc 2 dong trong file -> NguonGoc

BufferedReader br = new BufferedReader(newFileReader(nguonGocFile);

String line = null;

while ((line = br.readLine())!=null) {

NguonGoc nguonGoc = new NguonGoc();

nguonGoc.setLink(line);

line = br.readLine();

nguonGoc.setNgayTRichRut(line);

nguonGocList.add(nguonGoc);

n++; // đếm số lượng

}

Những thực thể có đối tượng nguonGoc sẽ được thêm nguonGoc khi đọc vào từ từ file ngẫu nhiên.

// nap danh sach Person vao personList

// Doc 4 dong lien tiep trong file ->Person

br = new BufferedReader(new FileReader(personFile));

line = null;

while ((line = br.readLine()) != null) {

Person person = new Person();

person.setDinhDanh(line); // add dinh danh

line = br.readLine();

person.setNhanHienThi(line); // add nhan hien thi

line = br.readLine();

person.setChucVu(line); // add Chuc vu

line = br.readLine();

person.setMoTa(line); // add mo ta

person.setNguonGoc(nguonGocList.get(rand(0,n-1)));

personList.add(person);

}

Lưu

* Các thực thể và quan hệ sau khi đã nạp vào các ArrayList tương ứng sẽ được inport lên Graphdb theo từng đối tượng

Hàm import thực thể lên GraphDB:

**public** String insertEntity(ThucThe thucthe, String str) {

System.***out***.println(thucthe.getDinhDanh());

**return** strInsert = "INSERT DATA {" + "" + str + " <http://TEST/identification> \"" + thucthe.getDinhDanh()

+ "\" ." + "" + str + " <http://TEST/description> \"" + thucthe.getMoTa() + ".\" ." + "" + str

+ " <http://TEST/label> \"" + thucthe.getNhanHienThi() + "\" ." + "" + str + " <http://TEST/Ngayrut> \""

+ thucthe.getNguonGoc().getNgayTRichRut() + "\" ." + "" + str + " <http://TEST/Link> \""

+ thucthe.getNguonGoc().getLink() + "\" ." + "}";

}

Import thực thể Country lưu trữ trên GraphDB

**public** **void** insertCountry() {

// RepositoryConnection repositoryConnection = null;

Country country = **new** Country();

**for** (**int** i = 0; i < manager.personList.size(); i++) {

country = manager.countryList.get(i);

// for (int j = 0; j < manager.countryList.size(); j++) {

str = "<http://TEST/" + country.getDinhDanh() + ">";

strInsert = "INSERT DATA {" + "" + str + " <http://TEST/ThuDo> \"" + country.getThuDo() + "\" ." + "}";

**try** {

repositoryConnection = getRepositoryConnection();

// repositoryConnections = getRepositoryConnection();

insert(repositoryConnection, strInsert);

insert(repositoryConnection, insertEntity(country, str));

} **catch** (Throwable t) {

t.getMessage();

} **finally** {

repositoryConnection.close();

// repositoryConnections.close();

}

}

# }

* Sau khi đã import hết cái thực thể và quan hệ lên GraphDB lưu trữ ta tiếp tục nhặt ngẫu nhiên các thực thể và relationship và lưu lại tạo thành quan hệ để truy vấn
* **public** **void** insertRelationship(String filePath) {
* **for** (**int** i = 0; i < 500; i++) {
* listRelationship = manager.getRelationShip(filePath);
* **int** r1 = (**int**) Math.*floor*(Math.*random*() \* manager.personList.size());
* **int** r2 = (**int**) Math.*floor*(Math.*random*() \* manager.personList.size());
* **int** r3 = (**int**) Math.*floor*(Math.*random*() \* listRelationship.size());
* String s = manager.personList.get(r1).getDinhDanh();
* String p = listRelationship.get(r3);
* String o = manager.personList.get(r2).getDinhDanh();
* strInsert = "PREFIX : <http://TEST/>" + "INSERT DATA {" + ":" + s + " :" + p + " :" + o + " ." + "}";
* **try** {
* repositoryConnection = getRepositoryConnection();
* insert(repositoryConnection, strInsert);
* } **catch** (Throwable t) {
* t.getMessage();
* } **finally** {
* repositoryConnection.close();
* }
* }
* }

Quan hệ hoàn chỉnh sẽ gồm có 3 phần S-P-O.

# PHẦN III:CHƯƠNG TRÌNH

## Các package

|  |  |
| --- | --- |
| Các package | Chức năng |
| application | * Làm việc với Graphdb * Tạo và Lưu giữ các quan hệ lên Graphdb * Truy vấn hiển thị kết quả truy vấn * Kiểm thử |
| create | * Khởi dựng các các thực thể , quan hệ , nguồn gốc trích rút, thời gian |
| Createdata | * Nạp dữ liệu cho cho các đối tượng từ file text |

## Biểu đồ sự phuộc giữa các package

## Các lớp trong chương trình

### Danh sách các lớp và chức năng của từng lớp

|  |  |  |
| --- | --- | --- |
| Lớp | package | Chức năng |
| Insert | application | Insert dữ liệu chọn ngẫu nhiên lên Graphdb để lưu trữ |
| Main | application | Thực hiện truy vấn và ghi thông tin ra file Log |
| Query | application | Thực hiện các câu truy vấn đơn giản và nâng cao |
| ThucThe | create | Tạo đối tượng thực thể , các đội tượng Country , Event ,… sẽ kế thừa |
| Country | create | Khởi tạo đối tượng Country |
| Event | create | Khởi tạo đối tượng Event |
| Location | create | Khởi tạo đối tượng Location |
| NguonGoc | create | Khởi tạo đối tượng NguonGoc |
| Person | create | Khởi tạo đối tượng Person |
| RelationShip | create | Khởi tạo đối tượng RelationShip |
| Organization | create | Khởi tạo đối tượng Organization |
| CreateRelationShip | createdata | Tạo RelationShip |
| Manager | createdata | Nạp dữ liệu cho các thực thể , quan hệ, nguồn gốc , thời gian |

### Các phương thức quan trọng trong từng lớp

#### .Class Insert

- phương thức : insertCountry(), insertPerson(), insertOrganization(), insertRelationShip(), insertLocation(), insertEvent(), có chức năng lưu các đối tượng vào Graphdb

- phương thức : insertRelationship(String filePath) : Tạo quan hệ theo mô hình dữ liệu RDF có đủ S-P-O . Lưu trữ trên GraphDB

- phương thức: getRepositoryConnection()

- phương thức : insert(RepositoryConnection repositoryConnection, String insert)

#### .Class Main

- phương thức:

* menuSimple(Query query): hiển thị các lựa chon truy vấn đơn giản trả về kết quả truy vấn và thời gian truy vấn
* menuAdvance(Query query): hiển thị các lựu chọn truy vấn phức tạp trả về kết quả truy vấn và thời gian truy vấn

#### Class Query

- phương thức:

* singleQuery(RepositoryConnection repositoryConnection, int i)
* advanceQuery(RepositoryConnection repositoryConnection, int i)
* thực hiện các câu truy vấn của hàm main

#### Các class trong package create

* Gồm các phương thức geter , seter và constructor

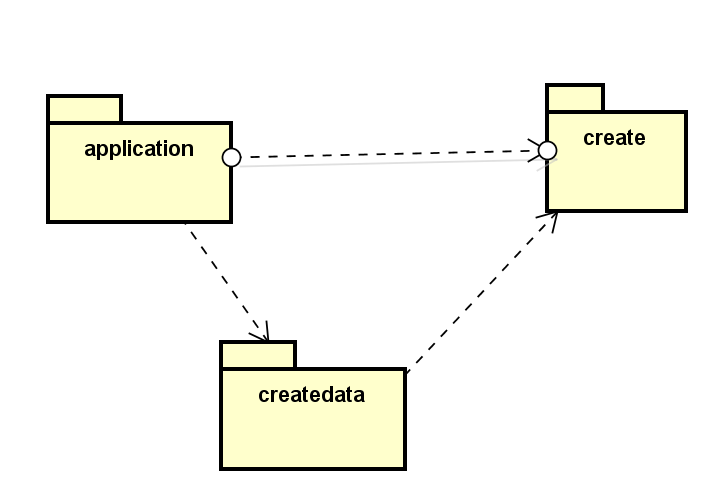
#### Class CreateRelationShip

* Phương thức:
* Rand(int a,int b): tạo số ngẫu nhiên trong khoảng a,b.

#### Class Manager

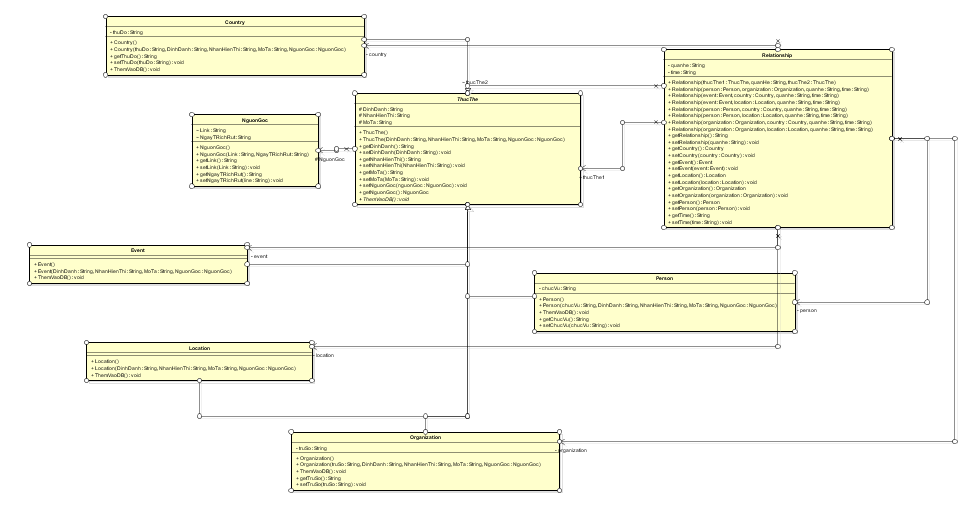
* Phương thức
* Rand(int a,int b): tạo số ngẫu nhiên trong khoảng a,b.
* Manager(String nguonGocFile ,String personFile, String countryFile, String quanHeFile, String organizationFile, String eventFile, String locationFile,String timeFile) : nap dữ liệu từ các file text lưu vào các ArrayList các thực thể

## Biểu đồ quan hệ giữa các package



## Biểu đồ quan hệ giữa các lớp

* Biểu đồ lớp thể hiện quan hệ giữa các thực thể



# PHẦN IV: KIỂM THỬ HIỆU NĂNG CỦA HỆ THỐNG

* Để đánh giá hiệu năng của hệ thống , ta đo thời gian thực hiện truy vấn cơ sở dữ liệu khi dữ liệu được lưu trữ với số lượng các quan hệ khác nhau trên Graphdb.

## Các câu truy vấn cơ bản

* Ý tưởng

Truy vẫn cơ bản : truy vấn thông tin trong một thực thể : DinhDanh, NhanHienThi,MoTa,… nội dung của các câu truy vấn sẽ được đặt trước, người sử dụng sẽ chọn ngẫu nhiên.

Gồm có các câu đơn giản :

* 1. Mô tả Vieng\_Chan(Location)
* 2. Mô tả Viet\_Nam(Location)
* 3. Chức vụ Chu\_Tich\_Quoc\_Hoi\_Nguyen\_Thi\_Kim\_Ngan(Person)
* 4. Mô tả Van\_Mieu\_Quoc\_Tu\_Giam(Location)
* 5. Thủ đô Campuchia(Country)
* 6. Trụ sở Apple(Organization)
* 7. Chức vụ Bo\_truong\_bo\_y\_te\_Nguyen\_Thi\_Kim\_Tien(Person)
* 8. Ngày trích rút Nhật Ban(Country)
* 9. Mô tả Paris(Country)
* 10. Mô tả AFF\_Cup(Event)

## Các câu truy vấn phức tạp

* Ý tưởng: Truy vấn mang tính chất thống kê , kết quả trả về sẽ là một list các thông tin , nội dung của các câu truy vấn sẽ được đặt trước, người sử dụng sẽ chọn ngẫu nhiên.

Gồm có các câu:

* Những sự kiện nào tổ chức tại campuchia?
* Chu\_Tich\_Quoc\_Hoi\_Nguyen\_Thi\_Kim\_Ngan sinh sống ở những đâu?
* Những ai đang sinh sống tại Hà Nội
* Nguyen VAn Khang có anh em với những ai?
* Những sự kiện nào diễn ra tại Việt Nam
* Pho\_Thu\_Tuong\_Truong\_Hoa\_Binh du lịch ở những đâu?
* Những ai làm việc tại Intel?
* Bo\_truong\_bo\_y\_te\_Nguyen\_Thi\_Kim\_Tien có quê hương ở những đâu?
* Những ai đi đến Đài Loan?
* Apple co trụ sở tại những đâu?

## Độ lớn dữ liệu sinh và kết quả thu được

Kết quả kiểm thử :

|  |  |  |
| --- | --- | --- |
| Độ lớn dữ liệu | Thời gian truy vân | |
|  | Truy vấn đơn giản | Truy vấn phức tạp |
| 2000 | 137ms | 540ms |
|  |  |  |
|  |  |  |
|  |  |  |

## Ưu nhược điểm của công nghệ sử dụng

* Ưu điểm
* trực quan dễ nhìn
* Hiệu suất cao , tốc độ truy vấn nhanh
* Linh động không cần xác định cấu trúc của mô hình
* Nhược điểm
* Không thể truy vấn nhiều đối tượng cùng một lúc
* Không quen thuộc khó sử dụng

# PHẦN IV: KẾT LUẬN

Chương trình của chúng em đã đáp ứng được một số yêu cầu của thầy nhưng còn có thiếu sót . Bài tập lớn giúp chúng em bổ xung nâng cao kiến thức . Áp dụng vào các chương trình lớn sau này. Em cảm ơn thầy đã tận tình giúp đỡ và chỉ bảo chúng em hoàn thành bài tập lớn.