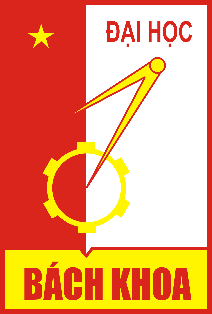
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG**

──────── \* ───────



BÁO CÁO MÔN HỌC

**LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG**

***Đề tài: Sinh giả lập, lưu trữ, truy vấn dữ liệu***

***sử dụng GraphDB***

**Giảng viên hướng dẫn: TS. Trịnh Tuấn Đạt**

**Lớp: 104403**

**Nhóm 9:**

|  |  |  |
| --- | --- | --- |
| **Họ và tên** | **MSSV** | **Đóng góp (%)** |
| **Nguyễn Việt Hùng** | **20161974** | **30** |
| **Lê Nam Anh** | **20160092** | **30** |
| **Ngọ Doãn Nghĩa** | **20162905** | **30** |
| **Nguyễn Văn Lâm** | **20152135** | **10** |
| **Nguyễn Văn Quý** | **20143715** | **0** |

**HÀ NỘI 12 – 2018**

# Phân chia công việc

|  |  |  |  |
| --- | --- | --- | --- |
| Họ và tên | MSSV | Công việc hoàn thành | Đóng góp (%) |
| Nguyễn Việt Hùng | 20161974 | Sinh giả lập các thực thể, truy vấn dữ liệu cơ bản & nâng cao, tìm hiểu cách giao tiếp bằng API. | 30 |
| Lê Nam Anh | 20160092 | Sinh giả lập các quan hệ, thêm dữ liệu đã sinh vào GraphDB, kiểm thử hiệu năng chương trình. | 30 |
| Ngọ Doãn Nghĩa | 20162905 | Sinh giả lập các thuộc tính thực thể, tạo câu truy vấn, tìm hiểu cú pháp SPARQL, tìm hiểu GraphDB. | 30 |
| Nguyễn Văn Lâm | 20152135 | Tạo các lớp thực thể, tìm hiểu GraphDB. | 10 |
| Nguyễn Văn Quý | 20143715 |  | 0 |

# Mục Lục

[Phân chia công việc 2](#_Toc533580636)

[Mục Lục 3](#_Toc533580637)

[Chương 1: Tìm hiểu về công nghệ 4](#_Toc533580638)

[1.1. Giới thiệu chung 4](#_Toc533580639)

[1.2. Các thuật ngữ 9](#_Toc533580640)

[1.3. Hướng dẫn cài đặt chi tiết 9](#_Toc533580641)

[1.4. Sử dụng graphdb 13](#_Toc533580646)

[1.5. Truy vấn dữ liệu 19](#_Toc533580647)

[1.6. Mô hình dữ liệu 21](#_Toc533580648)

[1.7. Cách giao tiếp bằng API 21](#_Toc533580649)

[Chương 2: Dữ liệu mô phỏng 25](#_Toc533580650)

[2.1. Mô hình dữ liệu 25](#_Toc533580651)

[2.2. Cấu trúc dữ liệu của chương trình: 25](#_Toc533580652)

[2.3. Giải thuật: 25](#_Toc533580653)

[Chương 3: Giới thiệu về chương trình 26](#_Toc533580654)

[3.1. Package 26](#_Toc533580655)

[3.2. Class 26](#_Toc533580656)

[3.3. Thuộc tính, phương thức 27](#_Toc533580657)

[3.4. Biểu đồ sự phụ thuộc giữa các package: 28](#_Toc533580658)

[3.5. Biểu đồ lớp 29](#_Toc533580659)

[Chương 4: Kiểm thử hiệu năng 33](#_Toc533580660)

[4.1. Truy vấn cơ bản 33](#_Toc533580661)

[4.1. Truy vấn nâng cao 35](#_Toc533580662)

[4.2. Kiểm thử hiệu năng 37](#_Toc533580663)

[Tài liệu tham khảo 40](#_Toc533580664)

# Chương 1: Tìm hiểu về công nghệ

## Giới thiệu chung

1. Giới thiệu về GraphDB

Onotext GraphDB là một [cơ sở dữ liệu đồ thị](http://graphdb.ontotext.com/documentation/free/glossary.html#glossary-sr) mạnh mẽ và hiệu quả cao với sự hỗ trợ RDF và [SPARQL](http://graphdb.ontotext.com/documentation/free/introduction-to-semantic-web.html#introduction-to-semantic-web-sparql).

GraphDB sử dụng RDF4J làm thư viện, tận dụng các API của nó để lưu trữ và truy vấn, cũng như hỗ trợ nhiều loại ngôn ngữ truy vấn (ví dụ: SPARQL và SeRQL) và cú pháp RDF (ví dụ: RDF/XML, N3, Turtle)

Sự phát triển của GraphDB được hỗ trợ một phần bởi SEKT, TAO, TripCom, LarKC và các dự án nghiên cứu châu Âu khác của FP6 và FP7.

Thông tin cấp phép đầy đủ có sẵn trong các tệp giấy phép nằm trong /docthư mục của gói phân phối.

GraphDB là một nhóm các cơ sở dữ liệu [RDF](http://graphdb.ontotext.com/documentation/free/introduction-to-semantic-web.html#introduction-to-semantic-web-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 kếtcũng 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.

GraphDB là một trong số ít ba công cụ có thể thực hiện [suy luận ngữ nghĩa](http://graphdb.ontotext.com/documentation/free/introduction-to-semantic-web.html#introduction-to-semantic-web-reasoning-strategies) ở quy mô cho phép người dùng rút ra các sự kiện ngữ nghĩa mới từ các sự kiện hiện có. Nó xử lý tải lớn, truy vấn và suy luận trong thời gian thực.

Onotext cung cấp ba phiên bản GraphDB: Miễn phí, Tiêu chuẩn và Doanh nghiệp.

* GraphDB Free - tối ưu hóa thương mại, dựa trên tệp, cùngAs & tối ưu hóa truy vấn, chia tỷ lệ thành 10 tỷ tỷ câu lệnh RDF trên một máy chủ với giới hạn hai truy vấn đồng thời;
* GraphDB Standard Edition (SE) - tối ưu hóa truy vấn, dựa trên tệp, cùng loại và truy vấn, tỷ lệ lên tới 10 tỷ tỷ câu lệnh RDF trên một máy chủ và không giới hạn số lượng truy vấn đồng thời.
* GraphDB Enterprise Edition (EE) - một cụm có tính sẵn sàng cao với triển khai cơ sở dữ liệu chính và công nhân để phục hồi và trả lời truy vấn song song hiệu năng cao.

1. Kiến trúc:

GraphDB được đóng gói như một lưu trữ và suy luận Layer (SAIL) cho [RDF4J](http://graphdb.ontotext.com/documentation/free/architecture-components.html#architecture-components-rdf4j) và được sử dụng rộng rãi trong những tính năng và cơ sở hạ tầng của RDF4J, đặc biệt là mô hình [RDF](http://graphdb.ontotext.com/documentation/free/introduction-to-semantic-web.html#introduction-to-semantic-web-rdf), phân tích cú pháp RDF và câu lệnh truy vấn.

[Suy luận](http://graphdb.ontotext.com/documentation/free/reasoning.html) được thực hiện bởi[Reasoner (TRREE Engine)](http://graphdb.ontotext.com/documentation/free/architecture-components.html#architecture-components-trree) trong đó các câu lệnh được suy luận, được lưu trữ trong các cấu trúc dữ liệu, được tối ưu hóa cao và giữ trong bộ nhớ để đánh giá truy vấn và suy luận thêm.

GraphDB triển khai giao diện [sail API](http://graphdb.ontotext.com/documentation/free/architecture-components.html#architecture-components-sail) 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 [Bàn làm](http://graphdb.ontotext.com/documentation/free/workbench.html) việc được nhúng hoặc Bàn làm việc RDF4J hoặc các công cụ khác được tích hợp với RDF4J.

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Khung RDF4J là một khung để lưu trữ, truy vấn và suy 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.

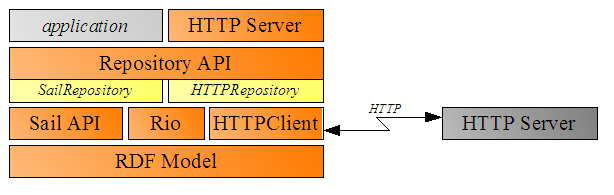
Về cơ bản có hai cách để sử dụng RDF4J:

* Như một máy chủ độc lập;
* Được nhúng trong một 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 HTTP RESTful. Các phần mở rộng khác nhau có sẵn hoặc đang được phát triển bởi các bên thứ ba.

**Kiến trúc RDF4J**

****

Khung RDF4J là một bộ các thành phần được ghép với nhau trong đó các triển khai thay thế có thể dễ dàng trao đổi.  RDF4J đi kèm với nhiều triển khai Lớp lưu trữ và suy luận (SAIL) mà người dùng có thể chọn cho hành vi mong muốn (trong bộ nhớ lưu trữ, hệ thống tệp, cơ sở dữ liệu quan hệ, v.v.). GraphDB là một thành phần SAIL plugin cho khung RDF4J.

Các ứng dụng thường sẽ giao tiếp với RDF4J thông qua Repository API. Điều này cung cấp một mức độ trừu tượng đủ cao để các chi tiết của các thành phần cơ bản cụ thể vẫn bị ẩn, tức là, các thành phần khác nhau có thể được hoán đổi mà không yêu cầu sửa đổi ứng dụng.

Repository APIcó một số triển khai, một trong số đó sử dụng HTTP để giao tiếp với kho lưu trữ từ xa hiển thị API kho lưu trữ thông qua HTTP.

**Sail API**

Các API Sail là một tập hợp các giao diện Java có hỗ trợ RDF lưu trữ, tìm kiếm, xóa và suy luận. 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. Đặc điểm chính của nó là:

* 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;
* xếp chồng SAIL;
* Kiểm soát đồng thời cho bất kỳ loại kho lưu trữ.

1. Thành phần
2. **Engine**

* **Query optimiser (Tối ưu hóa truy vấn)**

Trình tối ưu hóa truy vấn cố gắng xác định cách hiệu quả nhất để thực hiện một truy vấn nhất định bằng cách xem xét các [kế hoạch truy vấn](http://graphdb.ontotext.com/documentation/free/explain-plan.html) có thể . Khi các truy vấn được gửi và phân tích cú pháp, chúng sẽ được chuyển đến trình tối ưu hóa [truy vấn](http://graphdb.ontotext.com/documentation/free/data-loading-query-optimisations.html#data-loading-query-optimisations-query) nơi tối ưu hóa xảy ra. GraphDB cho phép [gợi ý](http://graphdb.ontotext.com/documentation/free/retain-bind-position-special-graph.html) hướng dẫn trình tối ưu hóa truy vấn.

* **TRREE Engine (Công cụ TRREE)**

GraphDB được triển khai trên đỉnh của công cụ TRREE. TRREE là viết tắt của 'Triple Reasoning and Rule Entailment Engine'. TRREE thực hiện lý luận dựa trên việc [xâu chuỗi](http://graphdb.ontotext.com/documentation/free/introduction-to-semantic-web.html#introduction-to-semantic-web-reasoning-strategies) các quy tắc yêu cầu đối với các mẫu ba RDF với các biến. Chiến lược lý luận của TRREE là [tổng thể hóa](http://graphdb.ontotext.com/documentation/free/introduction-to-semantic-web.html#introduction-to-semantic-total-materialisation) , mặc dù các [tối ưu hóa](http://graphdb.ontotext.com/documentation/free/inference-optimisations.html) khác nhau được sử dụng. Thông tin chi tiết của ngôn ngữ quy tắc có thể được tìm thấy trong phần [Reasoning](http://graphdb.ontotext.com/documentation/free/reasoning.html).

* **Lưu trữ**

GraphDB lưu trữ tất cả dữ liệu của nó trong các tệp trong thư mục lưu trữ được định cấu hình, thường được gọi là ['storage'](http://graphdb.ontotext.com/documentation/free/storage.html#storage-what-is-graphdb-s-persistence-strategy) . Nó bao gồm hai chỉ số chính trên các câu lệnh POS và PSO, [chỉ mục ngữ cảnh](http://graphdb.ontotext.com/documentation/free/storage.html#storage-context-indices) CPSO , [chỉ mục bằng chữ](http://graphdb.ontotext.com/documentation/free/storage.html#storage-literal-index) và bộ đệm trang .

* **Entity Pool**

Entity Pool là thành phần chính của lớp lưu trữ GraphDB. Nó chuyển đổi các thực thể ( [URI](http://graphdb.ontotext.com/documentation/free/introduction-to-semantic-web.html#introduction-to-semantic-web-uniform-resource-identifiers-uris) , nút trống và chữ) thành [ID nội bộ](http://graphdb.ontotext.com/documentation/free/configuring-a-repository.html#configuring-a-repository-configuration-parameters-entity-id-size) (số nguyên 32 hoặc 40 bit). Nó hỗ trợ hành vi giao dịch, giúp cải thiện việc sử dụng không gian và hành vi cụm.

1. **Connectors**

Trình kết nối cung cấp các tìm kiếm từ khóa và tổng hợp cực nhanh, thường được triển khai bởi một thành phần hoặc dịch vụ bên ngoài, nhưng có lợi ích bổ sung là luôn tự động cập nhật dữ liệu kho lưu trữ GraphDB. GraphDB đi kèm với trình kết nối [Lucene GraphDB](http://graphdb.ontotext.com/documentation/free/lucene-graphdb-connector.html)

Trình kết nối cung cấp đồng bộ hóa ở cấp thực thể, trong đó một thực thể được xác định là có một mã định danh duy nhất ( [URI](http://graphdb.ontotext.com/documentation/free/glossary.html#glossary-uri) ) và một tập hợp các thuộc tính và giá trị thuộc tính. Về mặt [RDF](http://graphdb.ontotext.com/documentation/free/glossary.html#glossary-rdf) , điều này tương ứng với một bộ ba có cùng chủ đề. Ngoài các thuộc tính đơn giản (được xác định bởi một bộ ba), chuỗi thuộc tính hỗ trợ Trình kết nối. Chuỗi thuộc tính được định nghĩa là một chuỗi ba lần trong đó mỗi đối tượng của bộ ba là chủ đề của bộ ba tiếp theo.

1. **Workbench**

Các [Workbench](http://graphdb.ontotext.com/documentation/free/workbench.html) là công cụ quản lý GraphDB dựa trên web. Giao diện người dùng tương tự như Ứng dụng Web RDF4J Workbench, nhưng có nhiều chức năng hơn.

Bàn làm việc GraphDB có thể được sử dụng cho:

* Quản lý kho lưu trữ GraphDB;
* Tải và xuất dữ liệu;
* Thực hiện các truy vấn và cập nhật SPARQL;
* Quản lý không gian tên;
* Quản lý bối cảnh;
* Xem / chỉnh sửa tài nguyên RDF;
* Giám sát truy vấn;
* Giám sát tài nguyên;
* Quản lý người dùng và quyền;
* Quản lý kết nối;
* Cung cấp API REST để tự động hóa các tác vụ khác nhau để quản lý và quản lý kho lưu trữ.

[Workbench](http://graphdb.ontotext.com/documentation/free/workbench.html) GraphDB được đóng gói dưới dạng một tệp .war riêng biệt trong bản phân phối GraphDB. Nó chỉ có thể được sử dụng như một bàn làm việc hoặc là bàn làm việc + máy chủ cơ sở dữ liệu.

## Các thuật ngữ

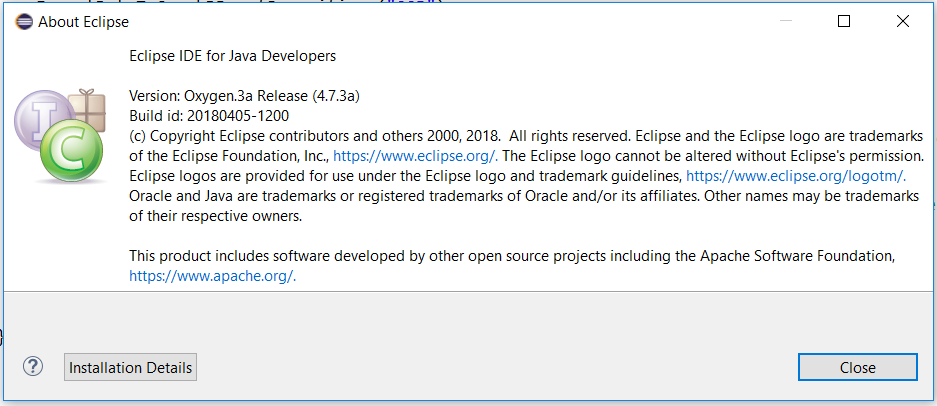
* RDF (Resource Description Framework): Đồ thị RDF
* Triple: Mỗi triple gồm 3 thành phần subject, predicate và object
* IRI (Internationalized Resource Identifier): là sự tổng quát của URI.
* Literals: là ***lexical form:*** một chuỗi Unicode hoặc là một **datatype IRI:** như một IRI
* Blank node: Node trống
* Graph: Đồ thị
* [SPARQL](http://graphdb.ontotext.com/documentation/free/introduction-to-semantic-web.html#introduction-to-semantic-web-sparql): Ngôn ngữ truy vấn
* Workbench: giao diện làm việc với GraphDB
* Connector: trình kết nối
* Repository: kho lưu trữ
* Domain: miền giá trị

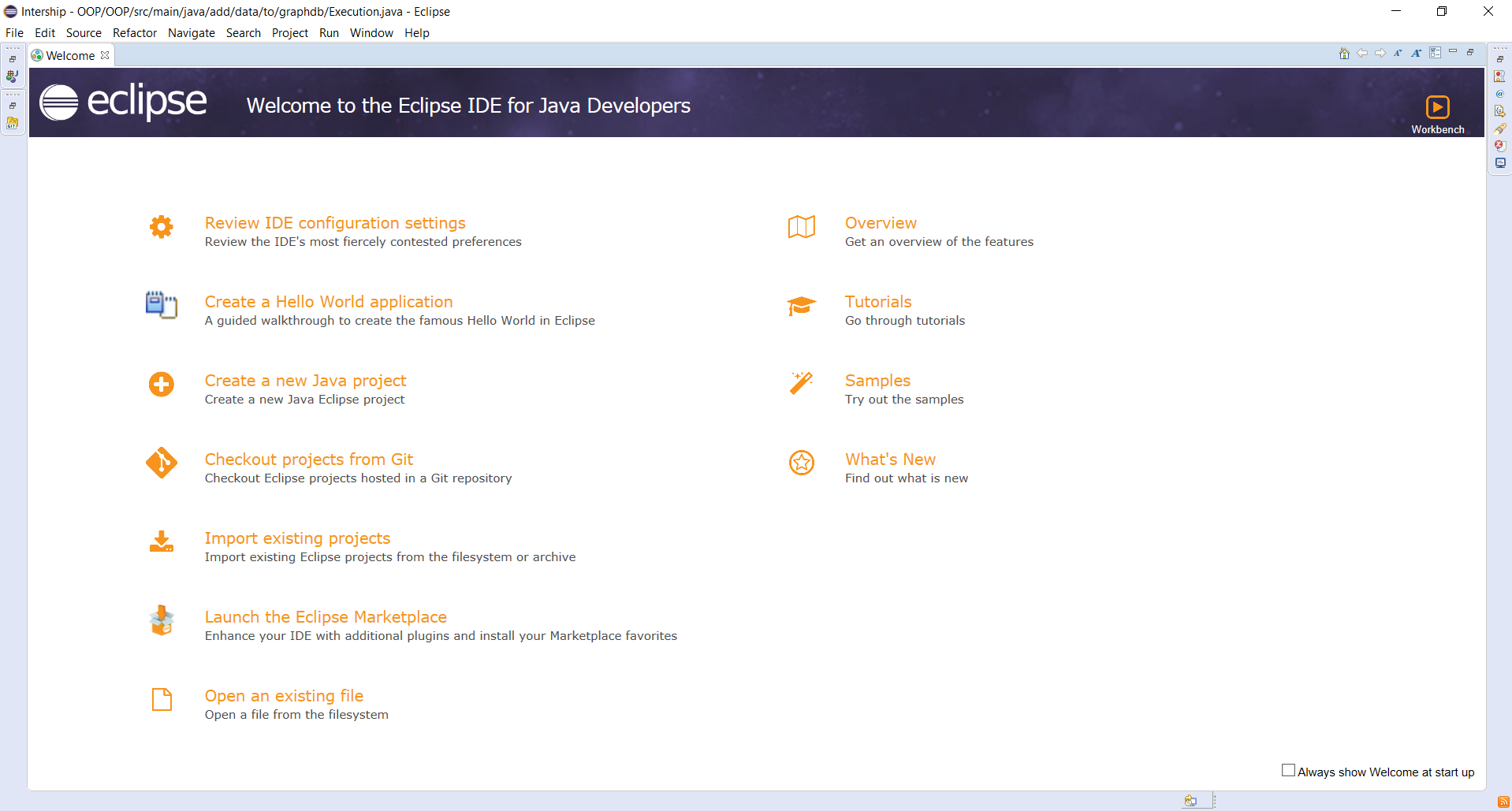
## Hướng dẫn cài đặt chi tiết

2. Cài đặt GraphDB trên Windows
3. Tải xuống tệp .exe GraphDB của bạn.
4. Nhấp đúp chuột vào tập tin ứng dụng và làm theo chỉ dẫn của trình cài đặt trên màn hình.
5. Định vị ứng dụng GraphDB trên menu bắt đầu của Windows và khởi động cơ sở dữ liệu. Máy chủ GraphDB và giao diện Workbend mở tại http://localhost:7200/.
6. Cài đặt GraphDB trên Mac OS
7. Tải xuống tệp .dmg GraphDB.
8. Nhấp đúp chuột vào nó và nhận được một đĩa ảo trên máy tính để bàn của bạn. Sao chép chương trình từ đĩa ảo vào thư mục ứng dụng đĩa cứng của bạn và bạn đã thiết lập.
9. Bắt đầu cơ sở dữ liệu bằng cách nhấp vào biểu tượng ứng dụng. Máy chủ GraphDB và Bàn làm việc mở tại <http://localhost:7200/>.
10. Cài đặt GraphDB trên Linux
11. Tải xuống GraphDB .rpm hoặc tệp .deb.
12. Cài đặt gói với hoặc và tên của gói đã tải xuống. Ngoài ra, bạn có thể nhấp đúp vào tên gói .sudo rpm-isudodpkg-i
13. Bắt đầu cơ sở dữ liệu bằng cách nhấp vào biểu tượng ứng dụng. Máy chủ GraphDB và Bàn làm việc mở tại <http://localhost:7200/>.
14. Cài đặt Maven Project để sử dụng API RDF4J

* **Cài đặt môi trường:**

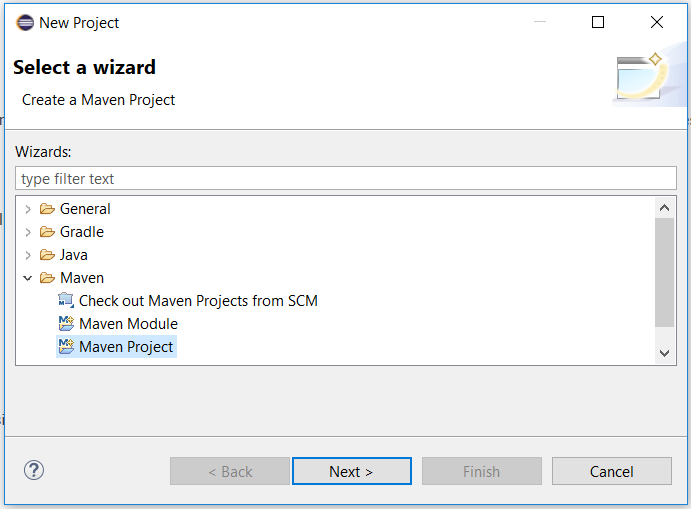
Trước khi bắt đầu, cần phải cài đặt Eclipse IDE. Trong phần trình bày, nhóm sử dụng Eclipse IDE for Java Developers version 4.7.3a.





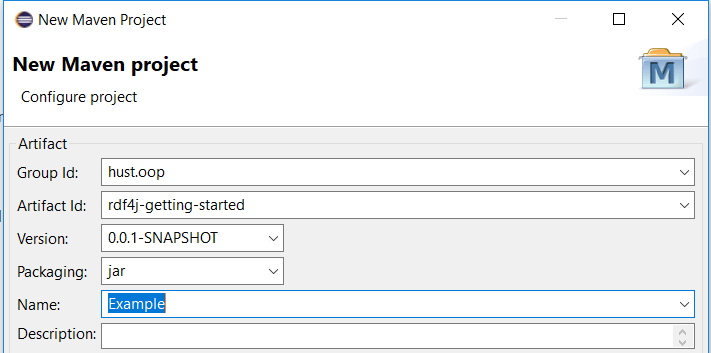
* **Tạo một Project mới:**

Để tạo một Project mới, chọn File -> New -> New Project. Một hộp thoại sẽ xuất hiện, trong hộp thoại này chọn Maven Project và nhấn Next.



Tiếp theo cần phải điền thêm thông tin về Maven Project vào các trường:

* Group id: tên Package, giống với tên của Package trong Java.
* Artifact id: tên của Maven Artifact.
* Name: tên Project

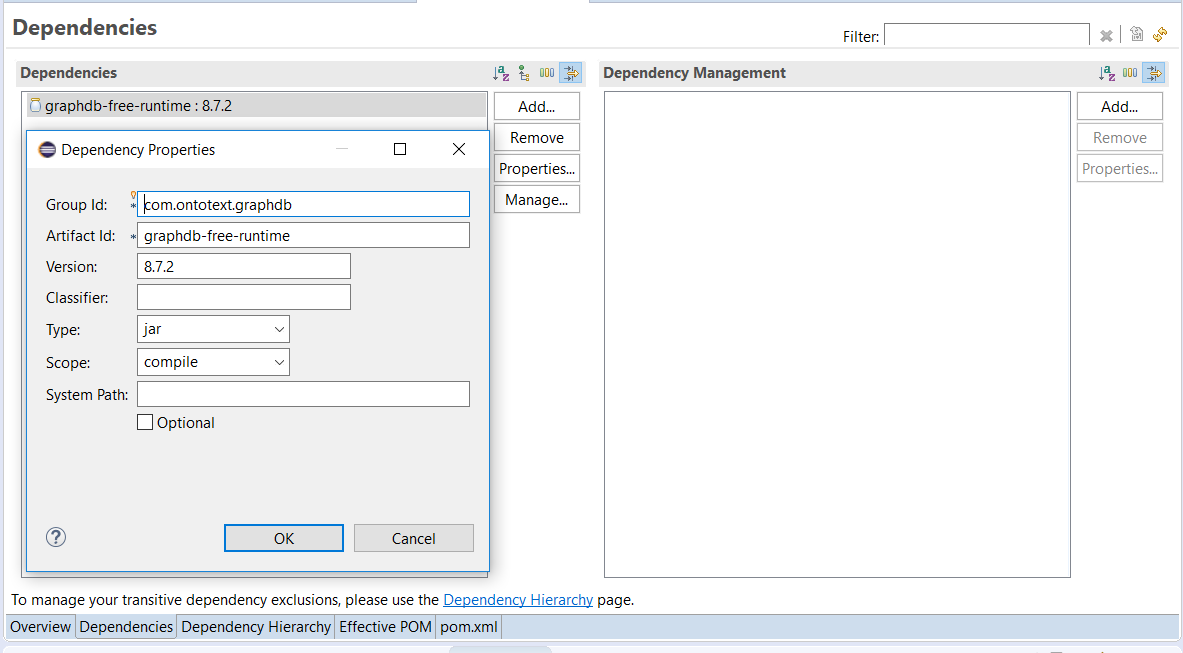


Click Finish để tạo một Maven Project mới.

* **Định nghĩa POM:**

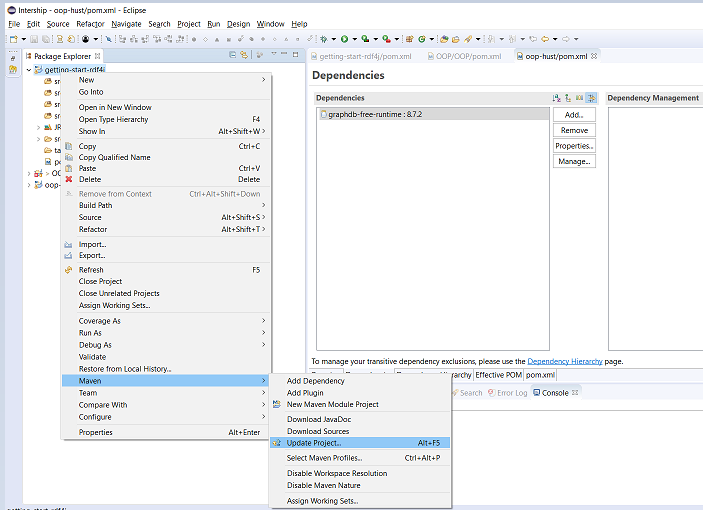
Để thêm API RDF4J vào Project, ta phải định nghĩa trong một file có tên là pom.xml, thường được gọi là “POM”.

Mở Project và click đúp vào file pom.xml, sau đó chuyển sang tab Dependencies và nhấn vào nút Add.



Sau đó điền Group id, Artifact id, Version của API muốn Import (như hình) vào và nhấn OK.

Cuối cùng, click chuột phải vào Project -> Maven -> Update Project để cập nhật những thay đổi.



Qua các bước trên, ta đã có một Project với các API cần thiết để thao tác với GraphDB.

## Sử dụng graphdb

1. Chạy graphdb:

* Tải xuống tệp phân phối GraphDB của bạn và giải nén nó.
* Khởi động giao diện GraphDB Server và Workbench bằng cách thực thi tập lệnh khởi động nằm trong thư mục $ graphdb\_home / bin: GraphDB
* Một thông báo xuất hiện trong bảng điều khiển của bạn cho bạn biết rằng GraphDB đã được bắt đầu ở chế độ bàn làm việc. Để truy cập Workbench, hãy mở http://localhost: 7200/ trong trình duyệt của bạn.

1. Cấu hình GraphDB:

* **Đường dẫn và cài đặt mạng:**

Cấu hình của tất cả các đường dẫn thư mục GraphDB và cài đặt mạng được đọc từ tệp conf/graphdb.properties. Nó kiểm soát nơi lưu trữ dữ liệu cơ sở dữ liệu, tệp nhật ký và dữ liệu nội bộ. Để gán giá trị mới, sửa đổi tệp hoặc ghi đè cài đặt bằng cách thêm -D<property>=<new-value>dưới dạng tham số vào tập lệnh khởi động. Ví dụ: để thay đổi số cổng cơ sở dữ liệu:

graphdb - Dgraphdb. connector. port = <your-port>

* **Cài đặt máy ảo Java**

Ta có thể kiểm soát kích thước heap bằng cách cung cấp một giá trị rõ ràng cho tập lệnh khởi động, chẳng hạn như hoặc đặt một trong các biến môi trường sau: graphdb -Xms10g -Xmx10g

* GDB\_HEAP\_SIZE biến môi trường để đặt cả kích thước heap tối thiểu và tối đa (được khuyến nghị).
* GDB\_MIN\_MEM biến môi trường để chỉ đặt kích thước heap tối thiểu.
* GDB\_MAX\_MEM biến môi trường để chỉ đặt kích thước heap tối đa.

1. Dùng cơ sở dữ liệu

Để dừng cơ sở dữ liệu, hãy tìm định danh quy trình GraphDB và gửi. Điều này sẽ gửi tín hiệu tắt máy và cơ sở dữ liệu dừng lại. Nếu cơ sở dữ liệu được chạy ở chế độ không phải daemon, bạn cũng có thể gửi ngắt **Ctrl + C** để dừng nó .kill <process-id>.

1. Một số thao tác trong WorkBench:

* **Tạo 1 kho:**

1. Chuyển đến **Setup -> Repositories**.
2. Nhấp vào **Create new repository**.

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

1. Nhập **myrepo** làm ID Kho lưu trữ và để lại tất cả các cài đặt cấu hình tùy chọn khác với các giá trị mặc định của chúng.

* **Tải dữ liệu thông qua GraphDB Workbench**

1. Chuyển đến **Import -> RDF**.
2. Mở tab *Dữ liệu người dùng* và nhấp vào *Tải lên tệp RDF* để tải lên các tệp từ tập dữ liệu mẫu **Tin tức** được cung cấp trong thư mục phân phối.

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

* **Tải dữ liệu thông qua SPARQL hoặc RDF4J API**

Cơ sở dữ liệu GraphDB cũng hỗ trợ API rất mạnh với điểm cuối SPARQL hoặc RDF4J tiêu chuẩn mà dữ liệu có thể được đăng với cURL, API máy khách Java cục bộ hoặc bảng điều khiển RDF4J. Nó phù hợp với tất cả các tiêu chuẩn. Nó cho phép mọi hoạt động cơ sở dữ liệu được thực thi thông qua yêu cầu máy khách HTTP.

1. Xác định vị trí điểm cuối URL GraphDB chính xác:
   * Chọn Setup -> Repositories
   * Nhấp vào biểu tượng liên kết bên cạnh tên kho lưu trữ
   * Sao chép URL kho lưu trữ.
2. Chuyển đến thư mục chứa tệp dữ liệu cục bộ của bạn.
3. Thực thi tập lệnh:

curl -X POST -H "Content-Type:application/x-turtle" -T localfilename.ttl

http://localhost:7200/repositories/repository-id/statements

* **Tạo đồ thị:**

Kiểm soát các truy vấn SPARQL đằng sau biểu đồ trực quan bằng cách tạo cấu hình biểu đồ trực quan của riêng bạn. Để tạo một cái, bấm vào Create graph config. Sử dụng các truy vấn mẫu để hướng dẫn bạn trong cấu hình.

Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Các phần sau của biểu đồ có thể được cấu hình:

* + - * Starting point - đây là trạng thái ban đầu của biểu đồ của bạn.
* Search box - bắt đầu với hộp tìm kiếm để chọn mỗi lần một tài nguyên bắt đầu khác nhau.
* Fixed node - bạn có thể muốn bắt đầu khám phá mỗi lần với cùng một tài nguyên.
* Query results - trạng thái cấu hình ban đầu có thể là biểu thị trực quan của kết quả truy vấn SPARQL
* Graph expansion - xác định cách các nút và liên kết mới được thêm vào biểu đồ trực quan khi người dùng mở rộng một nút hiện có. Biến ?node là bắt buộc và sẽ được thay thế bằng IRI của nút mở rộng.
* Node basics - truy vấn CHỌN này kiểm soát cách thức loại, nhãn, nhận xét và xếp hạng cho các nút trong biểu đồ. Các loại nút tương ứng với các màu khác nhau. Thứ hạng nút là một số từ 0 đến 1 và xác định kích thước của một nút. Nhãn là văn bản trên đầu của mỗi nút và nếu trống, tên địa phương IRI được sử dụng. Một lần nữa? Liên kết nút được thay thế bằng nút IRI.
* Predicate label - xác định văn bản nào sẽ hiển thị cho mỗi IRI cạnh. Truy vấn nên có biến ?edge để thay thế nó bằng IRI cạnh.
* Node extra - Nhấp vào biểu tượng thông tin để hiển thị các thuộc tính bổ sung cho một nút. Kiểm soát những gì nhìn thấy trong bảng điều khiển bên. Biến ?node được thay thế bằng IRI nút.

Lưu cấu hình của bạn và tải lại để khám phá dữ liệu của bạn theo cách bạn muốn xem.

* **Miền - Cấp đồ thị**

Để khám phá tính kết nối của một lớp nhất định, nhấp đúp vào vòng tròn lớp hoặc nút **Biểu đồ phạm vi miền** từ bảng điều khiển bên. Bạn có thể thấy một sơ đồ hiển thị lớp này và các thuộc tính của nó với miền và phạm vi của chúng, trong đó miền đề cập đến tất cả các tài nguyên và phạm vi chủ đề - cho tất cả các tài nguyên đối tượng. Ví dụ: nếu bạn bắt đầu từ lớp học pub: Company, bạn sẽ thấy một cái gì đó như:

**<pub-old: Mention pub-old: hasInstance pub: Company> <pub: Company pub: description xsd: string>**

Ảnh có chứa bản đồ, văn bản

Mô tả được tạo tự động

* **Mối quan hệ lớp**

Để khám phá mối quan hệ giữa các lớp, điều hướng đến Explore -> Class relationships. Bạn có thể thấy một sơ đồ phức tạp chỉ hiển thị các mối quan hệ hàng đầu, trong đó mỗi mối quan hệ là một bó các liên kết giữa các thể hiện riêng lẻ của hai lớp. Mỗi liên kết là một câu lệnh RDF trong đó chủ đề là một thể hiện của một lớp, đối tượng là một thể hiện của một lớp khác và liên kết là vị ngữ. Tùy thuộc vào số lượng liên kết giữa các thể hiện của hai lớp, gói có thể dày hơn hoặc mỏng hơn và có được màu của lớp có nhiều liên kết đến hơn. Các liên kết này có thể theo cả hai hướng.

Trong ví dụ dưới đây, bạn có thể thấy mối quan hệ giữa các lớp của bộ dữ liệu mẫu **Tin tức** được cung cấp trong thư mục phân phối. Bạn có thể quan sát rằng lớp có số lượng liên kết lớn nhất (gói dày nhất) là **pub-old: Document**.

Ảnh có chứa đồ họa véc-tơ

Mô tả được tạo tự động

## Truy vấn dữ liệu

* **Truy vấn dữ liệu trong** [**Workbench**](http://graphdb.ontotext.com/documentation/free/workbench.html)**:**

SPARQL là ngôn ngữ truy vấn giống như SQL cho cơ sở dữ liệu đồ thị RDF với các loại sau:

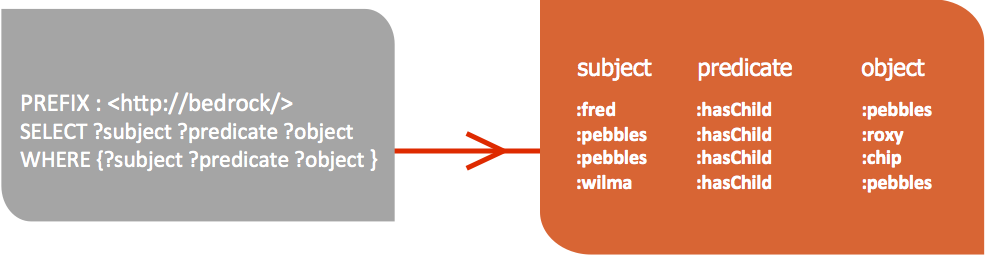
* **SELECT** - trả về kết quả dạng bảng;
* **CONSTRUCT** - tạo một biểu đồ RDF mới dựa trên kết quả truy vấn;
* **ASK** - trả về CÓ, nếu truy vấn có giải pháp, nếu không thì KHÔNG CÓ;
* **DESCRIBE**- trả về dữ liệu RDF về một tài nguyên; hữu ích khi bạn không biết cấu trúc dữ liệu RDF trong nguồn dữ liệu;
* **INSERT** - chèn ba lần vào một biểu đồ;
* **DELETE** - xóa ba lần từ một biểu đồ.
* **Một số câu lệnh truy vấn đơn giản:**
* **Insert:**

Ảnh có chứa văn bản

Mô tả được tạo tự động

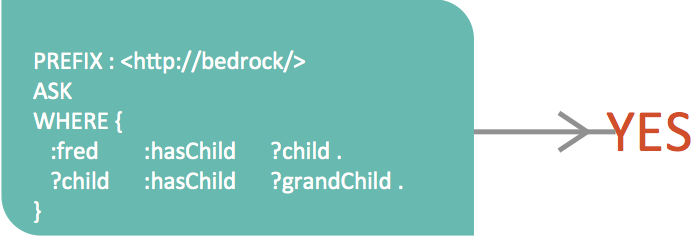
***Insert các thuộc tính và các quan hệ của thực thực thể vào database***

* **Selete:**



***Kết quả trả về 1 bảng gồm 3 trường subject predicate và object***

* ASK



***Kết quả trả về là Yes hoặc No***

* **Delete:**

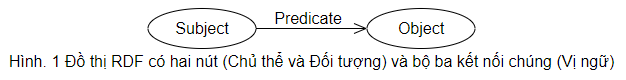
Ảnh có chứa ảnh chụp màn hình

Mô tả được tạo tự động

Câu lệnh sẽ xóa bản ghi trong đồ thị Graph có title là “A new book” và creator là “A. N. Other”.

## Mô hình dữ liệu

* Mô hình dữ liệu dựa trên đồ thị:
  + Cấu trúc cốt lõi của cú pháp là trừu tượng là 1 bộ ba, mỗi bộ gồm subject, predicate và object.
  + Một tập hợp bộ ba như vậy được gọi là đồ thị RDF, một đồ thị RDF có thể được hiển thị dưới dạng một nút và sơ đồ cung hướng, trong đó mỗi bộ ba được biểu diễn dưới dạng một liên kết nút-cung-nút.

****

Có ba loại nút trong đồ thị RDF: IRI, Literal và blank node

## Cách giao tiếp bằng API

Nhóm sử dụng RDF4J API để giao tiếp với GraphDB.

Repository API giúp dễ dàng thao tác với RDF Repository. Nó cung cấp nhiều phương thức để query và update dữ liệu. Interface cho Repository API nằm trong package *org.eclipse.rdf4j.repository.* Interface chính của Repository API là Repository Interface. Có ba implementations chính của interface này là:

* SailRepository là Repository sử dụng kih tạo và truy cập RDF4J repository ở local.
* HTTPRepository hoạt động như một kho lưu trữ trên máy chủ từ xa, có thể truy cập qua HTTP.
* SPARQLRepository hoạt động như một proxy cho bất kỳ SPARQL endpoint nào (cho dù endpoint đó được implement sử dụng RDF4J hay không).

Giới thiệu các API sử dụng:

* Interface Repository
* Thuộc package **org.eclipse.rdf4j.repository.**
* Một RDF4J repository chứa dữ liệu RDF có để được query hoặc update qua việc kết nối đến nó, phụ thuộc vào các implement repository để quyết định xem nó có được hỗ trợ nhiều kết nối đồng thời hay không.
* Các phương thức chính sử dụng:

|  |  |  |
| --- | --- | --- |
| Kiểu trả về | Tên phương thức | Mô tả phương thức |
| RepositoryConnection | getConnection() | Tạo kết nối đến repository để sử dụng cho việc query và update cho repository đó. **throws RepositoryException** nếu không thành công. |
| void | initialize() | Khởi tạo repository. Và nó **throws RepositoryException** nếu khởi tạo không thành công. |
| void | shutDown() | Đóng repository, giải phóng tất cả tài nguyên mà nó chiếm. **throws RepositoryException** nếu không thành công. |

* Interface RepositoryConnection
* Thuộc package **org.eclipse.rdf4j.repository.**
* Nó được **extends** từ **AutoCloseable,** là một Interface chính cho việc update và query dữ liệu. Mặc định thì nó ở chế độ auto-commit. Cần phải đóng nó sau khi sử dụng để giải phóng tài nguyên và tránh locks không cần thiết.
* Các phương thức chính sử dụng:

|  |  |  |
| --- | --- | --- |
| Kiểu trả về | Tên phương thức | Mô tả phương thức |
| void | begin() | Bắt đầu một transaction mới, cần commit() hoặc rollback() để kết thúc một transaction. |
| Update | prepareUpdate(QueryLanguage ql, String update) | Chuẩn bị cập nhật operation.  **Throws:**  **MalformedQueryException** nếu chuỗi update không đúng.  **RepositoryException** nếu không thành công, |
| void | commit() | Commits the active transaction. |
| TupleQuery | prepareTupleQuery(QueryLanguage ql, String query) | Chuẩn bị một query trả về tập giá trị tuples.  **Throws:**  **IllegalArgumentException** nếu chuỗi query không phải tuple query.  **MalformedQueryException** nếu chuỗi query không đúng.  **RepositoryException** nếu prepare query không thành công, |

* Class HTTPRepository
* Thuộc package **org.eclipse.rdf4j.repository.http.**
* HTTPRepository hoạt động như một kho lưu trữ trên máy chủ từ xa, có thể truy cập qua HTTP.
* Các phương thức sử dụng:

|  |  |  |
| --- | --- | --- |
| Kiểu trả về | Tên phương thức | Mô tả phương thức |
| RepositoryConnection | getConnection() | Tạo kết nối đến repository để sử dụng cho việc query và update cho repository đó. |

* Interface Value
* Thuộc package **org.eclipse.rdf4j.model.**
* Được extends từ interface **Serializable,** là supertype của tất cả các đối tượng mô hình RDF (URIs, Blank node và literals).
* Interface BindingSet
* Thuộc package **org.eclipse.rdf4j.query.**
* Được extends từ inerface **Iterable<Binding>** và **Serializable.**
* Là một tập các giá trị ràng buộc, thể hiện một kết quả truy vấn. Các giá trị được đánh chỉ mục theo tên của ràng buộc tương ứng với tên các biến được sử dụng trong phép chiếu của truy vấn ban đầu.

|  |  |  |
| --- | --- | --- |
| Kiểu trả về | Tên phương thức | Mô tả phương thức |
| Value | getValue(String bindingName) | Nhận giá trị của liên kết với tên được chỉ định từ BindingSet này. |

* Class QueryLanguage
* Thuộc package **org.eclipse.rdf4j.query.**
* Là bảng liệt cho các ngôn ngữ truy vấn RDF như SPARQL và SeRQL. Các đối tượng QueryL Language được xác định bằng tên của chúng, được xử lý không phân biệt chữ hoa chữ thường.
* Thuộc tính sử dụng:

|  |  |  |
| --- | --- | --- |
| Modifier and Type | Tên thuộc tính | Mô tả |
| static QueryLanguage | SPARQL | SPARQL (Simple Protocol and RDF Query Language) là ngôn ngữ để truy vấn và cập nhật dữ liệu RDF. |

* Interface TupleQuery
* Thuộc package **org.eclipse.rdf4j.query.**
* Được extends từ interface **Query**
* Phương thức sử dụng:

|  |  |  |
| --- | --- | --- |
| Kiểu trả về | Tên phương thức | Mô tả phương thức |
| TupleQueryResult | evaluate() | Lấy giá trị query.  **throws**: **QueryEvaluationException** nếu có lỗi. |

* Interface TupleQueryResult
* Thuộc package **org.eclipse.rdf4j.query.**
* Được extends từ **QueryResult<BindingSet>**
* TupleQueryResult biểu diễn kết quả truy vấn dưới dưới dạng một mảng các BindingSet. Cần phải đóng TupleQueryResult sau khi sử dụng để giải phóng tài nguyên mà nó chiếm.
* Interface Update
* Thuộc package **org.eclipse.rdf4j.query.**
* Được extends từ **Operation**, có nhiệm vụ động cập nhật trên một kho lưu trữ, xây dựng bằng một trong các ngôn ngữ truy vấn được hỗ trợ (ví dụ SPARQL).

|  |  |  |
| --- | --- | --- |
| Kiểu trả về | Tên phương thức | Mô tả phương thức |
| void | evaluate() | Thực hiên update repository.  **throws**: **UpdateExecutionException** nếu không thể thực hiện thành công. |

# Chương 2: Dữ liệu mô phỏng

## Mô hình dữ liệu

Dữ liệu được lưu dưới dạng các triple, mỗi triple sẽ lưu trữ thuộc tính hoặc quan hệ của thực hệ của thực thể.

Mỗi triple có 3 trường, subject, predicate và object:

Subject sẽ là một IRI hoặc node trống

Predicate sẽ là 1 IRI

Object sẽ là 1 IRI, node trống hoặc literal

## Cấu trúc dữ liệu của chương trình:

Dữ liệu gồm 5 thực thể Person, Organzation, Location, Event, Time.

Dữ liệu được lưu trữ dưới dạng các triple, mỗi triple sẽ lưu trữ thuộc tính hoạc quan hệ của thực thể

Các trường của subject, predicate, object được lưu dưới dạng các IRI

## Giải thuật:

1. *Sinh dữ liệu thực thể gốc (nhỏ):*
2. Với thuộc tính Định danh, Label và Mô tả của thực thể ta sẽ sinh bằng cách:

* Với thuộc tính Name của các thực thể, mỗi thực thể ta sẽ tạo ra một file txt với dữ liệu là tên của thực thể
* Với thuộc tính Label của các thực thể, mỗi thực thể ta sẽ tạo ra một file txt với dữ liệu là nhãn hiển thị của thực thể
* Với thuộc tính Mô tả của các thực thể, mỗi thực thể ta sẽ tạo ra một file txt với dữ liệu là Mô tả của thực thể

1. Với thuộc tính Ngày, Link trích rút, của thực thể ta sẽ sinh bằng cách:

* Tạo ra lớp sinh ngẫu nhiên thuộc tính Ngày và link trích rút.

1. *Sinh dữ liệu thực thể lớn:*

* Từ thực thể gốc ta sẽ gán các Label khác để tạo ra dữ liệu lớn.

1. *Sinh dữ liệu quan hệ:*
   * Với 2 thực thể, ta sẽ tạo ra 1 file txt chứa quan hệ của 2 thực thể đó.
   * Kết hợp danh sách các thực thể 1, thực thể 2 và các predicate để tạo quan hệ số lượng lớn.

# Chương 3: Giới thiệu về chương trình

## Package

Chương trình có các package sau:

* entity: dùng để khai báo các thực thể.
* create.property: dùng để xây dựng thuộc tính cho các thực thể bằng 2 cách: đọc từ file hoặc sinh ngẫu nhiên.
* create.entity: dùng để sinh các thực thể.
* create.relation: dùng để sinh các quan hệ.
* create.query.insert: tạo câu lệnh insert vào cơ sở dữ liệu, đồng thời cũng là phương pháp sinh dữ liệu số lượng lớn thông qua việc 1 câu lệnh insert sẽ có nhiều dòng (chuẩn là 10000 dòng, ứng với 10000 triple/1 câu lệnh insert).
* create.query.select: tạo câu lệnh select thực hiện truy vấn trên cơ sở dữ liệu.
* add.data.to.graphdb: thực thi các câu lệnh đã được tạo.

## Class

Mỗi package có các lớp sau:

* entity:
  + Entity: chứa các thuộc tính và phương thức cơ bản của mỗi thực thể để thực hiện kế thừa tránh lặp code.
  + Person/Organization/Country/Event/Location: khai báo các thực thể.
* create.property:
  + DataReader: đọc các thuộc tính từ file.
  + DateGenerator: sinh thuộc tính ngày tháng trích rút cho thực thể và sinh thực thể Time.
  + LinkGenerator: sinh thuộc tính link trích rút.
* create.entity:
  + Person/Organization/Country/Event/LocationGenerator: sinh các thực thể.
* create.relation:
  + RelationXY: (X, Y) thuộc tập sau (hiện có 8 phần tử, tức 8 loại quan hệ): (Country, Event), (Event, Location), (Event, Time), (Organization, Country), (Organization, Time), (Person, Country), (Person, Event), (Person, Location).
* create.query.insert:
  + InsertOriginalEntitiesQuery: tạo query dùng để nhập các thực thể gốc (có thông tin được nhập từ file txt viết tay, số lượng ít, xấp xỉ 1500).
  + InsertAdditionalPerson/Organization/Country/Event/LocationQueries: tạo câu lệnh dùng để sinh giả lập số lượng lớn các thực thể từ các thực thể gốc, bằng cách thay đổi label của mỗi thực thể thông qua việc ghép thêm 1 chỉ số (số tự nhiên 0,1,2...) vào cuối label. Số chỉ số bằng với số thực thể được sinh thêm từ 1 thực thể bất kỳ.
  + InsertRelationXY: (X, Y) thuộc tập như trên, dùng để tạo câu lệnh sinh giả lập số lượng lớn các quan hệ bằng cách chọn 1 thực thể, 1 hành động và 1 thực thể nữa, tổ hợp lại thành 1 triple.
  + SpaceToUnderScoreConverter: biến dấu cách trong xâu thành dấu gạch dưới “\_”, tạo điều kiện thực thi câu lệnh không bị lỗi.
* create.query.select:
  + BasicQueries: đọc các câu truy vấn đơn giản từ file.
  + ComplexQueries: đọc các câu truy vấn phức tạp từ file.
* add.data.to.graphdb
  + InsertDataToGraphDB: thực thi câu lệnh insert.
  + SelectDataFromGraphDB: thực thi câu truy vấn.
  + Execution: file thực thi.

## Thuộc tính, phương thức

* Entity/Person/Organization/Country/Event/Location: có các thuộc tính name, label, description, date, link để định hình thực thể. Person có thêm career.
* DataReader có phương thức đọc file theo 3 tiêu chí: tên file, dòng bắt đầu đọc, số dòng muốn đọc.
* DateGenerator và LinkGenerator có các phương thức giúp sinh ngẫu nhiên ngày và link.
* Person/Organization/Country/Event/LocationGenerator có phương thức tạo 1 ArrayList gồm các thực thể tương ứng và có thuộc tính count ghi lại số dòng (số thực thể) trong file txt.
* RelationXY có phương thức tạo ArrayList các đối tượng RelationXY và có các thuộc tính được khởi tạo giá trị trước là count (bằng số lượng hành động trong file txt) và các thuộc tính jumpS/P/O (số lượng dòng muốn đọc khi gọi phương thức DataReader trong phương thức tạo ArrayList của lớp này).
* Lớp InsertAdditionalPerson/Organization/Country/Event/LocationQueries có thuộc tính additionalIDCount đại diện cho số thực thể muốn tạo thêm từ 1 thực thể gốc và phương thức createQuery để viết câu truy vấn.
* Lớp InsertRelationXY cũng có phương thức createQuery để viết câu truy vấn.
* Lớp BasicQueries và ComplexQueries có phương thức đọc thông tin từ file.
* Lớp InsertDataToGraphDB chứa các phương thức giúp kết nối đến cơ sở dữ liệu dựa theo API, ứng với mỗi lớp có tên bắt đầu là Insert thì có 1 phương thức để thực thi câu lệnh Insert đó. Ngoài ra, lớp này còn có phương thức tạo giả lập thực thể hoặc quan hệ số lượng lớn tùy theo thông tin người dùng nhập vào bằng cách sử dụng phù hợp các phương thức thực thi các câu lệnh insert đã tạo.
* Lớp SelectDataFromGraphDB có phương thức thực thi câu truy vấn cũng như phương thức giúp hiển thị các câu truy vấn cho người dùng và thực hiện truy vấn theo yêu cầu.
* Lớp Execution chứa hàm main.

## Biểu đồ sự phụ thuộc giữa các package:

A picture containing screenshot, map, text

Description automatically generated

## Biểu đồ lớp

A picture containing map, text

Description automatically generated

***Hình 1. Biểu đồ các thực thể***

A close up of text on a white background

Description automatically generated

***Hình 2. Biểu đồ lớp thể hiện chức năng tạo số lượng lớn thực thể***

A screenshot of a cell phone

Description automatically generated

***Hình 3. Biểu đồ lớp thể hiện chức năng tạo số lượng lớn quan hệ***

A screenshot of text

Description automatically generated

***Hình 4. Biểu đồ lớp thể hiện chức năng truy vấn***

# Chương 4: Kiểm thử hiệu năng

## Truy vấn cơ bản

**prefix db: <http://dbpedia.org/resource/>**

**prefix foaf: <http://xmlns.com/foaf/0.1/>**

1. **Nghề của Vladimir\_Putin là gì?**

select ?career

where{

?person foaf:hasCareer ?career;

foaf:hasName db:Vladimir\_Putin.

}

1. **Lấy thông tin mô tả của nước Việt Nam?**

select ?description

where{

?country foaf:hasDescription ?description;

foaf:hasName db:Viet\_Nam.

}

1. **Đất nước nào tổ chức World Cup 2030?**

select ?name

where{

?country foaf:held ?event.

?event foaf:hasName db:World\_Cup\_2030.

?country foaf:hasName ?name.

}

1. **Công ty nào tổ chức 1 buổi triễn lãm sản phẩm vào ngày 5/7/2014?**

select ?name

where{

?organization foaf:held\_an\_exhibition\_of\_products\_in ?date;

foaf:hasName ?name.

filter(regex(str(?date), '5/7/2014'))

}

1. **Ai là thủ tướng nước Việt Nam?**

select ?name

where{

?person foaf:hasDescription db:Prime\_Minister\_of\_Viet\_Nam;

foaf:hasName ?name.

}

1. **Samsung tổ chức 1 buổi ra mắt sản phẩm mới tại đất nước nào?**

select ?name

where{

?organization foaf:held\_an\_introduction\_of\_new\_product\_in ?country;

foaf:hasName db:Samsung.

?country foaf:hasName ?name.

}

1. **Olympics 2040 được tổ chức tại nơi nào?**

select ?name

where{

?event foaf:is\_held\_in ?location;

foaf:hasName db:Olympics\_2040.

?location foaf:hasName ?name.

}

1. **Link trích rút của đoạn tin về Nguyễn\_Xuân\_Phúc?**

select ?link

where{

?person foaf:hasLink ?link;

foaf:hasName db:Nguyen\_Xuan\_Phuc.

}

1. **Ngày tháng trích rút đoạn tin về Facebook?**

select ?date

where{

?organization foaf:hasDate ?date;

foaf:hasName db:Facebook.

}

1. **Ai là người tổ chức cuộc thi Hoa hậu Hoàn vũ 2020?**

select ?name

where{

?person foaf:host ?event.

?event foaf:hasName db:Miss\_Universe\_2020.

?person foaf:hasName ?name.

}

## Truy vấn nâng cao

1. **Những sự kiện nào diễn ra tại Paris vào năm 2000?**

select ?name

where{

?event foaf:is\_held\_in ?location;

foaf:is\_held\_in ?date.

?location foaf:hasName db:Paris.

?event foaf:hasName ?name.

filter (regex(str(?date), '2000'))

}

1. **Có bao nhiêu cuộc thi đấu thể thao được tổ chức vào năm 2005?**

select (count(?name) as ?count)

where{

?event foaf:is\_held\_in ?date;

foaf:hasDescription ?description;

foaf:hasName ?name.

filter(regex(str(?date), '2005') && regex(str(?description), 'Sports\_Competition'))

}

1. **Liệt kê những ngày mà Microsoft tổ chức hội chợ thương mại tại Anh trong năm 2015?**

select ?time

where{

?organization foaf:held\_a\_tradefair\_in ?time;

foaf:held\_a\_tradefair\_in ?country.

?country foaf:hasName db:UK.

filter (regex(str(?time), '2015'))

}

1. **Những đất nước nào ở châu Á đã đăng cai tổ chức các kì World Cup trong khoảng từ năm 2040 đến 2050?**

select ?name

where{

?country foaf:held ?event;

foaf:hasDescription ?description;

foaf:hasName ?name.

?event foaf:hasName ?eventname.

filter (regex(str(?eventname), 'World\_Cup\_[2040-2050]') && regex(str(?description), 'Asia'))

}

1. **Liệt kê những vị tổng thống được sinh ra tại nơi tổ chức Euro 2040?**

select ?name

where{

?person foaf:hasName ?name;

foaf:hasCareer db:President;

foaf:was\_born\_in ?location.

?event foaf:is\_held\_in ?location;

foaf:hasName db:Euro\_2040.

}

1. **Những đất nước nào đã được ít nhất 1 công ty sản xuất ô tô tung ra sản phẩm mới vào nằm 2004? (chậm)**

select ?name

where{

?organization foaf:held\_an\_introduction\_of\_new\_product\_in ?date;

foaf:hasDescription ?description;

foaf:held\_an\_introduction\_of\_new\_product\_in ?country.

?country foaf:hasName ?name.

filter(regex(str(?date), '2004') && regex(str(?description), 'Car\_company'))

}

1. **Đếm số vị lãnh đạo trên thế giới đã gửi điện chúc mừng Việt Nam?**

select (count(?person) as ?count)

where{

?person foaf:telegraphed\_congratulation\_to ?country;

foaf:hasDescription ?description.

?country foaf:hasName db:Viet\_Nam.

filter(regex(str(?description), 'President') || regex(str(?description), 'Minister'))

}

1. **Liệt kê những địa điểm được chọn để tổ chức các cuộc thi Hoa hậu hoàn vũ từ năm 2030 đến 2040?**

select ?name

where{

?event foaf:is\_held\_in ?location;

foaf:hasName ?eventname.

?location foaf:hasName ?name.

filter (regex(str(?eventname), 'Miss\_Universe\_[2030-2040]'))

}

1. **Trong 2 quý đầu năm 2013, có những sự kiện nào được tổ chức ở London?**

select ?name

where{

?event foaf:is\_held\_in ?location;

foaf:is\_held\_in ?date;

foaf:hasName ?name.

?location foaf:hasName db:London.

filter(regex(str(?date), '[1-6]/2013'))

}

1. **Có những tổ chức nào tổ chức 1 buổi đấu giá các sản phẩm đặc biệt tại Mỹ trong tháng 2/2010?**

select ?name

where{

?organization foaf:held\_an\_auction\_of\_unique\_products\_in ?country;

foaf:held\_an\_auction\_of\_unique\_products\_in ?date;

foaf:hasName ?name.

?country foaf:hasName db:USA.

filter (regex(str(?date), '2/2010'))

}

## Kiểm thử hiệu năng

* Truy vấn cơ bản:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (N, M) | TVCB1 | TVCB2 | TVCB3 | TVCB4 | TVCB5 |
| 100, 200 | 10 ms | 11 ms | 14 ms | 16 ms | 10 ms |
| 5000, 7000 | 12 ms | 14 ms | 16 ms | 25 ms | 11 ms |
| 60K, 80K | 15 ms | 17 ms | 16 ms | 35 ms | 11 ms |
| 1M, 2M | 34 ms | 54 ms | 66 ms | 102 ms | 27 ms |
| 15M, 17M | 153 ms | 245 ms | 271 ms | 388 ms | 110 ms |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (N, M) | TVCB6 | TVCB7 | TVCB8 | TVCB9 | TVCB10 |
| 100, 200 | 14 ms | 13 ms | 13 ms | 14 ms | 12 ms |
| 5000, 7000 | 18 ms | 19 ms | 16 ms | 16 ms | 13 ms |
| 60K, 80K | 24 ms | 23 ms | 18 ms | 17 ms | 13 ms |
| 1M, 2M | 52 ms | 51 ms | 37 ms | 45 ms | 33 ms |
| 15M, 17M | 230 ms | 249 ms | 149 ms | 167 ms | 134 ms |

* Truy vấn nâng cao:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (N, M) | TVNC1 | TVNC2 | TVNC3 | TVNC4 | TVNC5 |
| 100, 200 | 15 ms | 16 ms | 12 ms | 11 ms | 11 ms |
| 5000, 7000 | 32 ms | 36 ms | 24 ms | 12 ms | 17 ms |
| 60K, 80K | 197 ms | 257 ms | 46 ms | 16 ms | 18 ms |
| 1M, 2M | 496 ms | 545 ms | 125 ms | 23 ms | 32 ms |
| 15M, 17M | 5193 ms | 6034 ms | 1104 ms | 151 ms | 292 ms |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| (N, M) | TVNC6 | TVNC7 | TVNC8 | TVNC9 | TVNC10 |
| 100, 200 | 23 ms | 11 ms | 16 ms | 19 ms | 15 ms |
| 5000, 7000 | 57 ms | 13 ms | 29 ms | 47 ms | 31 ms |
| 60K, 80K | 1078 ms | 19 ms | 79 ms | 169 ms | 69 ms |
| 1M, 2M | 32785 ms | 25 ms | 178 ms | 462 ms | 186 ms |
| 15M, 17M | 529355ms | 151 ms | 2160 ms | 5106 ms | 2044 ms |

* **Ưu nhược điểm của công nghệ:**
* Ưu điểm:
  + Dữ liệu được biểu diễn trực quan, dễ hiểu.
  + Truy vấn phức tạp khá nhanh.
  + Lưu trữ dữ liệu lớn dễ dàng.
* Nhược điểm:
  + Truy vấn khá phức tạp.
  + Khả năng tối ưu truy vấn kém.
  + Có thể mất nhiều triple để lưu một thông tin đơn giản.

# Tài liệu tham khảo

* API RDF4J:

<http://docs.rdf4j.org/javadoc/2.1/org/eclipse/rdf4j/>

* Sử dụng RDF4J:

[http://docs.rdf4j.org/programming/#\_creating\_a\_repository\_with\_rdf\_schema\_inferencing](http://docs.rdf4j.org/programming/%23_creating_a_repository_with_rdf_schema_inferencing)

* Tìm hiểu về GraphDB:

<http://graphdb.ontotext.com/documentation/free/>