

I. Semantic search란?

보다 적합한 결과를 얻기 위하여, 웹과 같은 dataspace에서, 탐색자의 의도와 용어의 문맥적 의미를 이해하여 탐색의 정확성을 높이기 위한 탐색을 말한다. Semantic search systems에서는 적합한 탐색결과를 제공하기 위하여, 다양한 방법 즉, context of search, location, intent, variation of words, synonyms, generalized and specialized queries, concept matching and natural language queries를 사용한다. Google과 Bing 같은 Major web search engines은 semantic search의 몇 가지 요소를 통합하여 사용하고 있다..

two major forms of search: navigational and research.

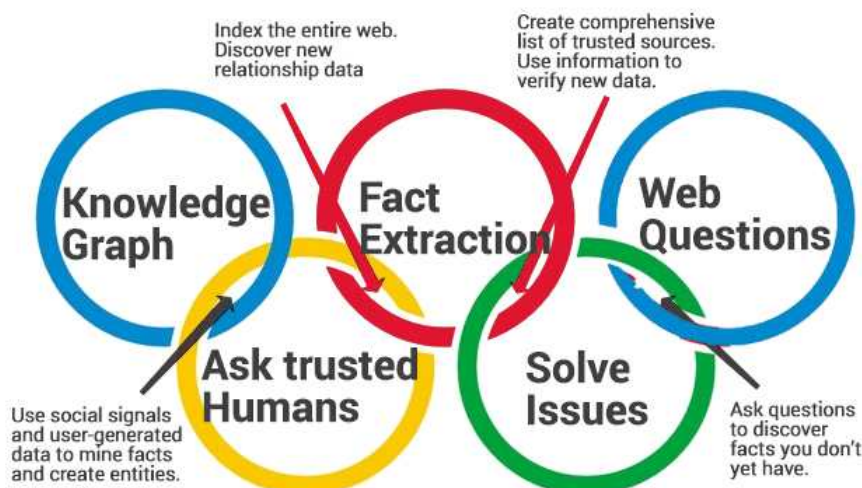
(1) navigational search,

the user is using the search engine as a navigation tool to navigate to a particular intended document. Semantic search is not applicable to navigational searches.

(2) research search,

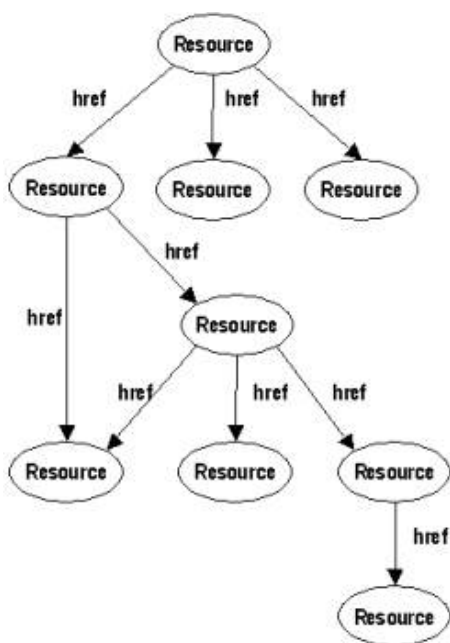
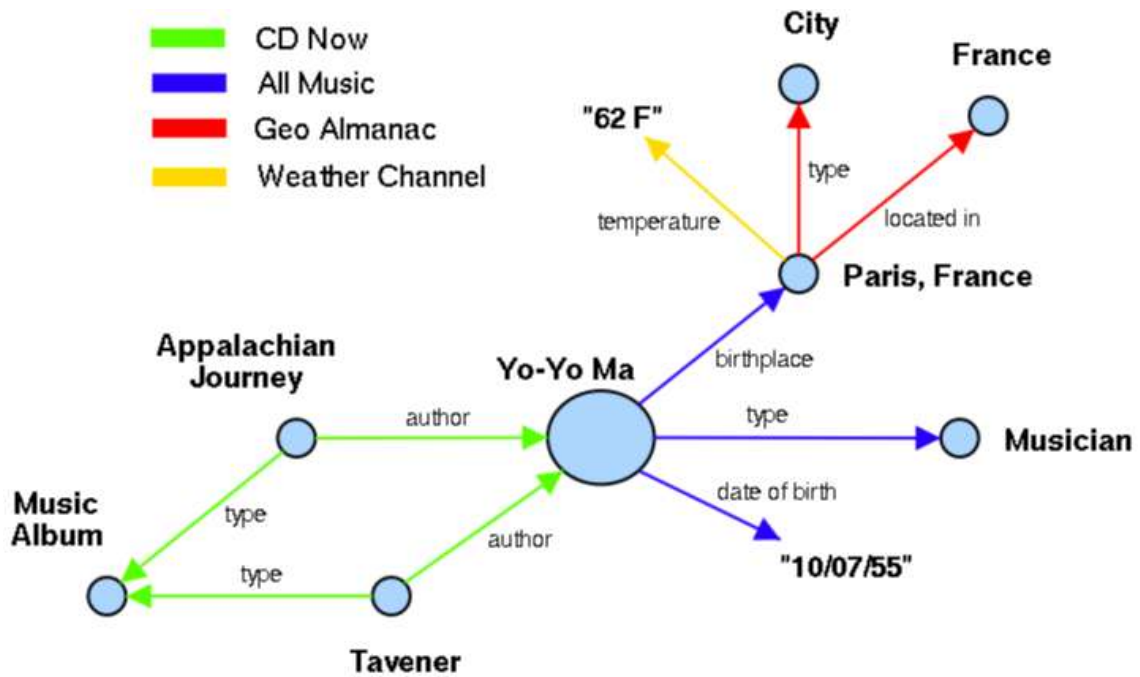
the user provides the search engine with a phrase which is intended to denote an object about which the user is trying to gather/research information. There is no particular document which the user knows about and is trying to get to. Rather, the user is trying to locate a number of documents which together will provide the desired information. Semantic search lends itself well with this approach that is closely related with exploratory search.

The 5 Steps of Google's Semantic Search

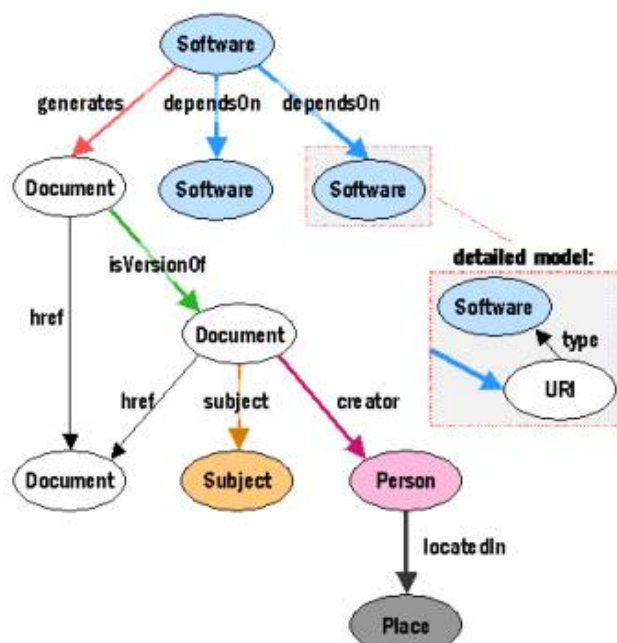


Each step leads to a more semantic web

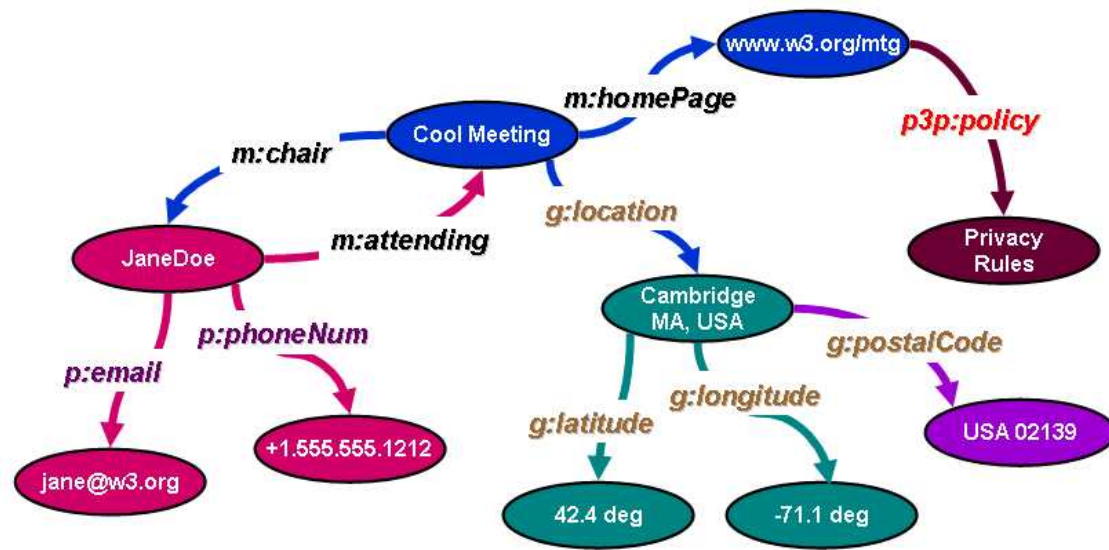
(C) davidamerland.com



a) Current Web



b) Semantic Web



<www.w3/org>

II. 온톨로지란?

온톨로지는 일종의 지식표현(knowledge representation)으로, 컴퓨터는 온톨로지로 표현된 개념을 이해하고 지식처리를 할 수 있게 된다. 프로그램과 인간이 지식을 공유하는데 도움을 주기 위한 온톨로지는, 정보시스템의 대상이 되는 자원의 개념을 명확하게 정의하고 상세하게 기술하여 보다 정확한 정보를 찾을 수 있도록 하는데 목적이 있다. 온톨로지는 시맨틱 웹을 구현할 수 있는 도구로서, 지식개념을 의미적으로 연결할수 있는 도구로서 RDF, OWL, SWRL 등의 언어를 이용해 표현한다. 온톨로지는 일단 합의된 지식을 나타내므로 어느 개인에게 국한되는 것이 아니라 그룹 구성원이 모두 동의하는 개념이다. 그리고 프로그램이 이해할 수 있어야 하므로 여러 가지 정형화가 존재한다.

정보시스템의 대상이 되는 분야에 존재하는 개체와 개념에 대한 명세로서, 사람과 컴퓨터간에 공유되는 지식을 개념화한 구체적인 형식이며, 개념화와 개념화간의 관계를 표현하는 것이다. 온톨로지는 단어와 관계들로 구성된 일종의 사전으로서 생각할 수 있으며, 그 속에는 특정 도메인에 관련된 단어들이 계층적으로 표현되어 있고, 추가적으로 이를 확장할 수 있는 연관관계가 포함되어 있어, 웹 기반의 지식 처리나 응용프로그램 사이의 지식 공유 등이 가능하도록 되어 있다. 즉, 시맨틱 웹의 목적인 자동적인 실행과 추론을 하기 위해 온톨로지는 가장 핵심적인 개념이라고 할 수 있다.

온톨로지의 구성 요소: 클래스(class), 인스턴스(instance), 관계(relation), 속성(property)

▶ 클래스(Class)

클래스는 일반적으로 우리가 사물이나 개념 등에 붙이는 이름을 말한다고 설명할 수 있다. "키보드", "모니터", "사랑"과 같은 것은 모두 클래스라고 할 수 있다.

▶ 인스턴스(Instance)

인스턴스는 사물이나 개념의 구체물이나 사건 등의 실질적인 형태로 나타난 그 자체를 의미한다. 즉, "LG전자 ST-500 울트라슬림 키보드", "삼성 싱크마스터 Wide LCD 모니터", "로미오와 줄리엣의 사랑"은 일반적으로 인스턴스라 볼 수 있다. 이와 같은 클래스와 인스턴스의 구분은 응용과 사용목적에 따라서 매우 달라질 수 있다. 즉, 같은 표현의 개체가 어떠한 경우에는 클래스가 되었다가 다른 경우에는 인스턴스가 될 수 있다.

▶ 속성(Property)

속성은 클래스나 인스턴스의 특정한 성질, 성향 등을 나타내기 위하여 클래스나 인스턴스를 특정한 값(value)와 연결시킨 것이다. 예를 들어, "삼성 싱크마스터 Wide LCD 모니터는 XX인치이다."라는 것을 표현하기 위하여, hasSize와 같은 속성을 정의할 수 있다.

▶ 관계(Relation)

관계는 클래스, 인스턴스 간에 존재하는 관계들을 칭하며, 일반적으로 taxonomic relation과 non-taxonomic relation으로 구분할 수 있다.

▶▶ Taxonomic Relation은 클래스, 인스턴스들의 개념분류를 위하여 보다 폭넓은 개념과 구체적인 개념들로 구분하여 계층적으로 표현하는 관계이다. 예를 들어, "사람은 동물이다"와 같은 개념간 포함관계를 나타내기 위한 "isA" 관계가 그것이다.

▶▶ Non-taxonomic relation은 Taxonomic Relation이 아닌 관계를 말한다. 예를 들어, "운동으로 인해 건강해진다"는 것은 "cause" 관계(인과관계)를 이용하여 표현한다.

III. 토픽 맵이란?

Topic Maps은 정보의 발견성을 높이기 위하여 지식의 표현과 교환을 위한 표준이며, 서로 다른 자원들에서 생산된 복수의 색인들을 통합하기 위하여 권말 색인 구조를 표현하기 위한 방법으로 1990년대 말에 개발되었다. The ISO standard is formally known as ISO/IEC 13250:2003.

토픽 맵은 collection of topics으로 구성되며, 각각의 토픽은 어떤 개념을 나타낸다. 토픽은 서로서로 associations에 의해 연관된다. 여기서 associations는 토픽들의 n-ary combination의 형태를 의미한다. 또한 토픽은 그 자신의 occurrences에 의해 몇 가지의

resources와 연관될 수도 있다.

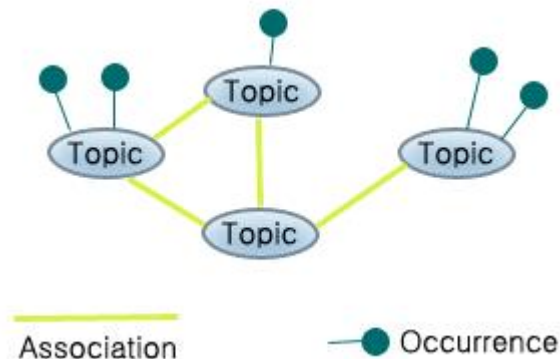


Figure 1. Topics, associations, and occurrences

Figure 1은 토픽 맵의 3가지 구성요소를 보여주고 있다. It also shows how the distinction between topic-to-topic and topic-to-resource relationships enables a partitioning of the model into a topic space that contains only topics and associations between topics and a resource space that contains the resources related to topics. This partitioning is interesting because it allows a topic map developed for one set of resources to be repurposed to index a different set of resources. In this way the topic map can be considered to be a portable form of knowledge.

Unlike domain-specific models, the topic map model has no predefined set of types. Instead, individual topic map authors or groups of authors in a community of practice can define the model for their domain of interest and share those models with other authors from other domains.

Topic Maps in Detail

1) Topics

A topic is a machine-processable representation of a concept. The Topic Maps standard does not restrict the set of concepts that can be represented as topics in any way.

2) Associations

Associations are the general form for the representation of relationships between topics in a topic map.

3) Occurrences

Occurrences are used to represent or refer to information about a concept represented by a topic.

4) Scope

Scope is the term used in the topic map standard to refer to a constraint or a context in which something is said about a topic.

Interchange and the XTM Syntax

ISO Topic Maps standard에서는 두 가지의 표준 교환 문법을 정의하고 있다: 하나는 SGML-based 이고 다른 하나는 XML-based 이다. A simple example of a topic map in XTM syntax is shown below.

```
<topicMap xmlns=http://www.topicmaps.org/xtm/1.0/ xmlns:xlink="http://www.w3.org/1999/xlink">
<topic id="band">
<baseName>
<baseNameString>Band</baseNameString>
</baseName>
</topic>

<topic id="person">
<baseName>
<baseNameString>Person</baseNameString>
</baseName>
</topic>

<!--Similarly for membership, group, singer and guitarist
-->

<!-- The Clash is a Band -->
<topic id="clash">
<instanceOf>
<topicRef xlink:href="#band"/>
</instanceOf>
<baseName>
<baseNameString>The Clash</baseNameString>
</baseName>
</topic>

<!-- Joe Strummer is a Person (note multiple names) -->
<topic id="joe-strummer">
<instanceOf>
<topicRef xlink:href="#person"/>
</instanceOf>
<baseName>
<scope>
<topicRef xlink:href="stage-name"/>
</scope>
```

```

<baseNameString>Joe Strummer</baseNameString>
</baseName>
<baseName>
<baseNameString>Joseph Mellor</baseNameString>
</baseName>
</topic>

<!--Joe Strummer is a member of The Clash -->
Note separate member elements used for the different roles played-->

<association>
<instanceOf>
<topicRef xlink:href="#membership"/>
</instanceOf>
<member>
<roleSpec>
<topicRef xlink:href="#group"/>
</roleSpec>

<topicRef xlink:href="#clash"/>
</member>
<member>
<roleSpec>
<topicRef xlink:href="#singer"/>
</roleSpec>

<topicRef xlink:href="#joe-strummer"/>
</member>
<member>
<roleSpec>

<topicRef xlink:href="#guitarist"/>
</roleSpec>
<topicRef xlink:href="#joe-strummer"/>
</member>
</association>
</topicMap>

```

IV. The Web Ontology Language (OWL)란?

이것은 온톨로지의 제작에 사용되는 지식표현언어의 한 종류이다. Ontologies는 taxonomies와 classification networks와 같은 다양한 도메인에서 지식의 구조를 기술하기 위한 공식적인 방법이다: the nouns representing classes of objects 그리고 the verbs representing relations between the objects.

Ontologies는 object-oriented programming에서 사용하는 class 계층구조와 닮았지만, 여러 가지 분명한 차이가 있다. Class hierarchies은 매우 느리게(전형적으로 월단위로) 진화한 소스 코드를 사용하여 구조를 표현하는 방법이지만, 온톨로지는 인터넷에서 끊임없이 진화하는 것으로 여겨지는 정보를 표현한다. Similarly, ontologies are typically far more flexible as they are meant to represent information on the Internet coming from all sorts of heterogeneous data sources. Class hierarchies on the other hand are meant to be fairly static and rely on far less diverse and more structured sources of data such as corporate databases.

OWL languages의 특징은 공식적으로 의미를 표현하기 위한 것이며, Resource Description Framework (RDF)라는 객체를 위한 W3C XML standard에 맞춰 만들어졌다 .

OWL and OWL2 are used to refer to the 2004 and 2009 specifications, respectively.

V. Linked Data And The Semantic Web

<<http://www.linkeddatatools.com/semantic-web-basics>>

Linked Data와 Semantic Web이란 무엇인가?

원칙적으로 Semantic Web은 Web 3.0 이다; 즉, 시스템들 또는 엔티티(entities) 끼리 서로 데이터를 링크하는 방법이다.

SW는 HTML 다큐먼트에 포함되어 있는 데이터를 사람이 읽는 것이 아니라, 컴퓨터가 읽을 수 있어야 한다는 사고의 전환에서 비롯되었다. 다시 말해서, 컴퓨터가 인간을 위해 좀 더 많은 사고적(thinking) 업무를 수행할 수 있어야 한다는 생각에서 개발되었다.

1. Graph Data

SW은 그렇게 잘 알려진 분야가 아니다. 초보자가 SW의 정의와 그것의 작동원리를 잘 이해하려 한다면, 먼저 그것의 데이터 저장방법 모델인 graph database에 대해 알아야 한다.

여러분이 전통적인 IT 분야에 지식을 갖고 있고, 계층형(for example XML) 또는 관계형 데이터베이스(for example MySQL, MS SQL)의 데이터저장방법에 대하여 알고 있겠지만, Resource Description Framework(RDF)에 대해선 잘 알지 못할 수도 있다.

RDF란 SW 커뮤니티에서 사용하는 일반적인 두문자어이며, 시멘틱 데이터의 웹을 제작하는데 필요한 기본적인 빌딩블록들 중의 한 요소이다. 또한 이것이 define(정의)하는 것은 여러분

이 익숙하지 않은 데이터베이스의 유형인 **graph database** 이다.

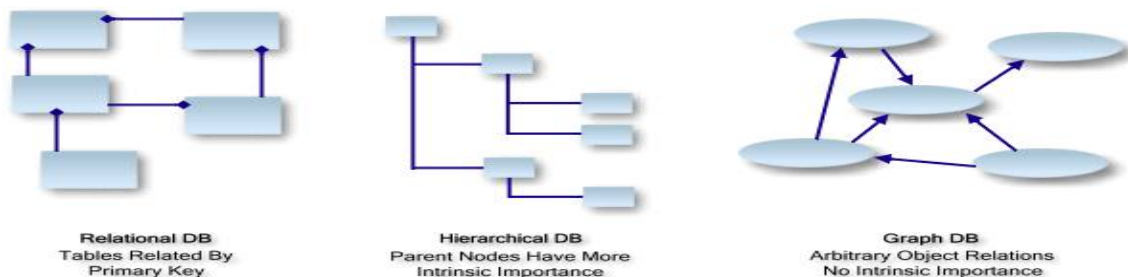
2) define이란?

- To state the precise meaning of (a word or sense of a word, for example).

비록 이런 데이터베이스에 대해 여러분이 생소하더라도, 이것이 바로 전 세계적으로 SW를 제작하는데 사용하는 데이터베이스의 한 종류 이다.

이제 graph database가 무엇이고, 어떻게 RDF가 그것을 정의하는지, 그리고 graph database를 시각화하는 방법이 무엇인지에 대하여 알아보자.

먼저 hierarchical, relational, 그리고 graph databases를 비교하여 이것들이 서로 어떻게 다른지를 살펴보기로 하자:



대부분의 데이터 저장 유형들에서는, 자신들의 여러 가지 데이터 요소(elements) 중에는 다른 것보다 보다 더 많은 precedence나 importance를 갖고 있는 몇 가지의 요소들(예를 들어, data nodes 또는 data tables)에 대한 개념을 정의하고 있다.

예를 들어, XML 다큐먼트를 보자. 전형적으로 XML 다큐먼트에는 한 개의 parent node와 함께 여러 개의 정보 노드를 사용하고 있다. 따라서 이러한 다큐먼트의 root는 최상위의 노드이며, 이 노드는 어떠한 부모 노드도 갖지 않는다.

위의 그림을 살펴보자. 맨 오른쪽의 data graph에는 어떠한 roots(또는 계층)도 존재하지 않으며, 서로 연결된 자원들로만 구성되어 있다. 또한 이 그래프의 어떠한 자원도 특별하게 다른 자원보다 본질적으로(intrinsic) 더 중요하지 않다는 것을 나타내고 있다.

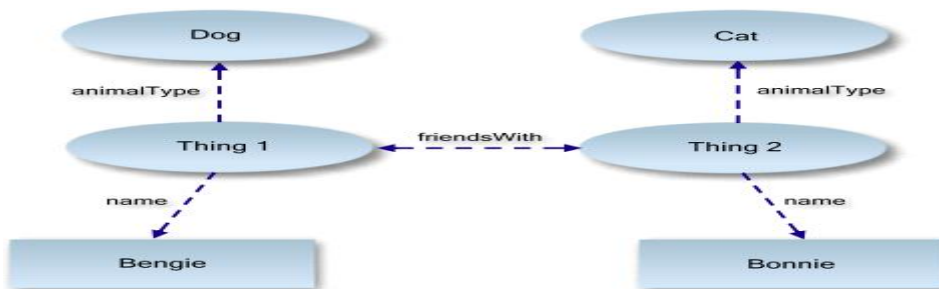
1.1 Example Of A Data Graph

사물들(things)이 서로 어떤 관계를 갖고 있는지를 기술하고 있는 지시문(statements)과 그 속에 있는 관계들을 RDF로 표현하는 방법을 알아보기에 앞서서, 먼저 그래프로 사물간의 관계를 시각화해 보자. 이것은 매우 쉽다.

먼저 아래의 개(벤지)와 고양이(보니) 간의 관계를 설명하는 지시문을 살펴보자:

- Bengie는 개다.
- Bonnie는 양이다.
- Bengie와 Bonnie는 친구이다.

위의 간단한 3개의 지시문을 데이터 그래프로 그려보면, 아래와 같다:



위의 그래프에 나타나 있는 관계를 살펴보면, 타원으로 표시된 "Thing 1" 과 "Thing 2" 라는 두 개의 사물과 사물의 속성이름이 하나는 "animalType"이고 또 하나는 "friendsWith"라는 것을 알 수 있다.

또한, "Thing 1"의 속성 이름의 값은 사각형으로 표시된 "Bengie"이고, "Thing 2"의 속성 이름의 값이 "Bonnie"이며, "Thing 1"은 개를, "Thing 2"는 고양이를 의미한다는 것도 알 수 있다. 끝으로, 이 두 개의 사물은 서로 친구(속성 "friendsWith"가 화살표로 양 방향 모두를 가리키고 있음)라는 것도 알 수 있다.

핵심 포인트 - 위의 그래프에서 화살표는 속성(properties)을 나타낸다: 때론, 이것을 RDF 용어(terminology)로는 predicates라 한다.

1.2 Graph Model 지시문의 예

`http://dbpedia.org/resource/Billie_Jean` has a **singer** whose value is **Michael Jackson**

- ▶ Subject: `http://dbpedia.org/resource/Billie_Jean` (URI)
- ▶ Predicate: `http://www.example.com/terms/singer` (URI)
- ▶ Object: `Michael_Jackson` (Literal)

이제 RDF에 대해 알아보기 전에, 다음과 같은 간단한 예를 먼저 맛을 보자:

1.3 Example Of RDF

```

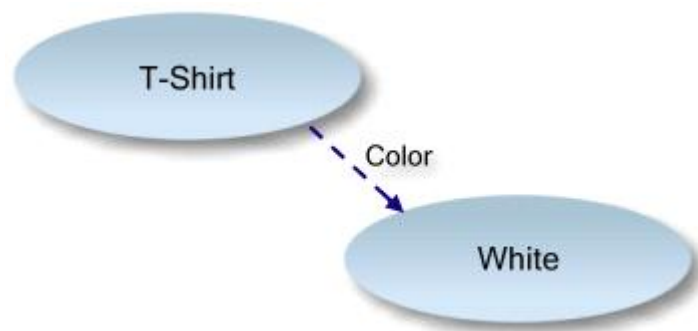
01. <?xml version="1.0" encoding="UTF-8"?>
02.
03. <rdf:RDF
04.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
05.   xmlns:dc="http://purl.org/dc/elements/1.1/"
06.   xmlns:region="http://www.country-regions.fake/">
07.
08.   <rdf:Description rdf:about="http://en.wikipedia.org/wiki/Oxford">
09.     <dc:title>Oxford</dc:title>
10.     <dc:coverage>Oxfordshire</dc:coverage>
11.     <dc:publisher>Wikipedia</dc:publisher>
12.     <region:population>10000</region:population>
13.     <region:principaltown rdf:resource="http://www.country-regions.fake/oxford"/>
14.   </rdf:Description>
15.
16. </rdf:RDF>

```

지금 위의 내용에 대해 걱정하지 마라. 우리는 나중에 이것을 다시 검토할 것이다. 지금은 단지 위와 같은 것이 RDF/XML - the XML form of RDF라는 것만을 이해하면 된다. RDF를 레코딩하는 여러 가지 방법들이 있지만, 우리는 단지 RDF/XML만을 살펴볼 것이다.

위의 콘텐츠에서 붉은 색으로 표시한 RDF/XML(<rdf:Description> tags 사이)를 RDF 지시문, 또는 때때로 RDF triple이라 부른다. 이 두 가지 이름 중에서 triple이 우리가 RDF를 이해하는데 가장 도움이 되는 용어인데, 그 이유는 이것이 지시문을 3가지의 요소로 지시문이 구성되어 있다는 것을 의미하기 때문이다: 즉, 지시문 구성요소인 주체(subject), 속성(predicate), 그리고 객체(object).

그래프의 형태로 이 3가지의 구성 용어를 사용하여, 티-셔츠의 색깔을 표현하고 있는 데이터 그래프의 예는 다음과 같다:



■ 주체: T-shirt; ■ 속성: color; ■ 객체: white.

핵심 포인트 - RDF는 SW의 데이터 구조를 정의하는 토대이지만, 데이터에 숨어있는 어의나 의미를 스스로 기술하지는 못한다. 이것은 나중에 RDFS (RDF Schema)와 OWL (Web Ontology Language)에서 다루기로 한다.

이번에는 간단한 RDF/XML 지시문을 통해, 이 구성요소들에 대하여 알아보자:

```
01. <?xml version="1.0" encoding="UTF-8"?>
02.
03. <rdf:RDF
04.     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
05.     xmlns:feature="http://www.linkeddatatools.com/clothing-features#">
06.
07.     <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">
08.
09.         <feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>
10.
11.     </rdf:Description>
12.
13. </rdf:RDF>
```

이 콘텐츠에서 여러분은 먼저 위의 RDF에서 주체, 속성, 객체가 어떻게 정의되고 기술되는지에 대해 감을 잡아야 한다.

2. Introducing RDF/XML

graph data model이 SW에서 데이터를 저장하는 모델이라면, RDF는 그것을 만드는 포맷이다. 이미 여러분은 RDF에서 주체, 속성, 그리고 객체로 표현되는 지시문의 정의방법에 대해 알았으며, subject->predicate->object relationship를 triple이라 부른다는 것도 알았고, RDF가 시멘틱 데이터의 웹을 구축하는 기본 포맷이라는 것도 알았다.

이제 간단한 예를 가지고 RDF 지시문에 대해 한 단계씩 알아보기로 한다.

2.1 Building An RDF document

>Add The RDF document Root Tag

먼저, RDF root node를 다음과 같이 추가해 보자:

```
1. <rdf:RDF
2.     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
3.
4.     <!-- Body Code Omitted -->
5.
6. </rdf:RDF>
```

위의 예에 있는 두 번째 줄에서, 여러분은 표준화된 W3.org namespace인

<http://www.w3.org/1999/02/22-rdf-syntax-ns#>

라는 URI를 보게 될 것이다. 이 namespace는 다른 컴퓨터 리더(reader)에게 이 태그로 봉해져 있는(enclosing) 다큐먼트가 RDF 다큐먼트라는 것과 여기서 사용된 rdf:RDF tag들이 이 namespace에 포함되어 있다는 것을 알려주는 것이다.

1) 컴퓨팅에서 namespace란 여러 종류의 사물을 조직하기 위하여 사용되는 심볼의 세트를 말한다. 따라서 이러한 사물들은 이름으로 참조(referred to)될 수 있다.

Namespaces는 일반적으로 다른 환경에서도 그 이름을 재사용할 수 있도록 계층구조를 갖는다. 하나의 추론으로, 각각의 사람은 올바른 이름뿐만 아니라 자신들의 친척과 공유하고 있는 가족이름을 갖고 있는 인명 시스템을 생각해 보자. 만일, 각 가족에서 가족 이름의 고유한 것이라면, 각각의 사람은 이름과 가족이름의 결합에 의해 유일하게 식별될 수 있다. 다시 말해서, 비록 많은 Janes이 있더라도 Jane Doe는 단지 한명 뿐이다; Doe라는 성의 namespaces에서, 단지 "Jane"은 이 사람을 확실하게 설명하는데 충분하지만, 모든 사람의 "global" namespace에서는 full name이 사용되어야만 한다.

위의 다큐먼트에서 namespace를 표현하고 있는 RDF node가 바로 이 RDF 다큐먼트의 roots node이다.

>Add A Statement

RDF 다큐먼트에는 하나 이상의 지시문이 포함될 수 있다. 간단하게 우리는 하나를 추가할 것이다. RDF/XML에서 주체(subject)를 정의하는 방법은 `<rdf:Description>` tag를 사용하는 것이다. 다음과 같이 붉은 색 지시문을 추가해 보자:

```
01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.
04.   <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">
05.
06.       <!-- Statement Code Omitted -->
07.
08.   </rdf:Description>
09.
10. </rdf:RDF>
```

위에서 붉은 색으로 표현된 `<rdf:Description>` tag의 의미는 간단하게 말해서 다음과 같다: "나는 객체 t-shirt에 대하여(about) 정의(describe)하며, 그것의 유일한 ID는 <http://www.linkeddatatools.com/clothes#t-shirt> 이다."

하나 더 주목!!! `<rdf:Description>`은 about 속성에 의해 지정된 resource에 대한 URI 정보를 갖고 있는 container 이다. 다음의 예에서 각각의 resources는 books이고, 그것의 식별자(URI)는 서명이다. 그리고 각각의 책에는 한명의 저자와 페이지 수만 표현되어 있다.

Book Source Example

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:lib="http://www.zvon.org/library">

  <rdf:Description about="Matilda">
    <lib:creator>Roald Dahl</lib:creator>
    <lib:pages>240</lib:pages>
  </rdf:Description>
```

!!! 위의 <rdf:Description />의 또 다른 표기법:

```
<rdf:Description about="Matilda" lib:creator = "Roald Dahl" lib:pages="240" />

<rdf:Description about="The BFG">
  <lib:creator>Roald Dahl</lib:creator>
  <lib:pages>208</lib:pages>
</rdf:Description>

<rdf:Description about="Heart of Darkness">
  <lib:creator>Joseph Conrad</lib:creator>
  <lib:pages>110</lib:pages>
</rdf:Description>

<rdf:Description about="Lord Jim">
  <lib:creator>Joseph Conrad</lib:creator>
  <lib:pages>314</lib:pages>
</rdf:Description>

<rdf:Description about="The Secret Agent">
  <lib:creator>Joseph Conrad</lib:creator>
  <lib:pages>249</lib:pages>
</rdf:Description>
</rdf:RDF>
```

위의 다큐멘트의 결과는 다음과 같다:

Author	Title	Pages
Roald Dahl	Matilda	240
Roald Dahl	The BFG	208
Joseph Conrad	Heart of Darkness	

110
Joseph Conrad
Lord Jim
314
Joseph Conrad
The Secret Agent
249

좀 이해가 됐나요? 계속 배워봅시다.

>Add Predicates

여러분이 주체에 대해 정의하면서 그것의 고유한 ID를 지정해 놓았지만, 그 주체의 특성에 대해서는 아무 것도 정의하지 않았다. RDF 지시문에서는 속성(predicates)을 사용하여 해당 주체의 특성들을 정의한다.

간단하게, T-shirt의 속성 중 하나인 size를 다음과 같이 추가해 보자:

```
01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:feature="http://www.linkeddatatools.com/clothing-features#">
04.
05.   <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">
06.
07.       <feature:size>12</feature:size>
08.
09.   </rdf:Description>
10.
11. </rdf:RDF>
```

위의 7번째 줄을 보자. 간단하게 말해서 이 주체는 <feature:size> 태그이름으로 기술된 한 개의 속성을 갖고 있으며, 이 속성의 문자 값은 12이다. RDF 전문용어로 이것이 바로 지시문(statement)이다.

그리고 3번째 줄에 이 속성이름뿐만 아니라 이 객체에서 사용되는 속성이름의 namespace에 대한 URI가 표시되어 있다는 것을 확인하라.

끝으로, T-shirt의 색깔인 color 속성을 하나 더 다음과 같이 추가해 보자:

```
01. <rdf:RDF
02.     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.     xmlns:feature="http://www.linkeddatatools.com/clothing-features#">
04.
05.     <rdf:Description rdf:about="http://www.linkeddatatools.com/clothes#t-shirt">
06.
07.         <feature:size>12</feature:size>
08.         <feature:color rdf:resource="http://www.linkeddatatools.com/colors#white"/>
09.
10.     </rdf:Description>
11.
12. </rdf:RDF>
```

이 <feature:color> 속성의 표현은 <feature:size>와 다르다는 것을 알았을 것이다. 지난 번 속성에서는 문자 값 12가 있었던 반면에, 이번 것에는 또 다른 지시문의 subject(ID)를 참조하도록 지정해 놓았다. 이러한 표현이 옳은 것이다. RDF에서 객체는 다른 지시문에 있는 주체를 참조할 수 있다.

!!! 속성 `rdf:resource`는 다른 <rdf:Description> element에서 정의하고 있는 resource에 about한 정보를 입력하는데 사용될 수 있다. 다음의 Book Sample을 살펴보자:

Book Sample

```
<rdf:Description about="RD">
<lib:firstName>Roald</lib:firstName>
<lib:surname>Dahl</lib:surname>
</rdf:Description>

<rdf:Description about="JC">
<lib:firstName>Joseph</lib:firstName>
<lib:surname>Conrad</lib:surname>
</rdf:Description>

<rdf:Description about="Matilda">
<lib:creator rdf:resource='RD'/>
<lib:pages>240</lib:pages>
</rdf:Description>

<rdf:Description about="Heart of Darkness">
<lib:creator rdf:resource='JC' />
<lib:pages>110</lib:pages>
</rdf:Description>
```

다시 본론으로 돌아가서, <feature:color> 속성은 3번째 줄의 `xmlns:feature`에 이미 포함되어 있다는 것도 이해하여야 한다. 다시 말해서, `xmlns:feature`의 이름 리스트에는 이미 `size`

와 color라는 속성이름이 포함되어 있다는 것이다. 따라서 위의 T-shirt 지시문은 "이 주체는 ID가 <http://www.linkeddatatools.com/colors#white>인 지시문을 참조하고 있는 객체와 더불어 feature:color이라는 이름의 한 개의 속성을 가지고 있다"라고 말하고 있다.

2.2 Breaking Down The Statement

이제 간단한 예의 RDF 다크문트를 살펴보고, 우리가 배운 것을 근거로 지시문의 구성요소를 분석해 보면 다음과 같다:

1. `<rdf:Description rdf:about="subject">`
2. `<predicate rdf:resource="object" />`
3. `<predicate>literal value</predicate>`
4. `</rdf:Description>`

2.3 A More Thorough Example

이제 맨 처음에 지나 왔던 그래프 데이터에서 다루었던 복잡한 예로 되돌아가 보자:

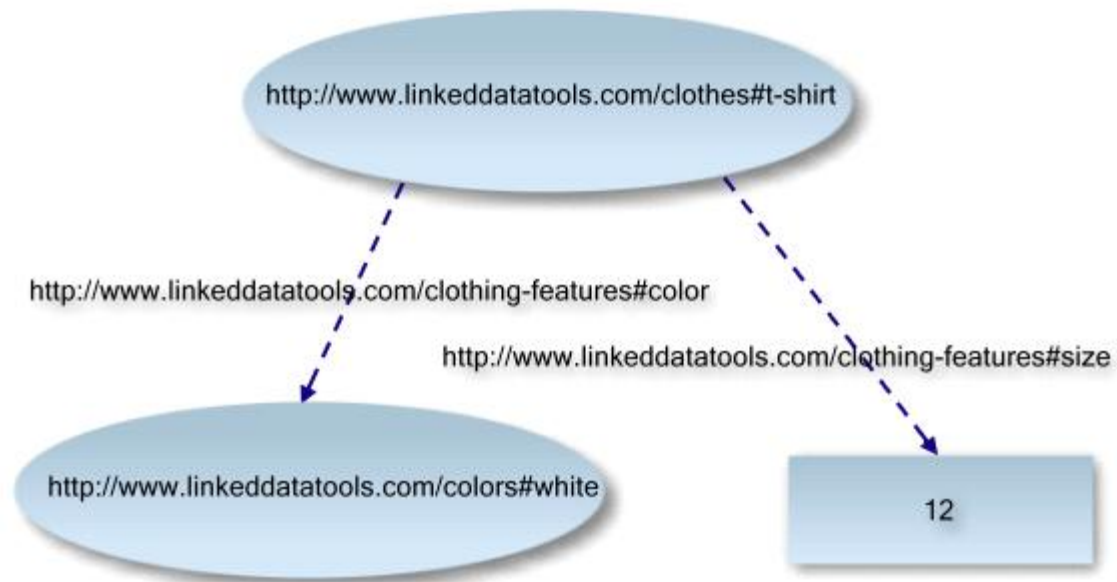
```
01.<?xml version="1.0" encoding="UTF-8"?>
02.
03.<rdf:RDF
04.xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
05.xmlns:dc="http://purl.org/dc/elements/1.1/"
06.xmlns:region="http://www.country-regions.fake/">
07.
08.<rdf:Description rdf:about="http://en.wikipedia.org/wiki/Oxford">
09.<dc:title>Oxford</dc:title>
10.<dc:coverage>Oxfordshire</dc:coverage>
11.<dc:publisher>Wikipedia</dc:publisher>
12.<region:population>10000</region:population>
13.<region:principaltown rdf:resource="http://www.country-regions.fake/oxford"/>
14.</rdf:Description>
15.
16.</rdf:RDF>
```

위의 예를 보고 자신의 이해력을 테스트하여 보완할 분야가 있는지를 알기 위하여, 다음의 질문에 답할 수 있는지를 스스로 확인해 보라:

- Subject of the statement?
- Predicates of the statement(including whether they are resources or literals)?
- Objects referenced by the **resource** predicates?

이제 RDF 다큐먼트를 이해했고, 그것이 데이터 그래프와 어떤 관계가 있는지를 알았다면, 여러분은 RDF graph data로 시멘틱스를 모델링하는 방법에 대하여 알 준비가 끝났다.

이제 우리는 T-shirt RDF에서 완전하게 자격을 갖춘 URIs는 다음의 그래프와 같다고 말할 수 있다:



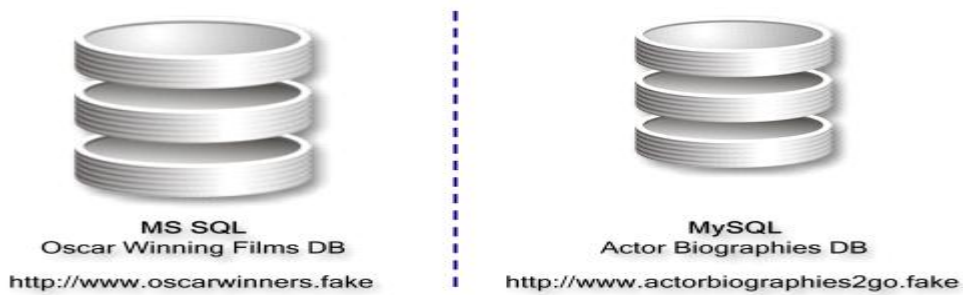
3. Semantic Modeling

RDF가 전 세계적으로 교환 가능한 데이터를 레코딩하기 위하여 유연하고 그래프-중심의 모델을 제공하는 반면에, 그것이 어떤 어의나 의미를 제공하지는 않는다. 이제 일반적으로 이용 가능한 데이터 모델을 검토해서 문제점이 무엇인지에 대해 알아보자.

3.1 Why Include Semantics In Data? Knowledge Integration

데이터에 시멘틱 의미를 추가하는 가장 중요한 이익들 중의 하나는 자동적으로 다양한 지식의 영역으로 가치를 뻗칠 수 있다는 것이다. 지식의 도메인(domains of knowledge)이란 무엇인가? 간단한 예로 살펴보자.

위의 예에서, 두 개의 웹 사이트들은 서로 독립적으로 출발하였다. 한 사이트는 현재와 과거의 Oscar 수상 영화에 대한 정보를 호스팅하고 있으며, 다른 사이트는 할리우드 남녀 영화배우의 전기에 대한 대규모의 데이터베이스 이다.



그렇지만 둘 다 자신들의 웹 사이트 데이터베이스에 보완할 정보가 있다. 우리는 먼저 이들 사이트 간에 시멘틱의 사용 없이 어떻게 정보공유가 이루어지는지를 살펴보고 난 다음에, 시멘틱스를 사용하여 동일한 정보를 두 사이트가 어떻게 서로 공유하는지에 대하여 알아보기로 한다.

1) Sharing Without Semantic Modeling

두 사이트 중에서 하나는 모든 오스카 수상영화에 대한 MS SQL database이고, 또 하나는 할리우드 배우에 대한 MySQL database이다. 이것들은 <http://www.oscarwinners.fake> 그리고 <http://www.actorbiographies2go.fake>에 각각 포함되어 있다. 그리고 이 두 사이트는 서로 독립적이며, 협력하지 않는다.

Oscar Winners 사이트는 그 이름처럼 과거에 제작된 모든 오스카 수상영화를 리스트하고 있으며, 또한 그것들에서 주연을 맡은 남녀 영화배우의 리스트도 갖고 있다. 그렇지만, 그 콘텐츠에는 이들의 이름과 생일날짜 이외의 다른 정보는 소장하고 있지 않다.

Actor Biographies 사이트는 현재뿐만 아니라 과거의 많은 할리우드 배우들에 대하여 완전한 전기 리스트뿐만 아니라 그들이 출연한 영화 리스트도 가지고 있다. 그러나 이것의 콘텐츠에는 해당 영화에 대한 어떠한 film-plots(영화구성), 또는 screen-shot(image)을 가지고 있지 않다.

이제 이 두 사이트가 최신의 데이터 모델보다 기존의 전통적인 데이터 모델을 사용하여 협력할 수 있는 방법에 대하여 살펴보자:

- 분명하게 말해서, <http://www.oscarwinners.fake>의 이용자는 출연배우의 이름을 클릭하면, 그들에 대해 더 많은 정보를 찾을 수 있다는 이점이 있다. - 이 정보는 <http://www.actorbiographies2go.fake>의 MySQL database에 저장되어 있다.

- 똑같이, <http://www.actorbiographies2go.fake>의 이용자가 출연배우들의 영화이름을 클릭하여 더 많은 정보를 얻을 수 있다. 이것은 <http://www.oscarwinners.fake>에 있는 MS SQL database에 저장되어 있다.

■ 두 사이트간의 어떠한 데이터 공유도 자신들의 데이터베이스에 있는 테이블을 결합(joining)시키지 못하고 있다. 그 이유는 먼저, 이것들은 처음부터 독립적으로 설계되었으므로, 두 데이터베이스에 있는 각각의 영화배우나 영화를 참조하는 primary key를 동기화(synchronized)시킬 수 없다. 이 문제를 해결하기 위하여 이것들은 mapped되어야 할 것이다. 그리고 또 하나의 문제는 이것들이 서로 호환성이 없는 데이터베이스 서버 시스템을 사용하고 있다는 것이다.

■ 자신들이 현재 사용하고 있는 데이터베이스의 협력을 위하여, 각 사이트의 소유자들은 공동으로 영화 및 배우에 대한 고유한 ID scheme을 만들어 정보를 공유할 수 있는 공동의 데이터 포맷을 결정해야 할 것이다. 예를 들어, 자신들의 웹사이트에서 서로의 필요에 따라 정보를 요구할 수 있는 안전한 XML 콘텐츠를 만든다면, 이것이 가능할 것이다. 이런 방식으로 자신들의 정보 공유를 발전시킬 수 있다.

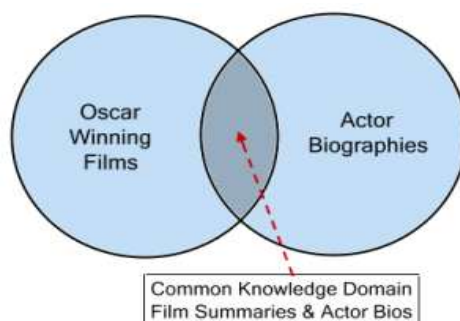
이제 RDF와 semantics을 소개하기로 한다. RDF와 SW를 사용하여 위와 같은 문제를 해결할 수 있는 방법에 대하여 알아보자 - 모든 것은 자동적으로 이루어지며, 수동적으로 이루어지진 않는다.

2) Sharing With The Semantic Web Model

시멘틱 모델링에서, 먼저 다음과 같은 중요한 용어에 대해 이해하여야 한다: Vocabulary 와 Ontology

- Vocabulary - contexts 간에 정의가 잘 일치하는 용어들의 집합이다.
- Ontology - 우리로 하여금 정의된 vocabulary 뒤에 숨어있는 contextual relationships를 정의한다. 이것은 지식의 영역을 정의하는 cornerstone 이다. 온톨로지를 공식적으로 정의하기 위해 사용하는 언어는 OWL (Web Ontology Language)이며, 이것은 RDFS (RDF Schema)를 확장시킨 것으로 다음 강의에서 설명하기로 한다.

본론으로 돌아가서, 시멘틱 모델링을 사용하여 우리가 두 개의 사이트의 시나리오를 어떻게 모델화 하여야 하는가?



무엇보다도 먼저, 두 사이트들은 공동의 표준적인 vocabulary를 개발하여, 이것을 문맥상에서(contextually) 일치하는 자신들의 데이터를 기술하는데 적용하여야 한다. 예를 들어, 용어 'film title'은 두 사이트 모두에서 동일한 사물(thing)을 의미하여야 하며, 용어 'actor name'과 'actor birthdate'도 마찬가지 이다.

이러한 활동을 통해, vocabulary 용어로 데이터의 잠재적 의미를 표현할 수 있으며, 또한 쿼리에 의해서도 동일한 데이터를 얻을 수 있다. 똑같은 결과가 똑같은 ontology나 common vocabulary를 채택하고 있는 두 사이트에 의해 나타날 것이다. 최종적으로 이들 두 사이트는 웹에서 서로 교신(communicate)할 수 있다.

만일 이러한 표준 vocabulary가 적재적소에 사용된다면(in place):

- 두 사이트는 동일한 용어로 서로에게 질의할 수 있다;
- The Oscar Winning Movies site는 on-demand 방식으로 the Actor Biographies site의 배우 이름을 쿼리할 수 있으며, 그 영화에 출연한 특별한 남녀배우에 대하여 보다 자세한 정보를 얻을 수 있다.
- The Actor Biographies site 역시 on-demand 방식으로 Oscar Winning Movies site에 있는 film plots을 이젠 쿼리할 수 있으며, 배우가 출연했던 영화들에 대하여 더 많은 자세한 정보를 얻을 수 있다.
- 이용자는 이러한 링크용 표준용어(linked standard terminology)를 사용하여, 공식적인 웹 온톨로지에서 정의하고 있는 contextual relationships(문맥전후 연관성)을 이용하여 영화배우나 영화에 대한 촬영장소와 같은 추가적 연관 정보, 영화촬영과 같은 날에 일어난 다양한 뉴스, 배우의 생년월일, 또는 동일한 제작자에 의해 제작된 영화 등등의 부수적 정보를 얻을 수도 있으며, 처음에는 이러한 정보를 얻는다고 상상조차 하지 못 했다.
- 이것은 두 사이트간에 이미 존재하고 있는 transformation, mapping, 또는 contracts와 같은 요구조건과 상관없이 이루어져야 한다. 결과적으로 이 사이트의 모든 정보는 바로 의미론(semantics)에 의해 생산된다.

우리는 다음 강의에서 SW 온톨로지의 구성(makeup)이 무엇이고, 여러분이 시멘틱 데이터베이스에서 어떻게 쿼리하는지, 그리고 심지어 그것에서 기계적 추론(machine inference)을 형성하는 방법에 대해 배울 것이다.

3.3 Metadata Initiatives

지식의 도메인에서 용어를 표현하는 표준 vocabularies, 또는 공식적 ontologies는 이미 다

양한 주제 - 예를 들어, media terms, biomedical terms, scientific terms - 에 대한 표준 vocabularies를 만드는 여러 전문기관에서 무료로 이용할 수 있다. 몇 가지 예를 살펴보면 다음과 같다:

- Dublin Core Metadata Initiative (DCMI) - 특히 공통적이고 일상적인 용어 그리고 미디어의 주요 용어에 특별하게 초점을 맞추면서 여러 주제에 대한 온톨로지를 만들고 있다.

- Friend Of A Friend (FOAF) - 사람, 그들의 활동, 다른 사람과의 관계 등을 기술하고 있는 기계가독형 온톨로지이며, RDF와 OWL을 사용하여 표현된 기술적 vocabulary 이다.

- OpenCyc(오픈사이크) - 일상적인 상식을 수집해 놓은 세계 최대의 온톨로지 이다.

The OpenCyc Platform is your gateway to the full power of Cyc, the world's largest and most complete general knowledge base and commonsense reasoning engine. OpenCyc contains hundreds of thousands of Cyc terms organized in a carefully designed ontology.

이제 SW에서 온톨로지를 정의하는 방법과 SW 데이터베이스에서 정보를 쿼리하는 방법에 대한 보다 완벽한 기술적 내용을 알아보자:

4: Introducing RDFS & OWL

이제 의미론과 함께 RDF data models을 다루는데 필요한 실질적인 기술에 대해 알아보자. RDF data는 RDFS 그리고 OWL과 같은 두 가지의 구문법(syntaxes)을 사용하여 시맨틱 메타데이터와 함께 표현(encoded) 될 수 있다:

이제부터 우리는 semantic metadata를 가지고 RDF 데이터를 주해(annotate)하는데 사용되는 공식적인 구문법을 알아본 다음에, 이런 데이터를 출판하고 쿼리하는 방법에 대해 알아볼 것이다.

RDF 데이터는 두 가지의 중요한 구문법인 RDFS와 OWL을 사용하여 시맨틱 메타데이터와 함께 표현된다. RDFS와 OWL 둘 다 W3C specifications 이다.

4.1 A Starting Example

여러분이 보고 있는 것이 OWL의 예이다:

```

01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
04.   xmlns:owl="http://www.w3.org/2002/07/owl#"
05.   xmlns:dc="http://purl.org/dc/elements/1.1/">
06.
07. <!-- OWL Header Example -->
08. <owl:Ontology rdf:about="http://www.linkeddatatools.com/plants">
09. <dc:title>The LinkedDataTools.com Example Plant Ontology</dc:title>
10. <dc:description>An example ontology</dc:description>
11. </owl:Ontology>
12.
13. <!-- OWL Class Definition Example -->
14. <owl:Class rdf:about="http://www.linkeddatatools.com/plants#planttype">
15. < rdfs:label>The plant type</rdfs:label>
16. <rdfs:comment>The class of plant types.</rdfs:comment>
17. </owl:Class>
18.
19. </rdf:RDF>

```

위의 내용에 대해 지금 자세히 알려고 하지마라. 나중에 살펴보기로 하자. 그렇지만, 주목해야 하는 것은 예제인 RDF 다큐먼트의 header에 전에는 없었던 두 개의 새로운 namespaces가 포함되어 있다는 것이다:

- 1) 3번째 줄의 RDFS (RDF Schema, <http://www.w3.org/2000/01/rdf-schema#>).
- 2) 4번째 줄의 OWL (Web Ontology Language, <http://www.w3.org/2002/07/owl#>)

의 namespaces 이다.

한 가지 더 주목할 것은 RDF에서 온톨로지를 정의하는 방법이다. 물론, 온톨로지 역시 RDF 다큐먼트 이다.

이제 전통적인 온톨로지 문서를 분해해 보자. 예로서, plant varieties(식물의 종류)를 정의하고 있는 간단한 온톨로지를 살펴보자.

4.2 OWL Header

```
01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
04.   xmlns:owl="http://www.w3.org/2002/07/owl#"
05.   xmlns:dc="http://purl.org/dc/elements/1.1/">
06.
07. <!-- OWL Header Example -->
08. <owl:Ontology rdf:about="http://www.linkeddatatools.com/plants">
09. <dc:title>The LinkedDataTools.com Example Plant Ontology</dc:title>
10. <dc:description>An example ontology written for the LinkedDataTools.com RDFS & OWL Introduction
    tutorial</dc:description>
11. </owl:Ontology>
12.
13. <!-- Remainder Of 다큐먼트 Omitted For Brevity... -->
14.
15.</rdf:RDF>
```

비록 온톨로지가 헤더에 포함되지 않아야 하더라도, 헤더는 여러분의 온톨로지가 무엇을 포함하고 있는지를 타인이 이해하는데 도움이 되는 정보를 포함시키기에 좋은 장소이다.

위에서와 같이, 우리는 온톨로지용으로 한 개의 title과 한 개의 description을 사용하였다. 또한 이곳은 갱신에 관한 version information을 포함시켜야 하는 장소이며, 여러분의 온톨로지가 다른 온톨로지서서 수입(import)된 것임을 표기하는 장소이기도 하다.

!!! **Point:** prefix **dc:** 의 namespace를 정의한 5번째 줄을 보라. 이것은 Dublin Core Metadata Initiative라는 namespace를 참조하며, machine readers에게 **dc:title** and **dc:description**와 같은 엘리먼트들은 이 온톨로지서서 정의하고 있다고 알려주고 있다.

4.3 OWL Classes, Subclasses & Individuals

온톨로지의 제 1차적 목적은 사물을 semantics나 meaning에 따라 분류하는 것이다. OWL에서, 이런 목적은 *classes* 와 *subclasses*를 사용하여 이룰 수 있다. 예를 들어, OWL에서는 이것을 *individuals(instance)* 라 부른다. 특정한 OWL class의 멤버가 individuals이며, 그것들이 확장된 것을 *class* 라 부른다. OWL에서 *class* 는 공동의 성질을 갖고 있는 individuals의 그룹이다.

An Example

다시 OWL ontology의 예를 보자. 이번에는 몇 가지의 classes와 subclasses가 추가되었다. 우리는 3가지의 plant classes를 정의한다:

- ▶ the **flowering plants** class,
- ▶ **shrubs** class,
- ▶ 위의 두 가지 클래스를 섭클래스로 가지고 있는 **planttype** class. 따라서 **planttype** class
는 모든 식물종류에서 최상위 클래스이다.

```

01. <rdf:RDF
02. xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03. xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
04. xmlns:owl="http://www.w3.org/2002/07/owl#"
05. xmlns:dc="http://purl.org/dc/elements/1.1/"
06. xmlns:plants="http://www.linkeddatatools.com/plants#">
07.
08. <!-- OWL Header Omitted For Brevity -->
09.
10. <!-- OWL Class Definition - Plant Type -->
11. <owl:Class rdf:about="http://www.linkeddatatools.com/plants#planttype">
12.
13. <rdfs:label>The plant type</rdfs:label>
14. <rdfs:comment>The class of all plant types.</rdfs:comment>
15.
16. </owl:Class>
17.
18. <!-- OWL Subclass Definition - Flower -->
19. <owl:Class rdf:about="http://www.linkeddatatools.com/plants#flowers">
20.
21. <!-- Flowers is a subclassification of planttype -->
22. <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/plants#planttype"/>
23.
24. <rdfs:label>Flowering plants</rdfs:label>
25. <rdfs:comment>Flowering plants, also known as angiosperms.</rdfs:comment>
26.
27. </owl:Class>
28.
29. <!-- OWL Subclass Definition - Shrub -->
30. <owl:Class rdf:about="http://www.linkeddatatools.com/plants#shrubs">
31.
32. <!-- Shrubs is a subclassification of planttype -->
33. <rdfs:subClassOf rdf:resource="http://www.linkeddatatools.com/plants#planttype"/>
34.
35. <rdfs:label>Shrubbery</rdfs:label>
36. <rdfs:comment>Shrubs, a type of plant which branches from the base.</rdfs:comment>
37.
38. </owl:Class>
39.
40. <!-- Individual (Instance) Example RDF Statement -->
41. <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">
42.
43. <!-- Magnolia is a type (instance) of the flowers classification -->
44. <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>
45.
46. </rdf:Description>

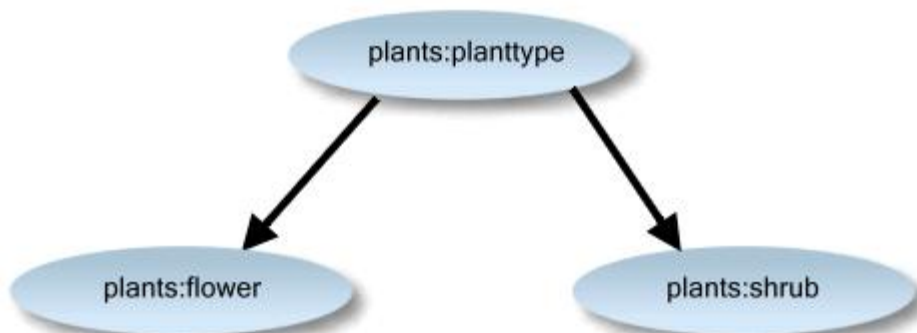
```

47.

48. </rdf:RDF>

Taxonomy - A Hierarchy Of Terms

우리가 한 것은 우리의 어의적 용어나 클래스를 계층적으로 정의한 것이다. SW 세계에서, 이러한 용어의 계층을 *taxonomy*라 부른다. 다음은 우리가 정의한 taxonomy hierarchy을 그래프로 나타낸 것이다:



An example of a taxonomy hierarchy

Note - 우리는 magnolis(목련)라 부르는 flower 클래스의 또다른 섹클래스를 만들지 않았다. 그것보다는 magnolia는 flower 클래스의 individual(instance) 이다. 왜 이런가? magnolia는 flower classification의 한 멤버이지만, 그것은 더 이상 flower subclassification이 아니다. 이것은 당연히 magnolia의 어의적 관점에서 보면 이것은 flower 클래스의 individuals(instances)이지 subclassification 즉, 다른 꽃이 아니라는 의미이다.

4.4 OWL Properties

OWL에서 Individuals은 *properties*와 관련 있다. OWL에는 두 종류의 property가 있다:

- **Object properties** (owl:ObjectProperty): 두 가지 OWL classes의 individuals (instances)와 관련이 있다.
- **Datatype properties** (owl:DatatypeProperty): OWL classes의 individuals (instances)의 literal values와 관련 있다.

An Example

먼저 a *data type* property (one which links an instance to a literal value)을 추가한 다음에, Magnolia가 속해 있는 *species family*의 이름을 추가해 보자.

```

01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
04.   xmlns:owl="http://www.w3.org/2002/07/owl#"
05.   xmlns:dc="http://purl.org/dc/elements/1.1/"
06.   xmlns:plants="http://www.linkeddatatools.com/plants#">
07.
08.   <!-- OWL Header Omitted For Brevity -->
09.
10.   <!-- OWL Classes Omitted For Brevity -->
11.
12.   <!-- Define the family property -->
13.   <owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/plants#family"/>
14.
15.   <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">
16.
17.     <!-- Magnolia is a type (instance) of the flowers class -->
18.     <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>
19.
20.     <!-- The magnolia is part of the 'Magnoliaceae' family -->
21.     <plants:family>Magnoliaceae</plants:family>
22.
23.   </rdf:Description>
24.
25. </rdf:RDF>

```

Important Point - 여기서, 만일 여러분이 object oriented programmer라면, 여러분은 마음으로부터 programmatic object classes and their associated properties를 생각해 낸 다음에, 그것들을 OWL classes에 대해 배운 것과는 비교하려 할 것이다. 그러지 마세요 - 정말 달라요. 위의 예제를 주목해 보자. 'family' property는 어떠한 class type과도 독립적이며, class flower (magnolia)의 instance에 할당된다. 똑같은 클래스의 또다른 instance는 이 property를 갖지 않을 수 있다. OWL에서는 그러하다. instance를 가지고 있는 properties가 그것들의 class types이 아니라 그것들의 instance를 기술한다는 것을 주목하라. 이 경우에, 여러분은 완전히 서로 다른 클래스용으로 동일한 'family' property를 사용할 수 있다.

끝으로, an *object* property (one which links an instance to another instance)를 추가해 보자. 우리가 상점을 운영하고 있고 이 식물(Magnolia)을 상점주인과 마찬가지로 우리도 인기가 있는 다른 식물에 링크하길 원한다고 하자(Let's say we're running a shop, and we want to link this plant (Magnolia) to another plant which we know as the shop owner is equally as popular). "similarlyPopularTo"라는 property를 추가해 보자:

```

01. <rdf:RDF
02.   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
03.   xmlns:owl="http://www.w3.org/2002/07/owl#"
04.   xmlns:dc="http://purl.org/dc/elements/1.1/"
05.   xmlns:plants="http://www.linkeddatatools.com/plants#">

```

```

06.
07. <!-- OWL Header Omitted For Brevity -->
08.
09. <!-- OWL Classes Omitted For Brevity -->
10.
11. <!-- Define the family property -->
12. <owl:DatatypeProperty rdf:about="http://www.linkeddatatools.com/plants#family"/>
13.
14. <!-- Define the similarlyPopularTo property -->
15. <owl:ObjectPropertyrdf:about="http://www.linkeddatatools.com/plants#similarlyPopularTo"/>
16.
17. <!-- Define the Orchid class instance -->
18. <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#orchid">
19.
20. <!-- Orchid is an individual (instance) of the flowers class -->
21. <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>
22.
23. <!-- The orchid is part of the 'Orchidaceae' family -->
24. <plants:family>Orchidaceae</plants:family>
25.
26. <!-- The orchid is similarly popular to the magnolia -->
27. <plants:similarlyPopularTordf:resource="http://www.linkeddatatools.com/plants#magnolia"/>
28.
29. </rdf:Description>
30.
31. <!-- Define the Magnolia class instance -->
32. <rdf:Description rdf:about="http://www.linkeddatatools.com/plants#magnolia">
33.
34. <!-- Magnolia is an individual (instance) of the flowers class -->
35. <rdf:type rdf:resource="http://www.linkeddatatools.com/plants#flowers"/>
36.
37. <!-- The magnolia is part of the 'Magnoliaceae' family -->
38. <plants:family>Magnoliaceae</plants:family>
39.
40. <!-- The magnolia is similarly popular to the orchid -->
41. <plants:similarlyPopularTordf:resource="http://www.linkeddatatools.com/plants#orchid"/>
42.
43. </rdf:Description>
44.
45. </rdf:RDF>

```

예를 들어, 우리는 URI <http://www.linkeddatatools.com/plants#orchid>와 더불어 Orchid를 표현하는 flowers 클래스의 새로운 individual(instance)을 정의하였다. 여러분이 우리가 동일한 클래스의 individual인 Mangnolia instance를 정의한 방법으로부터 이것을 이해할 수 있는지 확인해 보라. 이제, 우리의 첫 번째 두 개의 tutorials에서처럼, 여러분이 위에서 정의한 RDF graph에 따라 Orchid 그리고 Magnolia class instances와 이것들의 predicates를 보여주는 그래프를 그릴 수 있는지 확인해 보라.

Hint: 여러분은 양쪽 the *family* 그리고 *similarlyPopularTo* properties을 나타내는 화살표를 그려야 할 것이다. *family* property와 관련해서, 각각의 plant 용으로 서로 다른 family type literals을 포인트(point)해야 할 것이다. 그리고 *similarlyPopularTo* property와 관련

해서는 the instances가 서로서로 포인트해야 할 것이다.

Important - Point Note

위의 예에서, 우리는 *similarlyPopularTo* object property를 통해 똑같은 OWL 클래스의 2 instances간을 링크하는 a two way를 갖는다(Orchid와 Magnolia 둘 다 똑같은 클래스의 individuals 이다). 그렇지만 주목할 것은 object properties는 two way가 아닐 수도 있다 - 그것들은 they may be one way일 수도 있다는 것이다. 그리고 중요한 것은 똑같은 OWL 클래스의 instances 간이 일어나지 않아야 한다(need not be between instances of the same OWL class). 그것들은 완전히 다른 OWL classes일 수 있다.

5. Querying Semantic Data

이제 여러분은 RDF에 시멘틱스를 추가하는 방법, 그것을 출판하고 쿼리하는 방법을 알고 있다. 관계형 데이터베이스의 테이블에서 SQL을 사용하여 쿼리하는 것처럼, RDF 데이터의 triples는 SPARQL을 사용하여 쿼리한다. 우리는 몇 가지 기본적인 쿼리를 조사하여 SPARQL과 SQL을 비교해 본다.

만일 여러분이 전통적인 IT에 대한 배경을 갖고 있다면, 여러분은 이미 MS SQL or MySQL과 같은 관계형 데이터베이스에서 데이터를 검색하는데 사용하는 SQL(Structured Query Language, pronounced "sequel")에 대하여 잘 알고 있을 것이다.

비슷하게, RDF data stores 역시 자신들의 쿼리 언어인 SPARQL (SPARQL Protocol and RDF Query Language, pronounced "sparkle")을 사용하여 쿼리한다. 그렇지만 SPARQL은 좀 더 고급스럽다.

5.1 A Starting Example

SPARQL Is Similar To SQL

SQL처럼, SPARQL은 선택된 데이터의 어떠한 하위집합을 얻을 것인지를 결정하기 위하여 SELECT statement를 사용하여 query data set로부터 데이터를 선택한다.

또한 SPARQL 은 쿼리 데이터 세트에서 매치되는 것을 찾기 위한 그래프 패턴(graph pattern)을 정의하기 위하여 WHERE clause을 사용한다. SPARQL의 WHERE clause에서 graph pattern은 데이터에서 매치되는 것을 찾기 위하여, subject, predicate 그리고 object triple로 구성된다.

Note - SPARQL에서, 변수 이름(variable names)은 question mark ("?) symbol을 접두사(prefix)로 갖는다. query graph pattern에서, 이것들은 어떤 node와 match 한다 - 그것

이 resource 또는 literal이든 상관없다.

5.2 SPARQL General Form

SPARQL queries 는 아래와 같은 일반적 행태를 가지고 있다. 이것은 하나의 쿼리는 여러 섹션으로 세분될 수도 있으며, 그 섹션을 정의하는 clause or keyword가 있다는 것을 보여주고 있다.

PREFIX (Namespace Prefixes)

e.g. PREFIX plant: <http://www.linkeddatatools.com/plants>

SELECT (Result Set)

e.g. SELECT ?name

FROM (Data Set)

e.g. FROM <http://www.linkeddatatools.com/plantsdata/plants.rdf>

WHERE (Query Triple Pattern)

e.g. WHERE { ?planttype plant:planttype ?name }

ORDER BY, DISTINCT etc (Modifiers)

e.g. ORDER BY ?name

또는, 위에서 나타난 각 섹션의 예를 근거로, 우리는 다음과 같은 쿼리를 작성할 수 있다:

1. PREFIX plant: <http://www.linkeddatatools.com/plants>
2. SELECT ?name
3. WHERE {
4. ?planttype plant:planttype ?name.
5. }
6. ORDER BY ?name

5.3 SQL과 SPARQL의 비교

MA에 살고 있는 사람의 주소를 얻기 위한 SQL query는 다음과 같다:

```
SELECT Person.fname, Address.city
FROM Person, Address
WHERE Person.addr=Address.ID
AND Address.state="MA"
```

Conceptually, we are SELECTing a list of attributes FROM a set of tables WHERE certain constraints are met. These constraints capture the relationships implicit in the scheme, `Person.addr=Addresses.ID`, and the selection criteria, e.g. `Address.state="MA"`.

똑같은 데이터에 대한 SPARQL query는 다음과 같다:

```
SELECT ?name ?city
WHERE {
    ?who <Person#fname> ?name ;
        <Person#addr> ?adr .
    ?adr <Address#city> ?city ;
        <Address#state> "MA"
}
```

SPARQL reuses some key words familiar to SQL users: SELECT, FROM, WHERE, UNION, GROUP BY, HAVING and most aggregate function names.

5.4 Dbpedia에서의 실전 예제

<http://dbpedia.org/sparql> 에 들어가서 일단 아래 질의를 입력한 후 'Run Query'를 클릭해보도록 하자.

```
SELECT ?uri ?name ?page ?nick
WHERE{
    ?uri a foaf:Person ;
    foaf:name ?name;
    foaf:page ?page;
    foaf:nick ?nick.
} LIMIT 100
```

Dbpedia에서는 기본적으로 PREFIX가 많이 되어있으므로 쓰지 않아도 된다.:

- ▶ 첫 번째 줄에서 ?uri, ?name, ?page, ?nick은 이 네 개의 항목을 가져오겠다는 것이다.
- ▶ 세 번째 줄에서 ?uri a foaf:Person: 은 foaf:Person이라는 type을 가진 모든 항목을 ?uri라는 변수로 사용하겠다는 명령이다. (a == rdfs:typeOf)
- ▶ 네 번째 줄에서 foaf:name ?name 은 ?uri로 선택된 항목의 foaf:name을 ?name이라는 변수에 넣겠다는 명령이다.
- ▶ 다섯 번째 줄과 여섯 번째 줄도 같은 의미이다. 그리고 마지막 줄에서 LIMIT 100은 최대 100개 가져오겠다는 명령이다.

요약하자면, foaf:Person이라는 타입을 가진 모든 자원에서 foaf:name, foaf:page, foaf:nick의 데이터를 각각 ?name, ?page, ?nick이라는 변수에 넣어서 출력하겠다는 것이다. 다음 그림은 위 질의에 대한 결과화면이다. 결과화면은 html로 출력 선택을 하여 웹 페이지 형식으로 보이며 xml, RDF/XML, JSON 등 다양한 형태로 전송 받을 수 있다.

VI. 국립중앙도서관: Linked Open Data for NLK

<http://lod.nl.go.kr/home/about/introduction.jsp>

SPARQL Endpoint 페이지 보기

▶ SPARQL(Simple Protocol and RDF Query Language)이란?

온톨로지 질의 언어 입니다. W3C 에서 만든 RDF 질의 언어 이며 관계형 데이터 베이스 에 SQL 이 있다면 RDF 에는 SPARQL 이 있습니다. 관계형 데이터베이스 에 저장된 데이터로부터 원하는 정보를 꺼내오기 위해 SQL을 활용 하듯이, 웹에 공개된 각종 RDF 데이터들로부터 우리가 원하는 데이터를 꺼내오기 위해 SPARQL 을 사용합니다.

▶ SPARQL Endpoint 란?

intro3_icon_a SPARQL Endpoint는 웹을 통해 SPARQL을 질의할 수 있는 접근점을 의미합니다. SPARQL Endpoint의 URL을 이용하여 질의를 작성하고 그에 해당하는 결과를 다양한 유형의 데이터 포맷으로 받을 수 있습니다.

<DB Query>

```
SELECT id, name
FROM author
WHERE birthYear=1933
ORDER BY id DESC
LIMIT 3
OFFSET 4
```

<SPARQL Query>

```
SELECT ?id ?name
WHERE {
    ?id rdf:type <http://lod.nl.go.kr/ontology/Author> ;
    <http://xmlns.com/foaf/0.1/name> ?name .
}
ORDER BY DESC(?id)
LIMIT 3
OFFSET 4
```


<PREFIX란?>

PREFIX nlon: <http://lod.nl.go.kr/ontology/>

PREFIX foaf: <http://xmlns.com/foaf/0.1/>

SELECT ?id ?name

WHERE {

 ?id rdf:type nlon:Author ;

 foaf:name ?name .

}