COMS 4995

Introduction to Semantic Web

Lecture 3: RDF

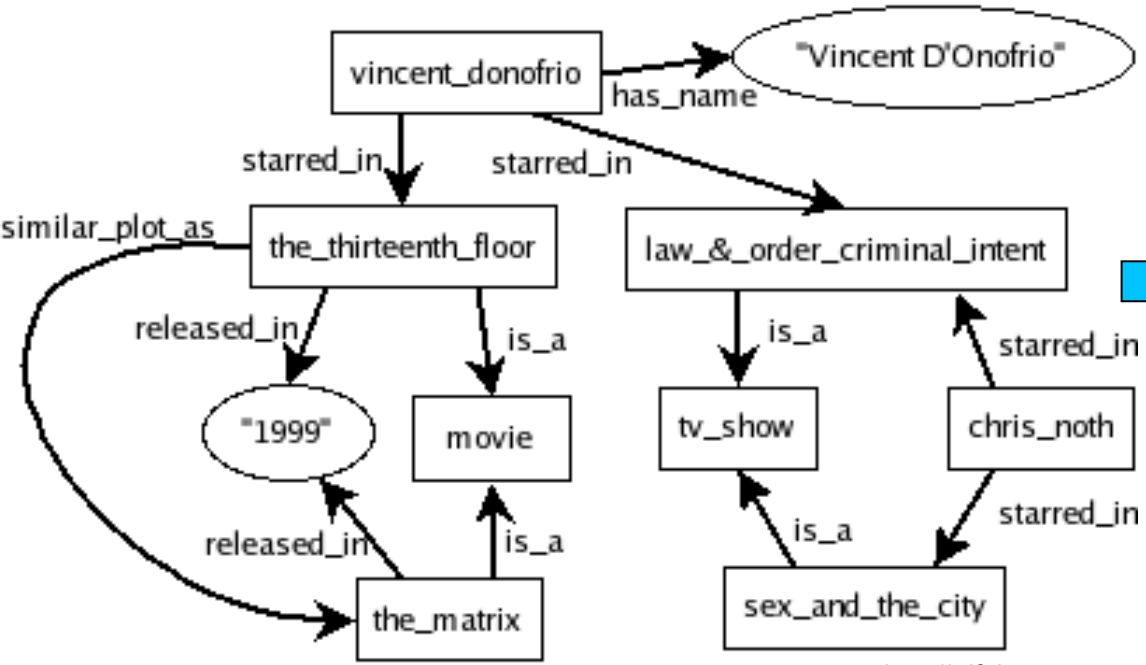
Knarig Arabshian

Data Sharing Problem

- On the Semantic Web, computers do the browsing for us.
- It enables computers to seek out knowledge distributed throughout the Web, mesh it, and then take action based on it.
- To use an analogy, the current Web is a decentralized platform for distributed *presentations* while the Semantic Web is a decentralized platform for distributed *knowledge*.
- RDF is the W3C standard for encoding knowledge.

Data Sharing Problem

- Reasons why we need a standard for distributed knowledge
  - Files on the Semantic Web need to express information flexibly: best expressed as a *graph.*



Start Node	Edge Label	End Node
vincent_donofrio	starred_in	law_&_order_ci
law_&_order_ci	is_a	tv_show
the_thirteenth_floor	similar_plot_as	the_matrix
...		

http://rdfabout.com

Data Sharing Problem

- Files on the Semantic Web need to be able to relate to each other.
  - file about product prices posted by a vendor and a file with product reviews posted independently by a consumer need to have a way of indicating that they are talking about the same products.
  - need globally unique identifiers that can be assigned in a decentralized way.
- We will use vocabularies for making assertions about things, but these vocabularies must be able to be mixed together.
  - vocabulary about TV shows developed by one group and a vocabulary about movies developed by another group must be able to be used together in the same file
  - talk about the same things, for instance to assert what 'actor' means and that a particular actor has appeared in both TV shows and movies.

Data Sharing Problem: Data vs. MetaData

- Data
  - Values or individual atoms of information
- Metadata
  - Describes relationship between data
  - Changes much less frequently than data
- Database can have tens or hundreds of tables but have thousands of rows
- Tables will not change often. Rows in tables will.

Data Sharing Problem: Modeling Information

Object-oriented Approach

```
Person
+FirsName: string
+LastName: string
+MiddleInitial: string
+Addresses: Address[]
```

```
AddressStreetNo:
string
+AptOrSuite: string
+City: string
+StateOrProvince:
string
+PostalCode: string
```

Relational Approach

```
CREATE TABLE Person {
ID: LONG,
FirstName: VARCHAR,
LastName: VARCHAR,
MiddleInitial: CHARACTER
};
```



```
CREATE TABLE Address {
ID: LONG
StreetNo: VARCHAR,
AptOrSuite: VARCHAR,
City: VARCHAR,
StateOrProvince: VARCHAR,
PostalCode: INTEGER
};
```



```
CREATE TABLE Person_Address_Link {
ID: LONG
PersonID: LONG,
AddressID: LONG
};
```

XML Approach

```
<Person>
  <FirsName>   </FirsName>
  <LastName>  </LastName>
  <MiddleInitial></MiddleInitial>
  <Addresses></Addresses>
</Person>
```

Data Sharing Problem: Syntax vs Semantics

- Objects
  - Syntactic Challenge
    - Consuming system must know detail of producer's *data structures* to address syntactic data-sharing problem
  - Semantic Challenge
    - Consumer must know *object data members* and how each maps to corresponding data structure within the consumer
- Consider how people communicate with one another:
  - medium of communication: words, smoke signals, morse code (syntax)
  - common language (semantics)

Data Sharing Problem: Syntax vs Semantics

- Serialized Objects
  - Binary object serialization
    - Producer generates objects that represent data to be shared
    - Data values in these objects are serialized in ordered collection of bytes
- Relational Database
  - Improves on serialized objects with use of standardized APIs (ODBC, JDBC) and query language (SQL)
  - Syntactic challenge:
    - Consumer needs to know beforehand what driver implementation is being used (Oracle, Microsoft SQL?)
    - Each database vendor provides own product
  - Semantic challenge:
    - Need detailed understanding of database schemata and table definitions for integration

Data Sharing Problem: Syntax vs Semantics

- XML
  - Syntax problem
    - Makes sharing even more easier. Language conforms to well-defined syntax that is compatible with widely used parsers.
    - Provides effective solution to syntax problem for data sharing
  - Semantic problem
    - XML terms have no meaning by themselves
    - Minor changes like reordering two tags could create more work for developers
    - No way to encode the meaning of the relationships between XML elements

Resource Description Framework (RDF)

- Designed to represent knowledge in a distributed world
- Knowledge
  - Designed for knowledge, and not data, means RDF is particularly concerned with meaning.
  - Everything mentioned in RDF means something-- may be a reference to something in the world, like a person or movie, or it may be an abstract concept, like the state of being friends with someone else.
  - By putting three entities together (subject, predicate, object), the RDF standard says how to arrive at a fact
    - The meaning of the triple "(John, the state of being friends, Bob)" might be that John and Bob are friends.
  - Standards built on top of RDF, including RDFS and OWL, add to RDF semantics for drawing logical inferences from data.

Resource Description Framework (RDF)

- Distributed World
  - Works well for distributed information
  - RDF applications can put together RDF files posted by different people around the Internet and easily learn from them new things that no single document asserted.
  - Links documents together by the common vocabularies they use
  - Allows any document to use any vocabulary – flexibility in expressing facts about a wide range of things

RDF

- Describes relationships between resources
- Represents data in triples: subject, predicate, object
- Analogous to English grammar:
  - Subject: thing that statement describes
  - Predicate: relationship between the subject and object
  - Object: modified by the predicate

RDF vs XML

- RDF is designed to represent *knowledge* in a *distributed* world.
  - concerned with meaning
  - concerned with sharing
- XML designed to address syntax problem.
  - XML nodes don't need to be associated with particular concepts
  - XML standard doesn't indicate how to derive a fact from a document.
  - You could develop your own standard on top of XML that says how to derive the set of facts in an XML document, but reinvented something like RDF.

RDF

Andrew knows Matt.

Andrew's surname is Perez-Lopez.

Matt knows John.

Ryan works with John.

How would you represent this as a graph?

RDF

- Nodes of RDF graph represented are the subjects and the objects of the statements
- Edges are the predicates (properties)
- Two kinds of nodes: resources and literals
- Literals represent concrete data values like numbers or strings and cannot be the subjects of statements, only the objects
- Resources
  - represent everything else and can be either subjects or objects
  - anything that can be named—represents object, act or concept

RDF

- Why graphs?
  - Good way of representing relationships
  - Intended to serve as a description language for data on the WWW and other networks
    - Information stored and managed in decentralized ways
    - Using graphs makes it easy to combine RDF data from multiple sources
    - Example:
      - RDF graphs from the Facebook can be joined with RDF graphs in LinkedIn.
      - Creates a bigger graph that can be mapped to one another.
  - XML has a tree structure. Merging two trees is not an easy task.

RDF

- Triple <subject><predicate><object>
  - is powerful tool for information integration
  - Collections of URIs and literals
  - Each URI and literal have global scope
- RDF statements need no translation when moving from one system to another
  - Valid in any context
  - Self-contained assertions of information
  - Independent from one-another

RDF Resources

- RDF uses URIs to describe resources
- URL is subset of URI
  - Used for identifying web addresses.
  - Can be used as identifiers in RDF documents
- Reason to use URIs in RDF
  - Clearly distinguish resources from each other
  - Exchange information about many different objects: books, places, people, publishing houses, events, relationships and other abstract concepts

RDF Resources

- Every statement with a named resource is uniquely named regardless of where it is stored or asserted
- Comparing to database
  - Particular row with a primary key is unique to one table within one database
  - URI is a name that is universally unique and valid in any context
- Provides portability of information

URI

scheme:[//*authority*]*path*[?*query*][#*fragment*]

- *scheme*: the name of a URI scheme that classifies the type of URI (http, ftp, mailto, file, irc)
- *authority*: typically domain name (example.org, john@example.com)
- *path*: organized hierarchically using '/' as separator (/etc/passwd)
- *query*: provides parameters (q=Semantic+Web+book)
- *fragment*: second level of identifying resources recognized by the preceding #. often used to address a sub-part of a retrieved resource such as section in an HTML file.

Example

- Class website

RDF


http://semwebprogramming.net/
people#Ryan
http://semwebprogramming.net/2008/06/ont/foaf-extension#worksWith
http://semwebprogramming.net/
people#John
http://xmlns.com/foaf/0.1/knows
Perez-
Lopez
http://semwebprogramming.net/
people#Matt
http://xmlns.com/foaf/0.1surName
http://xmlns.com/foaf/0.1k/nows
http://semwebprogramming.net/
people#Andrew

RDF Literals

- Concrete data values such as numbers, times, or truth value
- Value of every literal is generally described by a sequence of characters such as a string.
- Interpretation of such sequences determined based on a given datatype
- Knowing datatype is crucial for understanding intended meaning

RDF Predicates

- Properties or predicates: represent the connections between resources
- Predicates are also resources represented with URIs

Triple Syntax: Turtle

<http://semwebprogramming.net/people#Ryan> <http://semwebprogramming.net/2008/06/ont/foaf-extension#worksWith> <http://semwebprogramming.net/people#John> .

<http://semwebprogramming.net/people#Matt> <http://xmlns.com/foaf/0.1/knows> <http://semwebprogram
people#John> .

<http://semwebprogramming.net/people#Andrew> <http://xmlns.com/foaf/0.1/knows> <http://semwebprogramming.net/people#Matt> .

<http://semwebprogramming.net/people#Andrew> <http://xmlns.com/foaf/0.1/surName> “Perez-Lopez” .

Resources represented in angular brackets

End of statement represented by a '.'

Literals represented in quotes

Triple Syntax

- Inserting the full URI every time takes up too much space
- Using namespaces lets us abbreviate the URI
- URIs can be abbreviated using prefixes

Triple Syntax

@prefix people: <http://semwebprogramming.net/people> .

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

@prefix ext: <http://semwebprogramming.net/2008/06/ont/foaf-extension#> .

people:Ryan ext:worksWith people:John .

people:Matt foaf:knows people:John .

people:Andrew foaf:knows people:Matt .

people:Andrew foaf:surName “Perez-Lopez” .

Triple Syntax

```
@prefix people: <http://semwebprogramming.net/people> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ext: <http://semwebprogramming.net/2008/06/ont/foaf-extension#> .
people:Ryan ext:worksWith people:John .
people:Matt foaf:knows people:John .
people:Andrew foaf:knows people:Matt ;
              foaf:surName "Perez-Lopez" .
```

Semicolon after first line terminates the triple and at the same time fixes the subject for the next triple.
This allows us to write many triples for one subject without repeating the name of the subject

Triple Syntax

```
@prefix people: <http://semwebprogramming.net/people> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ext: <http://semwebprogramming.net/2008/06/ont/foaf-extension#> .
people:Ryan ext:worksWith people:John .
people:Ryan ext:worksWith people:Matt .
people:Matt foaf:knows people:John .
people:Andrew foaf:knows people:Matt ;
                    foaf:surName “Perez-Lopez .
```

Triple Syntax

```
@prefix people: <http://semwebprogramming.net/people> .
    @prefix foaf: <http://xmlns.com/foaf/0.1/> .
    @prefix ext: <http://semwebprogramming.net/2008/06/ont/foaf-extension#> .
people:Ryan ext:worksWith people:John, people:Matt .
people:Matt foaf:knows people:John .
people:Andrew foaf:knows people:Matt ;
                    foaf:surName “Perez-Lopez” .
```

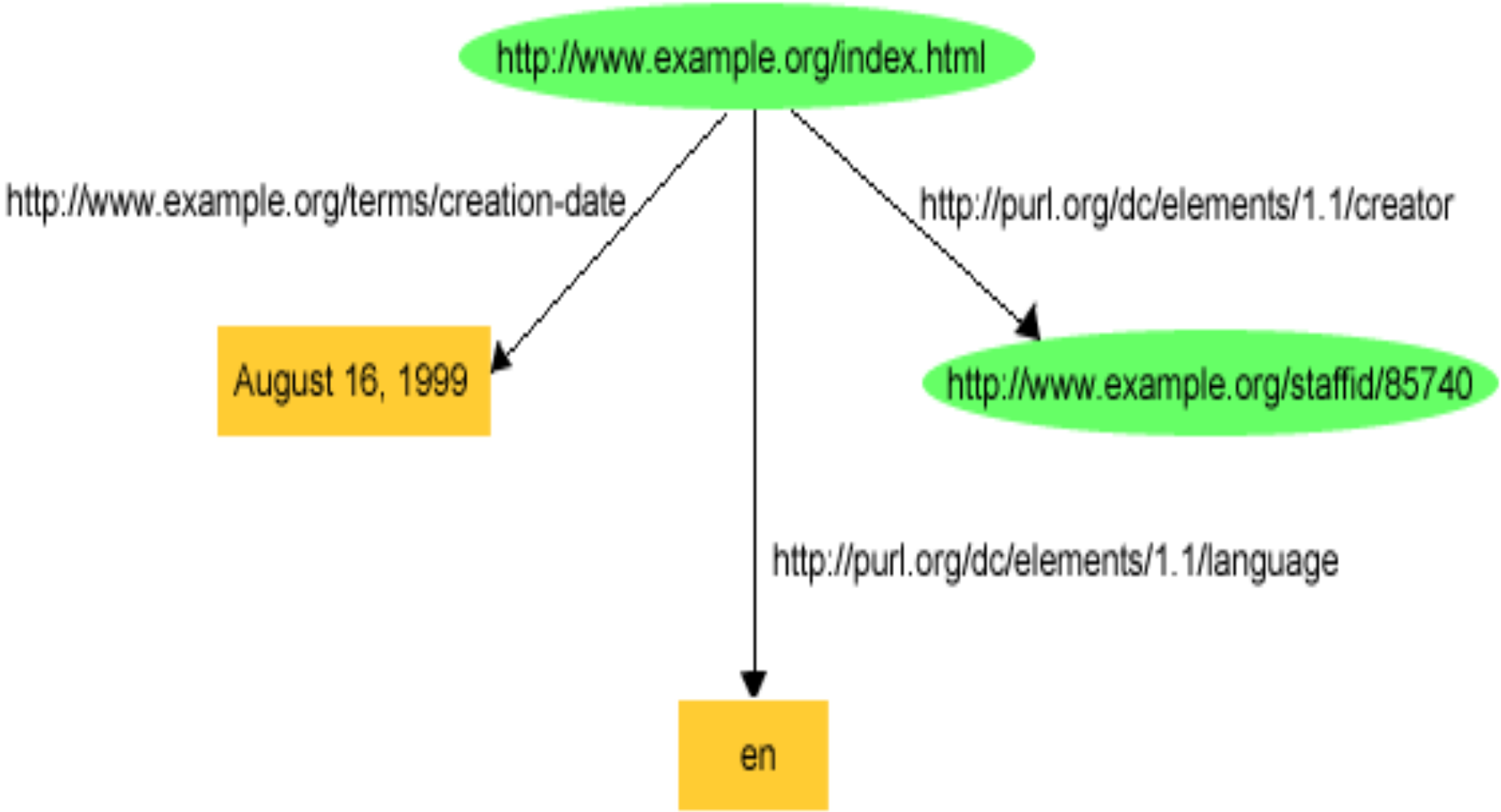
Comma finishes the triple but both subject and predicate are re-used for the next triple. So this Really represents two triples.

Examples

Example: Convert Turtle to Graph

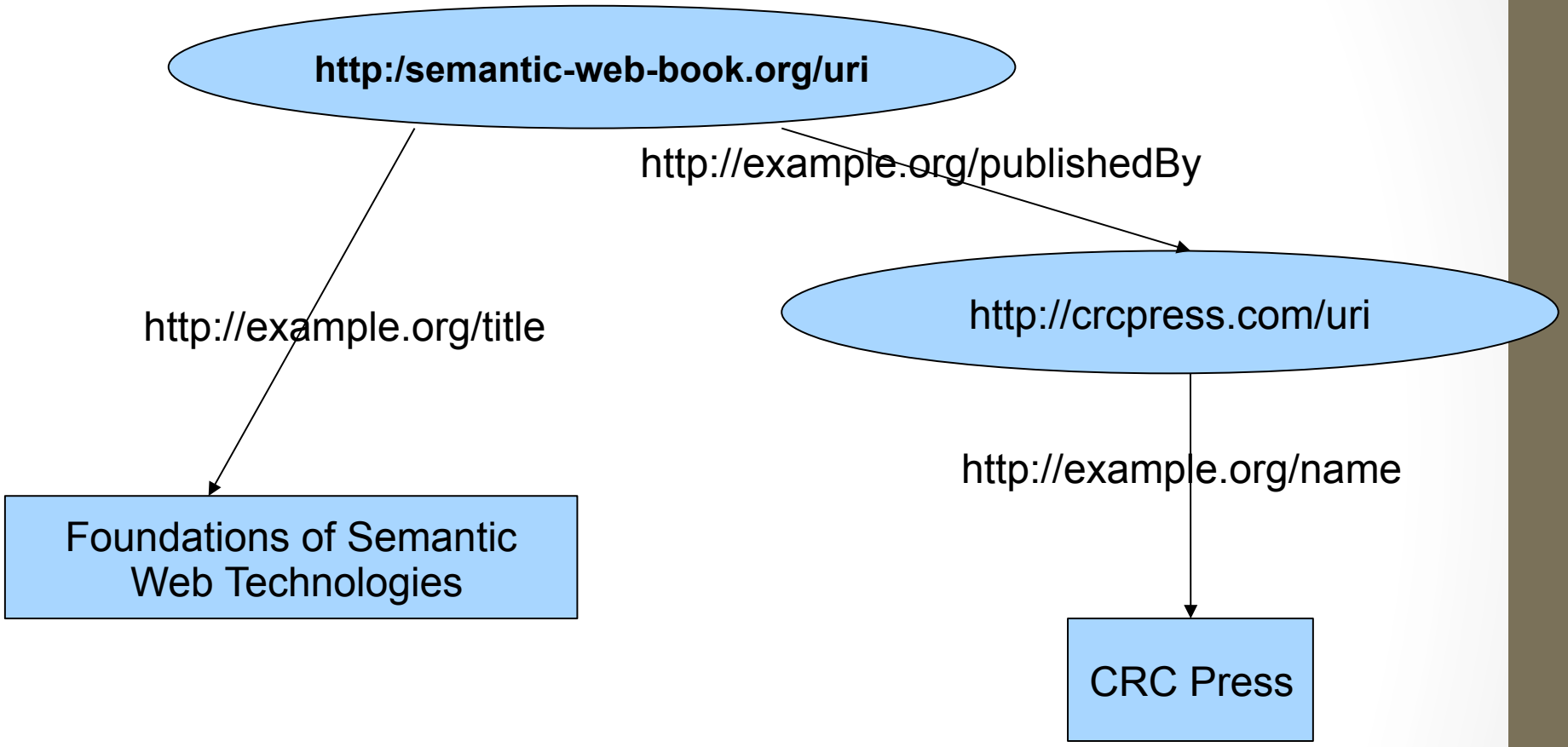
```
<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/
creator> <http://www.example.org/staffid/85740> .

<http://www.example.org/index.html> <http://www.example.org/terms/
creation-date> "August 16, 1999"
.

<http://www.example.org/index.html> <http://purl.org/dc/elements/1.1/
language> "en" .
```

Example



RDF Primer, http://www.w3.org/TR/rdf-primer

Exercise: Convert Graph to Turtle



Example

```
@prefix book: <http://semantic-web-book.org/> .
@prefix ex: <http://example.org/> .
@prefix crc: <http://crcpress.com/> .

book:uri      ex:publishedBy      crc:uri ;
              ex:title            “Foundations of Semantic Web Technologies” .
crc:uri       ex:name             “CRC Press” .
```

Convert Turtle to Graph

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix contact: <http://www.w3.org/2000/10/swap/pim/contact#>.

<http://www.w3.org/People/EM/contact#me> rdf:type contact:Person;
contact:fullName "Eric  Miller";

contact:mailbox <mailto:em@w3.org>;
contact:personalTitle "Dr.".
```

RDF

RDF Primer, http://www.w3.org/TR/rdf-primer

XML Syntax for RDF

Turtle is easy to read and write

- But XML is the basis for data transfer on the web
- There's a lot of tool (and programming library) support for XML
- Hence, the main syntax for RDF is XML-based.
- Turtle is not a W3C recommendation
- The normative syntax for RDF is its XML syntax

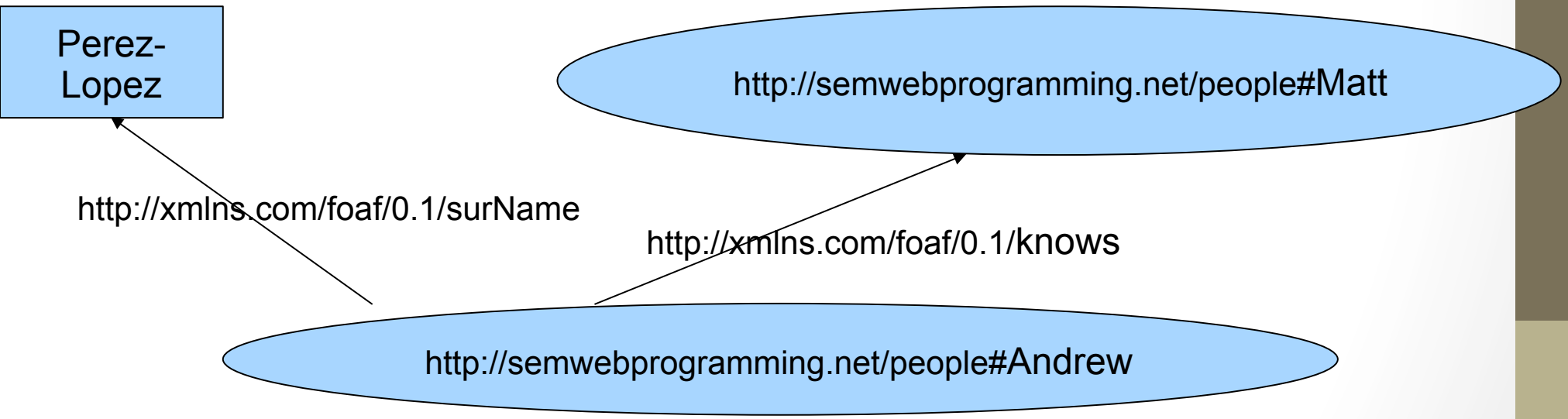
RDF Statements

- Statements about resources are grouped into <rdf:Description> elements
- Each description element has an rdf:about attribute which gives the subject of all of the statements within it
- Each subsequent elements within the description defines predicate and object of a statement
- Name of internal tags represents the predicate of a statement
- Object is represented differently depending on whether it is a resource or a literal

```
<rdf:Description rdf:about="subject">
    <predicate rdf:resource="object" />
    <predicate>literal value </predicate>
</rdf:Description>
```

RDF Statements

```
<rdf:Description rdf:about=”http://semwebprogramming.net/people#Andrew”>
   <foaf:surname>Perez-Lopez</foaf:surname>
   <foaf:knows rdf:resource=”http://semwebprogramming.net/people#Matt”/>
</rdf:Description>
```



XML Syntax for RDF

```
<rdf:RDF
   xmlns:people=”http://semwebprogramming.net/people#”
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:foaf="http://xmlns.com/foaf/0.1/"
   xmlns:ext="http://semwebprogramming.net/2008/06/ont/foaf-extension#">

   <rdf:Description rdf:about="http://semwebprogramming.net/people#Ryan">
      <ext:worksWith
         rdf:resource="http://semwebprogramming.net/people#John"/>
   </rdf:Description>

   <rdf:Description rdf:about="http://semwebprogramming.net/people#Matt">
      <foaf:knows
         rdf:resource="http://semwebprogramming.net/people#John"/>
   </rdf:Description>

    <rdf:Description rdf:about="http://semwebprogramming.net/people#Andrew">
     <foaf:surname>Perez-Lopez</foaf:surname>
      <foaf:knows
            rdf:resource="http://semwebprogramming.net/people#Matt"/>
   </rdf:Description>
</rdf:RDF>
```