RDF, RDFS and OWL: Graph Data Models for the Semantic Web
Semantic Web: The Idea

- **Semantic Web**
  - *does not mean* computers are going to understand the meaning of everything on the web
  - does mean that we can create some standards ways of representing web-information such the logical pieces of meaning can be mechanically manipulated by programs (applications) in a generic way
  - One such mechanism involves representing, manipulating and using metadata
Linked Data is about using the Web to connect related data that wasn't previously linked, or using the Web to lower the barriers to linking data currently linked using other methods.
Is there any reason to connect these two pages?
What is RDF?

- RDF (Resource Description Framework) was originally designed to develop metadata for the web
- Example: my SDSC web page

http://www.sdsc.edu/~gupta

The individual whose name is Amarnath Gupta, email <gupta@sdsc.edu>, is the creator of http://www.sdsc.edu/~gupta
Another Example

A book website

Now a program can “integrate” information from the two sites correctly (i.e., by URI and not by value)
## URIs

<table>
<thead>
<tr>
<th>Name</th>
<th>Definition</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>a character string that specifies where a known resource is available on the Internet and the mechanism for retrieving it</td>
<td>scheme://username:password@domain:port/path?query_string#fragment_id</td>
</tr>
<tr>
<td>URI</td>
<td>A character string used to identify a name or a resource on the Internet</td>
<td>ftp://example.org/resource.txt/relative/URI/with/absolute/path/to/resource.txt</td>
</tr>
<tr>
<td>URN</td>
<td>A character string that designates the universal name of something. URNs usually use namespaces</td>
<td>urn:isan:0000-0000-9E59-0000-O-0000-0000-2</td>
</tr>
</tbody>
</table>
Data Types in RDF

- A data type is:
  - A set of values called the *value space*
  - A set of character strings called the *lexical space*
  - A lexical to value mapping

- RDF has no built-in set of data types of its own
  - In RDF, one uses typed literals and explicitly indicates what datatype should be used to interpret it
  - RDF takes many of its data types from XML Schema
    - `xsd:string`, `xsd:boolean`, `xsd:date`
    - `ucsd:85740exterms:age “31”^^xsd:integer`
    - `ucsd:85740 exterms:age “pumpkin”^^xsd:integer`

Valid and correct

Valid but incorrect
## Representing the RDF graph (N-Triples)

- **Representing as a Collection of Edges**

<table>
<thead>
<tr>
<th>Subject</th>
<th>Predicate</th>
<th>Object</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://www.morganclaypool.com/doi/abs/10.2200/S00374ED1V01Y201107DTM019">http://www.morganclaypool.com/doi/abs/10.2200/S00374ED1V01Y201107DTM019</a></td>
<td>dcterms:publisher</td>
<td>:_anon1</td>
</tr>
<tr>
<td>:_anon1</td>
<td>name</td>
<td>Morgan &amp; Claypool</td>
</tr>
</tbody>
</table>
Representing the RDF graph (XML)

- Representing as Node Properties

```xml
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:edu="http://www.example.org/">

  <rdf:Description rdf:about="http://www.morganclaypool.com/doi/abs/10.2200/S00374ED1V01Y201107DTM019">
    <dcterms:title>Managing Event Information: Modeling, Retrieval, and Applications</dcterms:title>
    <dcterms:creator rdf:resource="http://www.sdsc.edu/~gupta"/>
    <dcterms:publisher rdf:resource=":_anon1"/>
  </rdf:Description>

  ...
```
Using Data Types in RDF/XML

- A typical data record

```xml
<rdf:Description rdf:ID="item10245"/>
  <exterms:model rdf:datatype="&xsd:string">Overnighter</exterms:model>
  <exterms:sleeps rdf:datatype="&xsd:integer">2</exterms:sleeps>
  <exterms:weight rdf:datatype="&xsd:decimal">2.4</exterms:weight>
  <exterms:packedSize rdf:datatype="&xsd:integer">784</exterms:packedSize>
</rdf:Description>
```
RDF and Nested Relations

- Consider a relation Product with a tuple-valued attribute

<table>
<thead>
<tr>
<th>Name</th>
<th>ID</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC</td>
<td>123</td>
<td>34.57</td>
</tr>
</tbody>
</table>

The RDF way

Product:123
Product:123
_product1
_product1
name
weight
rdf:value
unit

ABC
_product1
“34.57”^^xsd:decimal
kg
RDF “Collections”

- RDF Collection

```xml
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:s="http://example.org/students/vocab#">
  <rdf:Description rdf:about="http://example.org/courses/6.001">
    <s:students rdf:parseType="Collection">
      <rdf:Description rdf:about="s:student/Amy"/>
      <rdf:Description rdf:about="s:student/Mohamed"/>
      <rdf:Description rdf:about="s:student/Johann"/>
    </s:students>
  </rdf:Description>
</rdf:RDF>
```

- The `s:students` node
  - Has the built-in type `rdf:List`
  - Is a named **closed** collection
RDF “Collections” is a List

```xml
<rdf:Description rdf:about="http://example.org/courses/6.001">
  <s:students rdf:nodeID="sch1"/>
</rdf:Description>

<rdf:Description rdf:nodeID="sch1">
  <rdf:rest rdf:nodeID="sch2"/>
</rdf:Description>

<rdf:Description rdf:nodeID="sch2">
  <rdf:rest rdf:nodeID="sch3"/>
</rdf:Description>

<rdf:Description rdf:nodeID="sch3">
  <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
</rdf:Description>
```
RDF “Containers”

- `rdf:bag` – a named unordered collection

container membership properties have names of the form `rdf:_n`

Yes, it is still unordered
RDF “Containers”

- rdf:alt – represents alternatives (or choices) in a set

- The semantics of rdf:alt
  - An Alt container is intended to have at least one member, identified by the property rdf:_1.
  - This member is intended to be considered as the default or preferred value.
Reification

- Sometimes we want to make a statement about a statement
  - “John’s hair-color is black” says his passport.
- The main statement as a triple
  - (John hair-color black)
- Now, treat it as an entity called s100
  - s100:(John hair-color black)
- Next, make a statement about st
  - (John-passport makes-claim s100)
- The process is called reification
  - executed through rdf:statement

<table>
<thead>
<tr>
<th>s100</th>
<th>rdf:type</th>
<th>rdf:Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>s100</td>
<td>rdf:subject</td>
<td>URI(John)</td>
</tr>
<tr>
<td>S100</td>
<td>rdf:predicate</td>
<td>URI(hair-color)</td>
</tr>
<tr>
<td>s100</td>
<td>rdf:object</td>
<td>URI(black)</td>
</tr>
<tr>
<td>URI(John)</td>
<td>URI(hair-color)</td>
<td>URI(black)</td>
</tr>
<tr>
<td>S100</td>
<td>makes-claim</td>
<td>URI(John-passport)</td>
</tr>
</tbody>
</table>
RDF Schema (RDFS)

- RDF Schema provides a *type system* for RDF
  - Influenced by object-oriented languages

- Classes
  - any resource having an rdf:type property whose value is the resource rdfs:Class.

- Instance definition

  | ex:myVehicle | rdf:type | ex:MotorVehicle |

- Subclass definition

  | ex:Van | rdf:type | rdfs:Class |
  | ex:Van | rdfs:subClassOf | ex:MotorVehicle |

Transitive property?

A class may be a subclass of *multiple classes*
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
xml:base="http://example.org/schemas/vehicles">
  <rdfs:Class rdf:ID="MotorVehicle"/>
  <rdfs:Class rdf:ID="PassengerVehicle">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Truck">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="Van">
    <rdfs:subClassOf rdf:resource="#MotorVehicle"/>
  </rdfs:Class>
  <rdfs:Class rdf:ID="MiniVan">
    <rdfs:subClassOf rdf:resource="#Van"/>
    <rdfs:subClassOf rdf:resource="#PassengerVehicle"/>
  </rdfs:Class>
</rdf:RDF>
## Properties of Properties in RDFS

<table>
<thead>
<tr>
<th>Property name</th>
<th>Comment</th>
<th>Domain</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf:type</td>
<td>The subject is an instance of a class.</td>
<td>rdfs:Resource</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:subClassOf</td>
<td>The subject is a subclass of a class.</td>
<td>rdfs:Class</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:subPropertyOf</td>
<td>The subject is a subproperty of a property.</td>
<td>rdf:Property</td>
<td>rdf:Property</td>
</tr>
<tr>
<td>rdfs:domain</td>
<td>A domain of the subject property.</td>
<td>rdf:Property</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:range</td>
<td>A range of the subject property.</td>
<td>rdf:Property</td>
<td>rdfs:Class</td>
</tr>
<tr>
<td>rdfs:label</td>
<td>A human-readable name for the subject.</td>
<td>rdfs:Resource</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>rdfs:comment</td>
<td>A description of the subject resource.</td>
<td>rdfs:Resource</td>
<td>rdfs:Literal</td>
</tr>
<tr>
<td>rdfs:member</td>
<td>A member of the subject resource.</td>
<td>rdfs:Resource</td>
<td>rdfs:Resource</td>
</tr>
<tr>
<td>rdfs:isDefinedBy</td>
<td>The definition of the subject resource.</td>
<td>rdfs:Resource</td>
<td>rdfs:Resource</td>
</tr>
</tbody>
</table>
RDFS Subproperties

- R1 is a property with domain X and range Y
- R2 is a subproperty R1 if
  - (a R2 b) implies (a R1 b)
  - typeOf(a) ≤ X, typeOf(b) ≤ Y
- close_friend_of ≤ friend_of ≤ related_to ≤ knows
- supervisor_of ≤ colleague_of ≤ works_with ≤ knows
RDFS Inferences

- **Existential Rule.** If $\mu : G \rightarrow G$ is a mapping,
  - then $G \Rightarrow G$

- **Typing Rules.** These are type inference rules
  1. $(a \text{ dom } c), (a \times y) \Rightarrow (x \text{ type } c)$
  2. $(a \text{ range } d), (x a y) \Rightarrow (y \text{ type } d)$

- **Subclass Rules.** $sc$ is reflexive and transitive
  1. $(a \text{ type class}) \Rightarrow (a \text{ sc } a)$
  2. $(a \text{ sc } b), (b \text{ sc } c) \Rightarrow (a \text{ sc } c)$
  3. $(a \text{ sc } b), (x \text{ type } a) \Rightarrow (x \text{ type } b)$

- **Subproperty Rules.** $sp$ is reflexive and transitive
  1. $(a \text{ type prop}) \Rightarrow (a \text{ sp } a)$
  2. $(a \text{ sp } b), (b \text{ sp } c) \Rightarrow (a \text{ sp } c)$
  3. $(a \text{ sp } b), (x a y) \Rightarrow (x b y)$
What RDF/RDFS Cannot Model

- cardinality constraints on properties
  - a Person has exactly one biological father.
- a given property (such as ex:hasAncestor) is transitive
- two different properties are inverses of each other
- a given property is a unique identifier (or key) for instances of a particular class.
- two different classes (having different URIrefs) actually represent the same class.
- two different instances (having different URIrefs) actually represent the same individual.
- constraints on the range or cardinality of a property that depend on the class of resource to which a property is applied,
  - for a soccer team the ex:hasPlayers property has 11 values, while for a basketball team the same property should have only 5 values.
- describe new classes in terms of combinations (e.g., unions and intersections) of other classes, or to say that two classes are disjoint (i.e., that no resource is an instance of both classes).
OWL: The Web Ontology Language

- Classes and Individuals

An individual

```
<rdf:Description rdf:about="#William_Jefferson_Clinton"/>
```

- A class can be described by
  - a class identifier (a URI reference)
  - an exhaustive enumeration of individuals that together form the instances of a class
  - a property restriction
  - the intersection, union, complement of two or more class descriptions

- Properties

  - *Object properties* link individuals to individuals, or classes to classes, or classes to individuals
  - *Datatype properties* link individuals to data values.
Class Properties in OWL

- OWL adds a number of modeling elements
  - oneOf

```xml
<owl:Class>
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#NorthAmerica"/>
    <owl:Thing rdf:about="#SouthAmerica"/>
  </owl:oneOf>
</owl:Class>
```

- Existential and Universal Quantification

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:someValuesFrom rdf:resource="#Physician" />
</owl:Restriction>
```

```xml
<owl:Restriction>
  <owl:onProperty rdf:resource="#hasParent" />
  <owl:allValuesFrom rdf:resource="#Musician" />
</owl:Restriction>
```
Classes as Logical Entities

Can also use:

```xml
<owl:unionOf rdf:parseType="Collection">
</owl:unionOf>
```
More Logical Definitions

- Classes defined through negation

- Class Axioms

```xml
<owl:Class rdf:ID="Operetta">
  <rdfs:subClassOf rdf:resource="#MusicalWork"/>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#hasLibrettist" />
      <owl:minCardinality rdf:datatype="xsd:nonNegativeInteger">1</owl:minCardinality>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Class>
      <owl:complementOf rdf:resource="#Opera"/>
    </owl:Class>
  </rdfs:subClassOf>
</owl:Class>
```

- Necessary but not sufficient condition
  - Owl:equivalentClassOf is used for necessary and sufficient conditions
Memory-Based Storage

- Jena in-memory architecture
Main Memory BitMaps

- **BitMat**
  - Uses the observation that the no. of predicates is small
Storage of RDF/OWL Data

- **Triple Store Approach**
  - Storing an edge table, often with an additional ID field
    - Can be structured as a B+ tree
    - Literals compressed with a mapping dictionary
      - Strings are assigned IDs, and IDs are used in the triple stores
    - Often additional indices are stored for each permutation of the three columns and column pairs
      - SPO, SOP, OSP, OPS, PSO, POS
      - SP, SO, PO, PS, OP, OS
  - Typical issue
    - Too many self-joins for queries
      - Indexes reduce the number of joins
Storage of RDF/OWL Data

- Using Relational back-ends
  - Property Tables Approach
Storage of RDF/OWL Data

- Property Tables Approach (contd.)

Minerva System

There are many hybrid approaches.
OWL is a Semantic Graph

- One can view OWL documents as a graph with special node and edge semantics.

How do we query this graph?