The Web of Linked Data

SPARQL & THE LINKED DATA PROJECT
Semantic Web components

- The Web of linked data
The Web of linked data

Semantic Web

Web of Data

Linked Data
(HTTP URIs, RDF/XML, RDFa)

Vocabularies
(FOAF, SIOC, etc.)

microformats+GRDDL

Reasoning

Rules
(Prolog, RIF, SWRL, etc.)

Ontologies
(DL, KIF, etc.)
SPARQL BASICS
SPARQL Protocol and RDF Query Language (SPARQL)

- Simple protocol for querying remote databases over HTTP
  - SPARQL 1.0: W3C Recommendation January 15th, 2008
  - SPARQL 1.1: W3C Recommendation March 21st, 2013
- SPARQL queries RDF graphs
  - An RDF graph is a set of triples
- SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware
SPARQL and RDF

- It is the triples that matter, not the serialization
  - RDF/XML is the W3C recommendation but it is not a good choice because it allows multiple ways to encode the same graph
- SPARQL uses the Turtle syntax, an N-Triples extension
SPARQL and RDF

N3: "Notation 3" - extensive formalism

N-Triples: part of N3

Turtle: Extension of N-Triples (shortcuts)
Turtle - Terse RDF Triple Language

- A serialization format for RDF
- A subset of Tim Berners-Lee and Dan Connolly’s Notation 3 (N3) language
  - Unlike full N3, doesn’t go beyond RDF’s graph model
- A superset of the minimal N-Triples format
- Turtle has no official status with any standards organization, but has become popular amongst Semantic Web developers as a human-friendly alternative to RDF/XML
Example

http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName

Eric Miller
mailto:em@w3.org

http://www.w3.org/2000/10/swap/pim/contact#mailbox

Dr.

http://www.w3.org/2000/10/swap/pim/contact#personalTitle
"Turtle" notation

<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/2000/10/swap/pim/contact#fullName>  
"Eric Miller" .

<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/2000/10/swap/pim/contact#mailbox>  
<mailto:em@w3.org> .

<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/2000/10/swap/pim/contact#personalTitle>  
"Dr." .

<http://www.w3.org/People/EM/contact#me>  
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>  
<http://www.w3.org/2000/10/swap/pim/contact#Person> .
Turtle

• Simple triple:
subject predicate object .

:john rdf:label "John" .

• Grouping triples:
subject predicate object ; predicate object ...
Prefixes

• Mechanism for namespace abbreviation

```
@prefix abbr: <URI>
```

• Example:

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

• Default:

```
@prefix : <URI>
```

• Example:

```
@prefix : <http://example.org/myOntology#>
```
Identifiers

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

- **Qnames (Qualified names)**
  - `namespace-abbr?:localname`

- **Literals**
  - "string"(@lang)?(^type)?
    - "John"
    - "Hello"@en-GB
    - "1.4"^^xsd:decimal
Blank nodes

http://www.example.org/staffid/85740

http://www.example.org/terms/address

http://www.example.org/terms/city

Bedford

http://www.example.org/terms/street

1501 Grant Avenue

http://www.example.org/terms/postalCode

01730

http://www.example.org/terms/state

Massachusetts
Blank nodes

• Simple blank node:
  - [] or _:x

  :john ex:hasFather [] .
  :john ex:hasFather _:x .

• Blank node as subject:
  - [ predicate object ; predicate object object ... ] .

  [ ex:hasName "John"] .
  [ ex:authorOf :lotr ;
    ex:hasName "Tolkien"] .
Collections

• ( object1 ... objectn )

:doc1 ex:hasAuthor (:john :mary) .

• Short for

:doc1 ex:hasAuthor

[ rdf:first :john;
  rdf:rest [ rdf:first :mary;
  rdf:rest rdf:nil ]
] .
Example

@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntaxns# .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix : <http://example.org/#> .

<http://www.w3.org/TR/rdf-syntax-grammar>
  dc:title "RDF/XML Syntax Specification (Revised)" ;
  :editor [
    :fullName "Dave Beckett";
    :homePage <http://purl.org/net/dajobe/>
  ] .
RDF Triple stores

- Specialized databases for RDF triples
- Can ingest RDF in a variety of formats
- Support a query language (SPARQL)
<table>
<thead>
<tr>
<th>Relational database</th>
<th>RDF Triple store</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Data model</strong></td>
<td><strong>Data model</strong></td>
</tr>
<tr>
<td>– Relational data (tables)</td>
<td>– RDF graphs</td>
</tr>
<tr>
<td><strong>Data instances</strong></td>
<td><strong>Data instances</strong></td>
</tr>
<tr>
<td>– Records in tables</td>
<td>– RDF triples</td>
</tr>
<tr>
<td><strong>Query support</strong></td>
<td><strong>Query support</strong></td>
</tr>
<tr>
<td>– SQL</td>
<td>– SPARQL</td>
</tr>
<tr>
<td><strong>Indexing mechanisms</strong></td>
<td><strong>Indexing mechanisms</strong></td>
</tr>
<tr>
<td>– Optimized for evaluating queries as relational expressions</td>
<td>– Optimized for evaluating queries as graph patterns</td>
</tr>
</tbody>
</table>
SPARQL

- Uses SQL-like syntax

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
SELECT ?title
```

- Prefix mechanism to abbreviate URIs
- Variables to be returned
- Query pattern (list of triple patterns)
- FROM
- Name of the graph
SELECT

- Variables selection
- Variables: \( ?\text{string} \)

- Syntax: \( \text{SELECT } \var_1, \ldots, \var_n \)

SELECT \( ?\text{name} \)
SELECT \( ?x, ?\text{title} \)
WHERE

- Graph patterns to match
- Set of triples
  
  \[
  \{ (subject \ predicate \ object \ .)* \}
  \]
- Subject: URI, QName, Blank node, Literal, Variable
- Predicate: URI, QName, Blank node, Variable
- Object: URI, QName, Blank node, Literal, Variable
Graph patterns

• The pattern contains unbound symbols
• By binding the symbols (if possible), subgraphs of the RDF graph are selected
• If there is such a selection, the query returns the bound resources
Graph patterns

• (subject, ?p, ?o)
  – ?p and ?o are “unknowns”
Graph patterns

- The triplets in WHERE define the graph pattern, with ?p and ?o “unbound” symbols
- The query returns the list of matching p,o pairs
Example 1

SELECT ?cat, ?val
  ?x category ?cat }

• Returns:

[["Total Members",100],["Total Members",200],..., 
["Full Members",10],...]
Example 2

```
SELECT ?cat, ?val
    ?x category ?cat .
    FILTER(?val>=200) .}
```

- Returns:

```
[["Total Members",200],...]
```
Example 3

SELECT ?cat, ?val, ?uri
WHERE {
    ?x category ?cat.
    ?al contains ?x.
    ?al linkTo ?uri
}

- Returns:

```[
[["Total Members",100,http://...]],...
]```
Example 4

SELECT ?cat, ?val, ?uri
   ?x category ?cat.
OPTIONAL ?al contains ?x.
   ?al linkTo ?uri }

• Returns:

[["Total Members",100,http://...], ..., 
["Full Members",20, [],...]]
Other SPARQL Features

• Limit the number of returned results
• Remove duplicates, sort them,…
• Specify several data sources (via URIs) within the query (essentially, a merge)
• Construct a graph combining a separate pattern and the query results
• Use datatypes and/or language tags when matching a pattern
Other SPARQL Features

• Query types
  – Select: projections of variables and expressions
  – Construct: create triples (or graphs) based on query results
  – Ask: whether a query returns results (result is true/false)
  – Describe: describe resources in the graph
SPARQL use in practice

• Where to find meaningful RDF data to search?

• The Linked Data Project
THE LINKED DATA PROJECT
The Linked Data Project

• A fundamental prerequisite of the Semantic Web is the existence of large amounts of meaningfully interlinked RDF data on the Web

• Linked Data is not a specification, but a set of best practices for providing a data infrastructure that makes it easier to share data across the web

• You can then use semantic web technologies such as RDFS, OWL, and SPARQL to build applications around that data
The four principles of Linked Data

1. Use URIs as names for things
   - URIs are the best way available to uniquely identify things, and therefore to identify connections between things

2. Use HTTP URIs so that people can look up those names
   - You may have seen URIs that begin with ftp:, mailto:, or prefixes made up by a particular community, but using these other ones reduces interoperability, and interoperability is what it’s all about
The four principles of Linked Data

3. When someone looks up a URI, provide useful information, using the standards (RDF/RDFS, SPARQL)
   – A URI can just be a name and not actually the address of a web page, but this principle says that you may as well put something there
   – It can be an HTML web page, or something else; whatever it is, it should use a recognized standard
   – RDFS and OWL let you spell out a list of terms and information about those terms and their relationships in a machine-readable way—readable, for example, by SPARQL queries
   – Because of this, if a URI that identifies a resource leads to RDFS or OWL declarations about that resource, this is a big help to applications.
The four principles of Linked Data

4. Include links to other URIs so that they can discover more things
   – Imagine if none of the original HTML pages had a elements to link them to other pages: it wouldn’t have been much of a web
   – In addition to the HTML linking element, various RDF vocabularies provide other properties as a way to say “this data (or this element of data) has a specific relationship to another resource on the web.”
   – When applications can follow these links, they can do interesting new things
Linked Data example
Linked Data example
Linked Data example

**bestbuy.com**

- provides data about product offers (price, delivery time, etc.)

**Apple Inc.**

- provides data about product features (CPU, RAM, HD, etc.)

**DBPedia**

- provides general data about products or companies

**UN environmental indicators**

- provides statistical data, such as carbon footprint, waste, energy usage, etc.

---

Vocabularies used:
- GoodRelations ... http://purl.org/goodrelations/v1#
- Dublin Core ... http://purl.org/dc/terms/
Vocabularies

- Define terms (classes and properties)
- Typically RDFS or OWL family
Some commonly used RDF vocabularies

• Friend of a Friend (FOAF)
  – Defines classes and properties for representing information about people and their relationships
  – Project: http://www.foaf-project.org/
  – Namespace: http://xmlns.com/foaf/0.1/

• Description of a Project (DOAP)
  – Describes software projects
  – Project: http://trac.usefulinc.com/doap
  – Namespace: http://usefulinc.com/ns/doap#

• Dublin Core (DC)
  – Describes generic resources
  – Namespace: http://purl.org/dc/elements/1.1/
Some commonly used RDF vocabularies

• Semantically-Interlinked Online Communities (SIOC)
  – Represents content from blogs, wikis, forums, mailinglists, chats, ...
  – Project: http://sioc-project.org/
  – Namespace: http://rdfs.org/sioc/ns#

• Vocabulary of Interlinked Datasets (VoiD)
  – Describes the interlinking relationship between datasets
  – Project: http://semanticweb.org/wiki/VoiD,
    http://vocab.deri.ie/void
  – Namespace: http://rdfs.org/ns/void#

• vCard
  – Describes people and organisations
  – Project: http://www.w3.org/TR/vcard-rdf/
  – Namespace: http://www.w3.org/2006/vcard/ns#
Some commonly used RDF vocabularies

• Web Ontology Language (OWL)
  – Describes classes and relations
  – Project: http://www.w3.org/TR/owl2-overview/
  – Namespace: http://www.w3.org/ns/owl2-xml

• Simple Knowledge Organisation System (SKOS)
  – Supports the use of thesauri, classification schemes, subject heading systems and taxonomies
  – Project: http://www.w3.org/2004/02/skos/
  – Namespace: http://www.w3.org/2004/02/skos/core#
Some commonly used RDF vocabularies

• RDF Schema (RDFS)
  – Provides a data-modelling vocabulary for RDF data
  – Project: http://www.w3.org/TR/rdf-schema/
  – Namespace: http://www.w3.org/2000/01/rdf-schema#

• XML Schema Datatypes
  – Describes the properties and the content of RDF vocabularies
  – Project: http://www.w3.org/TR/xmlschema-2/
  – Namespace: http://www.w3.org/2001/XMLSchema#
Guidance

- To name things: rdfs:label, foaf:name, skos:prefLabel
- To describe people: FOAF, vCard
- To describe projects: DOAP
- To describe Web pages and other publications: dc:creator, dc:description, …
- To describe an RDF schema/vocabulary/ontology: VoiD
- To describe addresses: vCard
- To model simple data: RDF, RDFS, vocabularies
- To model existing taxonomies: SKOS
- To model complex data and/or allow for logical inference: OWL
THE LINKED OPEN DATA CLOUD
The first contributors

- **DBLP Computer science bibliography**
  - Richard Cyganiak, Chris Bizer (FU Berlin)
- **DBpedia Structured information from Wikipedia**
  - Universität Leipzig, FU Berlin, OpenLink
- **DBtune, Jamendo Creative Commons music repositories**
  - Yves Raimond (University of London)
- **Geonames World-wide geographical database**
  - Bernard Vatant (Mondeca), Marc Wick (Geonames)
- **Musicbrainz Music and artist database**
  - Frederick Giasson, Kingsley Idehen (Zitgist)
- **Project Gutenberg Literary works in the public domain**
  - Piet Hensel, Hans Butschalowsky (FU Berlin)
- **Revyu Community reviews about anything**
  - Tom Heath, Enrico Motta (Open University)
- **RDF Book Mashup Books from the Amazon API**
  - Tobias Gauß, Chris Bizer (FU Berlin)
- **US Census Data Statistical information about the U.S.**
  - Josh Tauberer (University of Pennsylvania), OpenLink
- **World Factbook Country statistics, compiled by CIA**
  - Piet Hensel, Hans Butschalowsky (FU Berlin)
September 2007
As of September 2008
September 2010
Datasets

• A dataset is a set of RDF triples that are published, maintained or aggregated by a single provider.
Linksets

- An RDF link is an RDF triple whose subject and object are described in different datasets.
- A linkset is a collection of such RDF links between two datasets.
LOD cloud statistics

triples distribution

links distribution

http://lod-cloud.net/state/
SPARQL SYNTAX: EXAMPLES ON OPEN DATASETS
Some popular data sets

- Dbpedia
- WikiData
  - http://dumps.wikimedia.org/wikidatawiki/
- GeoNames
  - http://download.geonames.org/all-geonames-rdf.zip
- LinkedGeoData
  - http://downloads.linkedgeodata.org/releases/
- Open Directory
  - http://rdf.dmoz.org/
- MusicBrainz
SPARQL endpoints

• An endpoint, also called a processor, is a service that accepts and processes SPARQL queries and returns results in different formats

• SPARQL is both a query language and a protocol
  – The language defines the query syntax
  – The protocol is used to describe how a SPARQL client (such as one accessible via a web browser) talks to a SPARQL endpoint/processor (e.g. http://dbpedia.org/sparql) both in an abstract sense and using a concrete implementation based on WSDL 2.0 (Web Service Definition Language)
SPARQL endpoints

• Accept queries and returns results via HTTP
  – Generic endpoints queries any web-accessible RDF data
  – Specific endpoints are hardwired to query against particular datasets
• The results of SPARQL queries can be returned in a variety of formats:
  – XML, JSON, RDF, HTML
  – JSON (JavaScript Object Notation): lightweight computer data interchange format; text-based, human-readable format for representing simple data structures and associative arrays
Examples of SPARQL endpoints

• Generic endpoints queries any web-accessible RDF data
  – OpenLink Virtuoso SPARQL Query Editor
    http://demo.openlinksw.com/sparql/

• DBpedia SPARQL interactive endpoints (example)
  – Virtuoso SPARQL Query Editor
    http://dbpedia.org/sparql/
  – SPARQL Explorer
    http://dbpedia.org/snorql/
  – Interactive SPARQL Query Builder by OpenLink
    http://dbpedia.org/isparql/

• Camera.it SPARQL interactive endpoints (example)
  – Virtuoso SPARQL Query Editor
    http://dati.camera.it/sparql
DBpedia

• DBpedia provides a complementary service to Wikipedia by exposing knowledge in a quality-controlled form compatible with tools covering ad-hoc structured data querying, business intelligence & analytics, entity extraction, natural language processing, reasoning & inference, machine learning services, and artificial intelligence in general
  – 130 Wikimedia projects, in particular the English Wikipedia, Commons, Wikidata and over 100 Wikipedia language editions

• Data is published strictly in line with Linked Data principles using open standards

http://wiki.dbpedia.org/OnlineAccess
DBpedia Architecture

Extraction Manager

- Extraction Job
  - Extractors: Label, Category, Image, Redirect, Disambiguation, Abstract, Geo, Pagelink, Generic Infobox, Mapping-based Infobox
- Parsers: DateTime, Units, Geo, String-List, Numbers

Destinations
- N-Triple Serializer
- SPARQL-Update Destination

PageCollections
- Article-Queue
- Database Wikipedia
- Live Wikipedia

Ontology-Mappings

Wikipedia
- Update Stream
- Dumps
- OAI-PMH

N-Triple Dumps

Triple Store
Virtuoso

SPARQL endpoint

Linked Data

The Web
- SPARQL clients
- HTML browser
- RDF browser
- DBpedia apps

Linked Data

SPARQL endpoint

The Web

HTML browser

RDF browser

DBpedia apps

SPARQL clients
DBpedia resources return XHTML or RDF through content negotiation
SPARQL query structure

• A SPARQL query includes, in order
  – Prefix declarations, for abbreviating URIs
  – A result clause, identifying what information to return from the query
  – The query pattern, specifying what to query for in the underlying dataset
  – Query modifiers: slicing, ordering, and otherwise rearranging query results
SPARQL query structure

- A SPARQL query includes, in order

```sparql
# prefix declarations
PREFIX foo: <http://example.com/resources/>
...
# result clause
SELECT ...
# query pattern
WHERE {
    ...
}
# query modifiers
ORDER BY ...
```
Example 1 – simple triple pattern

• In DBpedia
  – Find all subjects (?person) linked to :Turin with the dbo:birthplace predicate
  – Then return all the values of ?person.
• In other words, find all people that were born in Turin

PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?person
WHERE {
  ?person dbo:birthPlace :Turin .
}

• To find predefined DBpedia prefixes
  – https://dbpedia.org/sparql?nsdecl
Example 2 – simple triple pattern

• List all the places and the people that were born there

PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
SELECT ?place ?person
WHERE {
}
Example 3 – multiple triple pattern

- List all the names and the birthdates of the people that were born in Turin

```sparql
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?birth
WHERE
{ ?person dbo:birthPlace :Turin .
?person foaf:name ?name .
}
```
Example 3 – multiple triple pattern

- List all the names and the birthdates of the people that were born in Turin
- The same, but grouping triplets
- It is possible to convert data types

```prefix: <http://dbpedia.org/resource/>
prefix dbo: <http://dbpedia.org/ontology/>
prefix foaf: <http://xmlns.com/foaf/0.1/>
select str(?name) ?birth
where
{ ?person dbo:birthPlace :Turin ;
  dbo:birthDate ?birth ;
  foaf:name ?name .
}
data type conversion```
Example 4 – traversing a graph

- Find all the people that were born anywhere in Italy

PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT ?person ?place
WHERE {
  ?place dbo:country dbr:Italy .
}
Example 5 – LIMIT modifier

- LIMIT
  - limits the number of rows returned by query
- Find the first 25 people that were born anywhere in Italy

```reasoning
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
SELECT ?person ?place
WHERE {
  ?place dbo:country dbr:Italy .
} LIMIT 25
```
Example 6 – ORDER BY modifier

- ORDER BY
  - Sorts the query solutions on the value of one or more variables
- Find the first 25 people that were born anywhere in Italy sorted by name in descending order

```sparql
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT STR(?name) ?place
WHERE {
  ?person dbo:birthPlace ?place ;
  foaf:name ?name .
  ?place dbo:country dbr:Italy .
} ORDER BY DESC(STR(?name))
LIMIT 25
```
Example 7 – SPARQL filters

• FILTER constraints use boolean conditions to filter out unwanted query results
• Find all the people that were born in Turin after 1960, in alphabetical order

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?birth ?person
WHERE {
  ?person dbo:birthPlace :Turin .
  ?person foaf:name ?name .
  FILTER (?birth > "1960-01-01"^^xsd:date) .
}
ORDER BY ?name
```
SPARQL filters

• Conditions on literal values
• Syntax

::: FILTER expression

• Examples

Pay attention to data types!!!
XML Schema data types

FILTER (?age > 30)
FILTER (?birth > "1960-01-01"^^xsd:date)
FILTER isIRI(?x)
FILTER !BOUND(?y)
SPARQL filters

• BOUND(var)
  – true if var is bound in query answer
  – false, otherwise
  – !BOUND(var) enables negation-as-failure

• Testing types
  – isIRI(A): A is an “Internationalized Resource Identifier”
  – isBLANK(A): A is a blank node
  – isLITERAL(A): A is a literal
SPARQL filters

• Comparison between RDF terms

• Comparison between Numeric and Date types

• Boolean AND/OR

• Basic arithmetics
Example 8 – SPARQL filters

• Find the musicians that were born in Turin
• The SPARQL keyword a is a shortcut for the common predicate rdf:type (class of a resource)

PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
WHERE {
  ?person a dbo:MusicalArtist .
  ?person dbo:birthPlace :Turin .
  ?person foaf:name ?name .
  FILTER (LANG(?description) = 'en') .
} ORDER BY ?name
Example 9 – SPARQL filters

- Find the musicians that were born in a country with a population higher than 10,000,000 people, but not in the United States

```
PREFIX : <http://dbpedia.org/resource/>
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name ?birth ?countryOfBirth ?population
WHERE {
  ?person a dbo:MusicalArtist .
  ?countryOfBirth a dbo:Country ;
      dbo:populationTotal ?population .
  ?person foaf:name ?name .
  FILTER ((?population > 10000000) && (?countryOfBirth != :United_States)) .
} ORDER BY ?name
```
Example 10 – OPTIONAL patterns

- OPTIONAL tries to match a graph pattern, but doesn’t fail the whole query if the optional match fails
  - If an OPTIONAL pattern fails to match for a particular solution, any variables in that pattern remain unbound (no value) for that solution
- Find the musicians that were born in Turin, and show their image (when possible)

```
SELECT ?name ?birth ?thumbnail
WHERE {
  ?person a dbo:MusicalArtist .
  ?person dbo:birthPlace :Turin .
  ?person foaf:name ?name .
  OPTIONAL { ?person dbo:thumbnail ?thumbnail . } 
} ORDER BY ?name
```
PROVIDING RDF DATA
How to provide RDF data?

Application

SPARQL Query

Return in XML, JSON, ...

SPARQL “Engine”

GRDDL,
(e.g., microformats)

RDFa

SQL–SPARQL
“Bridge”

RDF Data

Documents (XHTML, XML, …)

(Relational) Database
From HTML to RDF

- Problem: usually HTML content and RDF data are separate
From HTML to RDF

• Separate HTML content and RDF data
• Maintenance problem
  – Both need to be managed separately
  – RDF content and web content have much overlap (redundancy)
  – RDF/XML difficult to author: extra overhead
• Verification problem
  – How to check differences as content changes?
• Visibility problem
  – Easy to ignore the RDF content (out of sight, out of mind)
From HTML to RDF

• Solution: embed RDF into web content using RDFa
RDFa

- W3C Recommendation (March 17th, 2015)
- RDFa (Resource Description Framework in Attributes) adds a set of attribute-level extensions to HTML5, XHTML and various XML-based document types for embedding rich metadata within Web documents
- The RDF data-model mapping enables its use for embedding RDF subject-predicate-object expressions within XHTML documents. It also enables the extraction of RDF model triples by compliant user agents
RDFa

- Extra (RDFa) markup is ignored by web browsers

```html
<div id="saleprice" rel="gr:hasPriceSpecification">
  <div class="saletext">
    Our Price:
  </div>
  <div class="salenum" typeof="gr:UnitPriceSpecification" about="#UnitPriceSpecification_9929089_sale">
    <!-- B:0LI -->
    <span class="price">$29.99</span>
    <span property="gr:hasCurrencyValue" datatype="xsd:float" content="29.99"></span>
    <span property="gr:hasCurrency" datatype="xsd:string" content="USD"></span>
    <span property="gr:hasUnitOfMeasurement" datatype="xsd:string" content="USD"></span>
    <span property="gr:valueAddedTaxIncluded" datatype="xsd:boolean" content="true"></span>
    <!-- E:0LI -->
  </div>
</div>
```
RDFa is not the only solution ...

510% increase between March, 2009 and October, 2010

Percentage of URLs with embedded metadata in various formats

Source: Peter Mika (Yahoo!), RDFa, 2011
Rich snippets

• Several solutions for embedding semantic data in Web
• Three syntaxes known (by Google) as “rich snippets”
  – Microformats
  – RDFa
  – HTML microdata
• All three are supported by Google, while microdata is the “recommended” syntax
First came microformats

- Microformats emerged around 2005
- Some key principles
  - Start by solving simple, specific problems
  - Design for humans first, machines second
- Wide deployment
  - Used on billions of Web pages
- Formats exist for marking up atom feeds, calendars, addresses and contact info, geo-location, multimedia, news, products, recipes, reviews, resumes, social relationships, etc.
Microformats example

```html
<div class="vcard">
  <a class="fn org url"
     href="http://www.commerce.net/">
    CommerceNet</a>
  <div class="adr">
    <span class="type">Work</span>: 
    <div class="street-address">169 University Avenue</div>
    <span class="locality">Palo Alto</span>,
    <abbr class="region" title="California">CA</abbr>&nbsp;&nbsp;
    <span class="postal-code">94301</span>
  </div>
  <div class="country-name">USA</div>
</div>
<div class="tel">
  <span class="type">Work</span> +1-650-289-4040
</div>
<div>Email: 
  <span class="email">info@commerce.net</span>
</div>
```
Then came RDFa

- RDFa aims to bridge the gap between human oriented HTML and machine-oriented RDF documents
- Provides XHTML attributes to indicate machine understandable information
- Uses the RDF data model, and Semantic Web vocabularies directly
Last but not least, microdata

• Microdata syntax is based on nested groups of name-value pairs
• HTML microdata specification includes
  – An unambiguous parsing model
  – An algorithm to convert microdata to RDF
• Compatible with the Semantic Web via mappings
Microdata example

```html
<div itemscope itemtype="http://schema.org/Movie">
  <h1 itemprop="name">Avatar</h1>
  <div itemprop="director" itemscope itemtype="http://schema.org/Person">
    Director: <span itemprop="name">James Cameron</span> (born August 16, 1954)
  </div>
  <span itemprop="genre">Science fiction</span>
  <a href="../movies/avatar-theatrical-trailer.html" itemprop="trailer">Trailer</a>
</div>
```
GRDDL: from XML to RDF

• GRDDL is a mechanism for Gleaning Resource Descriptions from Dialects of Languages

• GRDDL specification introduces markup based on existing standards for declaring that an XML document includes data compatible with the Resource Description Framework (RDF) and for linking to algorithms (typically represented in XSLT), for extracting this data from the document
References

• W3C Semantic Web
  – https://www.w3.org/standards/semanticweb/
• http://www.cambridgesemantics.com/semantic-university/sparql-by-example
• http://wiki.dbpedia.org/OnlineAccess
License

• This work is licensed under the Creative Commons “Attribution-NonCommercial-ShareAlike Unported (CC BY-NC-SA 3,0)” License.
• You are free:
  – to Share - to copy, distribute and transmit the work
  – to Remix - to adapt the work
• Under the following conditions:
  – Attribution - You must attribute the work in the manner specified by the author or licensor (but not in any way that suggests that they endorse you or your use of the work).
  – Noncommercial - You may not use this work for commercial purposes.
  – Share Alike - If you alter, transform, or build upon this work, you may distribute the resulting work only under the same or similar license to this one.
• To view a copy of this license, visit http://creativecommons.org/license/by-nc-sa/3.0/