CSC 481: Resource Description Framework (RDF)

Based on Chapter 3 of “Semantic Web for the Working Ontologist - Modeling in RDF, RDFS and OWL” by Allenmag and Hendler

Rodrigo Canaan
Assistant Professor
Computer Science Department
Cal Poly, San Luis Obispo
rcanaan@calpoly.edu

Note: book is available at Cal Poly Library

Note: book is available at Cal Poly Library

- https://csu-calpoly.primo.exlibrisgroup.com/permalink/01CALS_PSU/1qh1nk7/alma9910676171478029O1

Note: book is available at Cal Poly Library

- https://csu-calpoly.primo.exlibrisgroup.com/permalink/01CALS_PSU/1qh1nk7/alma9910676171478029O1

Note: book is available at Cal Poly Library

- https://csu-calpoly.primo.exlibrisgroup.com/permalink/01CALS_PSU/1qh1nk7/alma9910676171478029O1

- https://ebookcentral.proquest.com/lib/calpoly/detail.action?docID=689813&pq-origsite=primo#

Main considerations for semantic web

Main considerations for semantic web

- The AAA slogan (**A**nyone can say **A**nything about **A**ny topic)

Main considerations for semantic web

- The AAA slogan (**A**nyone can say **A**nything about **A**ny topic)
- Open world assumption

Main considerations for semantic web

- The AAA slogan (**A**nyone can say **A**nything about **A**ny topic)
- Open world assumption
- Non-unique naming

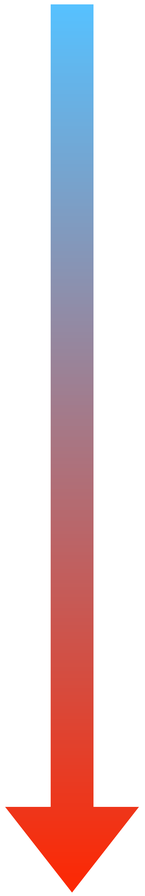
Main considerations for semantic web

- The AAA slogan (**A**nyone can say **A**nything about **A**ny topic)
- Open world assumption
- Non-unique naming
- Network effect

Semantic Web Standards

By level of expressivity

Less expressive



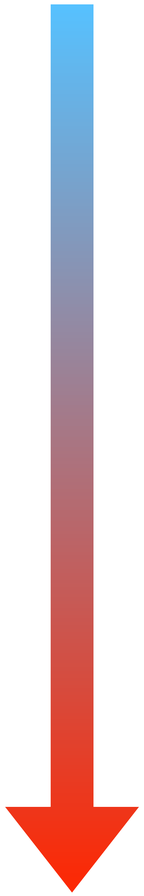
More expressive

- **RDF - Resource Description Framework**. Models data as triplets (subject, predicate, object)
- **RDFS - RDF Schema language**. Enhances RDF with notions of classes, subclasses and properties.
- **OWL - Web Ontology Language**. adds more vocabulary for describing properties and classes (e.g. disjointness, cardinality, symmetry of properties…)
    - Related to Description Logics (ch. 9 of the Brachman & Levesque book)

Semantic Web Standards

By level of expressivity

Less expressive



More expressive

Our focus today

- **RDF - Resource Description Framework**. Models data as triplets (subject, predicate, object)
- **RDFS - RDF Schema language**. Enhances RDF with notions of classes, subclasses and properties.
- **OWL - Web Ontology Language**. adds more vocabulary for describing properties and classes (e.g. disjointness, cardinality, symmetry of properties…)
    - Related to Description Logics (ch. 9 of the Brachman & Levesque book)

What is RDF?

- RDF is a **Data Model** - a way of thinking, organizing and exchanging data
- In other words, it is an abstraction, not a specific language/syntax
- Knowledge represented as a directed graph composed of triples
- Triples defined by:
  - A node for the subject
  - A node for the predicate
  - A node for the object
- **RDFS - RDF Schema language**. Enhances RDF with notions of classes, subclasses and properties.
- **OWL - Web Ontology Language**. adds more vocabulary for describing properties and classes (e.g. disjointness, cardinality, symmetry of properties…)
    - Related to Description Logics (ch. 9 of the Brachman & Levesque book)

From tabular data to triples

Table 3.1 Tabular Data about Elizabethan Literature and Music

ID	Title	Author	Medium	Year
1	*As You Like It*	Shakespeare	Play	1599
2	*Hamlet*	Shakespeare	Play	1604
3	*Othello*	Shakespeare	Play	1603
4	"Sonnet 78"	Shakespeare	Poem	1609
5	*Astrophil and Stella*	Sir Phillip Sidney	Poem	1590
6	*Edward II*	Christopher Marlowe	Play	1592
7	*Hero and Leander*	Christopher Marlowe	Poem	1593
8	*Greensleeves*	Henry VIII Rex	Song	1525

From tabular data to triples

bular Data about Elizabethan Literature and Music

Title	Author	Medium	Year
As You Like It	Shakespeare	Play	1599
Hamlet	Shakespeare	Play	1604
Othello	Shakespeare	Play	1603
"Sonnet 78"	Shakespeare	Poem	1609
Astrophil and Stella	Sir Phillip Sidney	Poem	1590
Edward II	Christopher Marlowe	Play	1592
Hero and Leander	Christopher Marlowe	Poem	1593
Greensleeves	Henry VIII Rex	Song	1525

From tabular data to triples

bular Data about Elizabethan Literature and Music

Title	Author	Medium	Year
As You Like It	Shakespeare	Play	1599
Hamlet	Shakespeare	Play	1604
Othello	Shakespeare	Play	1603
"Sonnet 78"	Shakespeare	Poem	1609
Astrophil and Stella	Sir Phillip Sidney	Poem	1590
Edward II	Christopher Marlowe	Play	1592
Hero and Leander	Christopher Marlowe	Poem	1593
Greensleeves	Henry VIII Rex	Song	1525

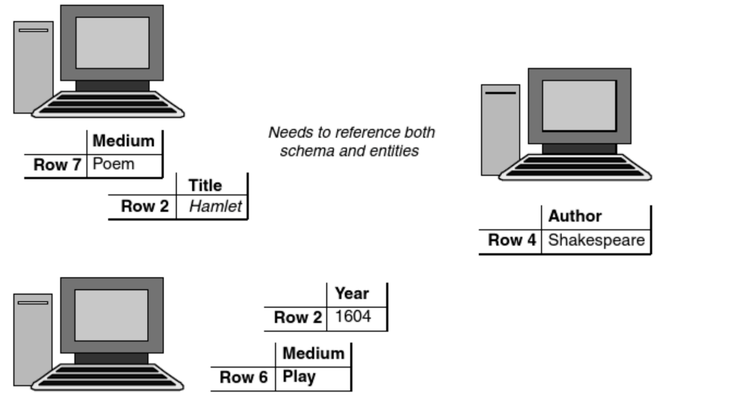


FIGURE 3.3

Distributing data across the Web, cell by cell.

From tabular data to triples

Table 3.1 Tabular Data about Elizabethan Literature and Music

ID	Title	Author	Medium	Year
1	*As You Like It*	Shakespeare	Play	1599
2	*Hamlet*	Shakespeare	Play	1604
3	*Othello*	Shakespeare	Play	1603
4	"Sonnet 78"	Shakespeare	Poem	1609
5	*Astrophil and Stella*	Sir Phillip Sidney	Poem	1590
6	*Edward II*	Christopher Marlowe	Play	1592
7	*Hero and Leander*	Christopher Marlowe	Poem	1593
8	*Greensleeves*	Henry VIII Rex	Song	1525

From tabular data to triples

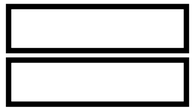
bular Data about Elizabethan Literature and Music

Title	Author	Medium	Year
As You Like It	Shakespeare	Play	1599
Hamlet	Shakespeare	Play	1604
Othello	Shakespeare	Play	1603
"Sonnet 78"	Shakespeare	Poem	1609
Astrophil and Stella	Sir Phillip Sidney	Poem	1590
Edward II	Christopher Marlowe	Play	1592
Hero and Leander	Christopher Marlowe	Poem	1593
Greensleeves	Henry VIII Rex	Song	1525

From tabular data to triples

bular Data about Elizabethan Literature and Music

Title	Author	Medium	Year
As You Like It	Shakespeare	Play	1599
Hamlet	Shakespeare	Play	1604
Othello	Shakespeare	Play	1603
"Sonnet 78"	Shakespeare	Poem	1609
Astrophil and Stella	Sir Phillip Sidney	Poem	1590
Edward II	Christopher Marlowe	Play	1592
Hero and Leander	Christopher Marlowe	Poem	1593
Greensleeves	Henry VIII Rex	Song	1525



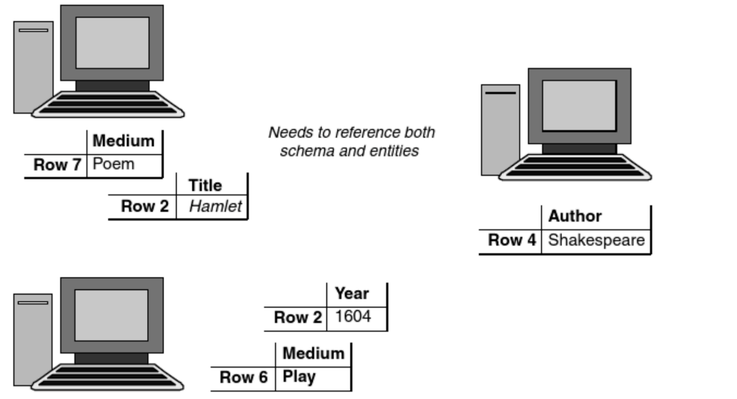


FIGURE 3.3

Distributing data across the Web, cell by cell.

Triples as building blocks of RDF

Table 3.3 Sample Triples

Subject	Predicate	Object
Shakespeare	wrote	King Lear
Shakespeare	wrote	Macbeth
Anne Hathaway	married	Shakespeare
Shakespeare	livedIn	Stratford
Stratford	isIn	England
Macbeth	setIn	Scotland
England	partOf	UK
Scotland	partOf	UK

Triples as building blocks of RDF

Table 3.3 Sample Triples

Subject	Predicate	Object
Shakespeare	wrote	King Lear
Shakespeare	wrote	Macbeth
Anne Hathaway	married	Shakespeare
Shakespeare	livedIn	Stratford
Stratford	isIn	England
Macbeth	setIn	Scotland
England	partOf	UK
Scotland	partOf	UK

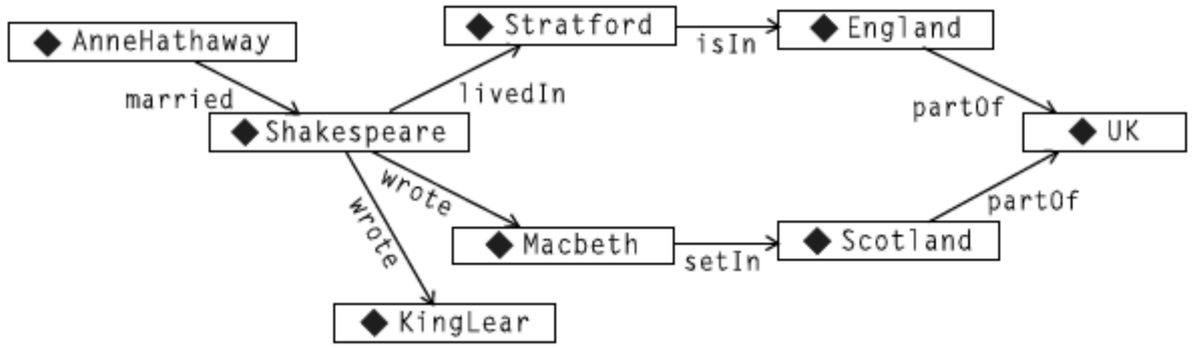


FIGURE 3.4

Graph display of triples from Table 3.3. Eight triples appear as eight labeled edges.

Merging data from multiple sources

Table 3.4 Triples about Shakespeare's Plays

Subject	Predicate	Object
Shakespeare	Wrote	*As You Like It*
Shakespeare	Wrote	*Henry V*
Shakespeare	Wrote	*Love's Labour's Lost*
Shakespeare	Wrote	*Measure for Measure*
Shakespeare	Wrote	*Twelfth Night*
Shakespeare	Wrote	*The Winter's Tale*
Shakespeare	Wrote	*Hamlet*
Shakespeare	Wrote	*Othello*
		etc.

(a)

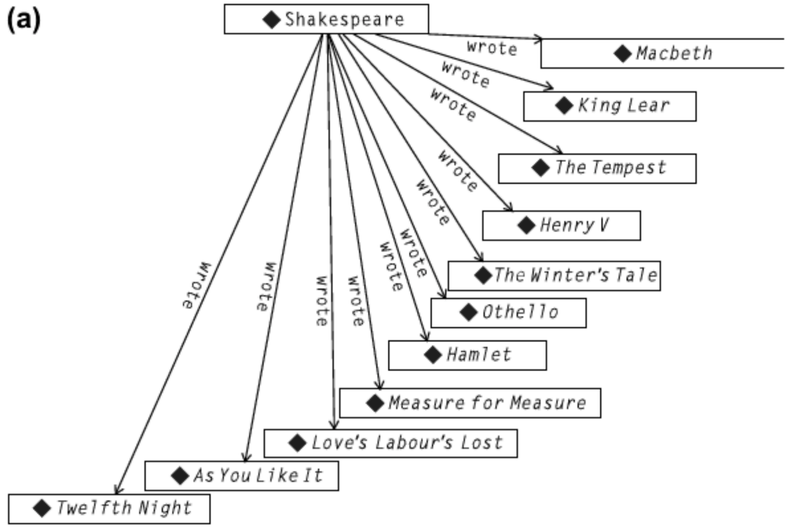
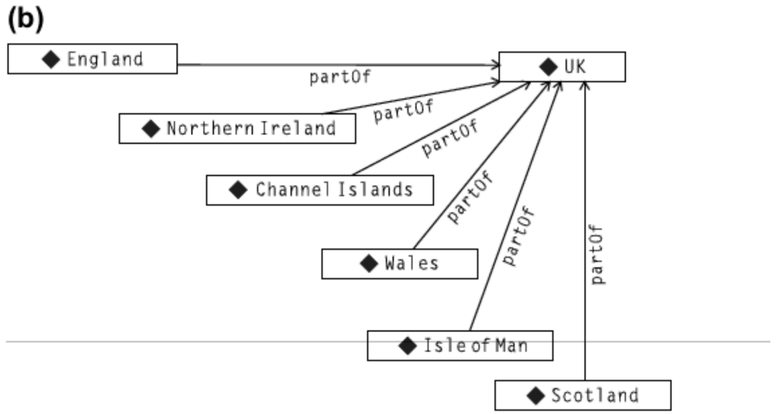


Table 3.5 Triples about the Parts of the United Kingdom

Subject	Predicate	Object
Scotland	part Of	The UK
England	part Of	The UK
Wales	part Of	The UK
Northern Ireland	part Of	The UK
Channel Islands	part Of	The UK
Isle of Man	part Of	The UK

(b)



Merging data from multiple sources

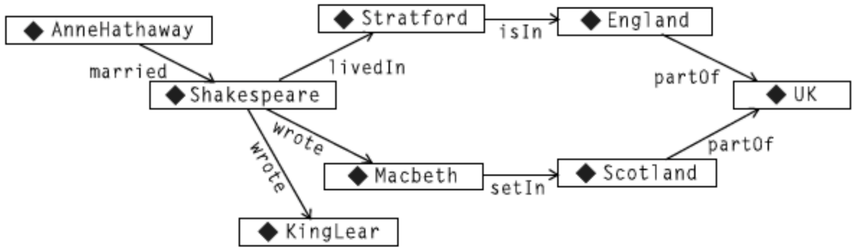
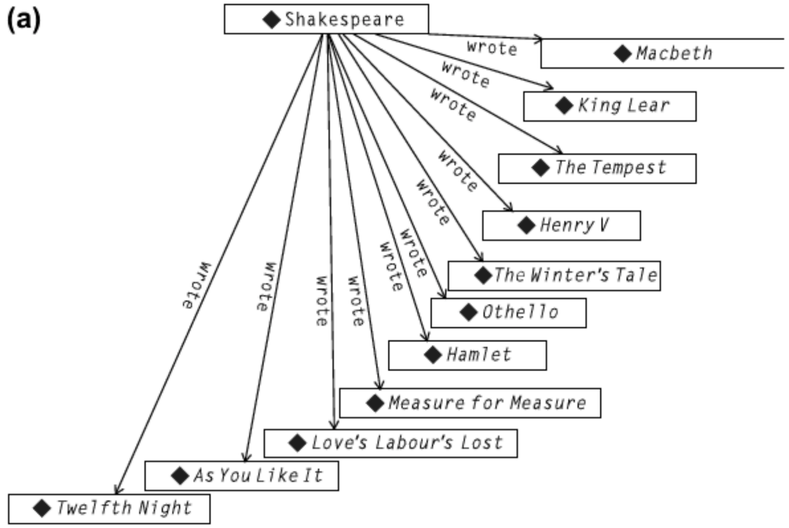
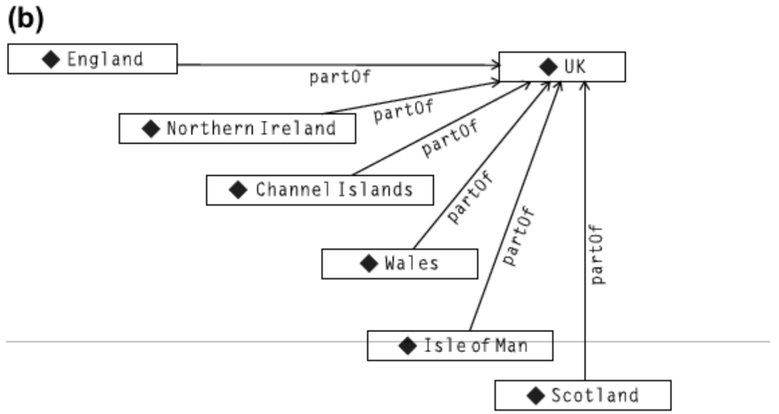


FIGURE 3.4

Graph display of triples from Table 3.3. Eight triples appear as eight labeled edges.





Merging data from multiple sources

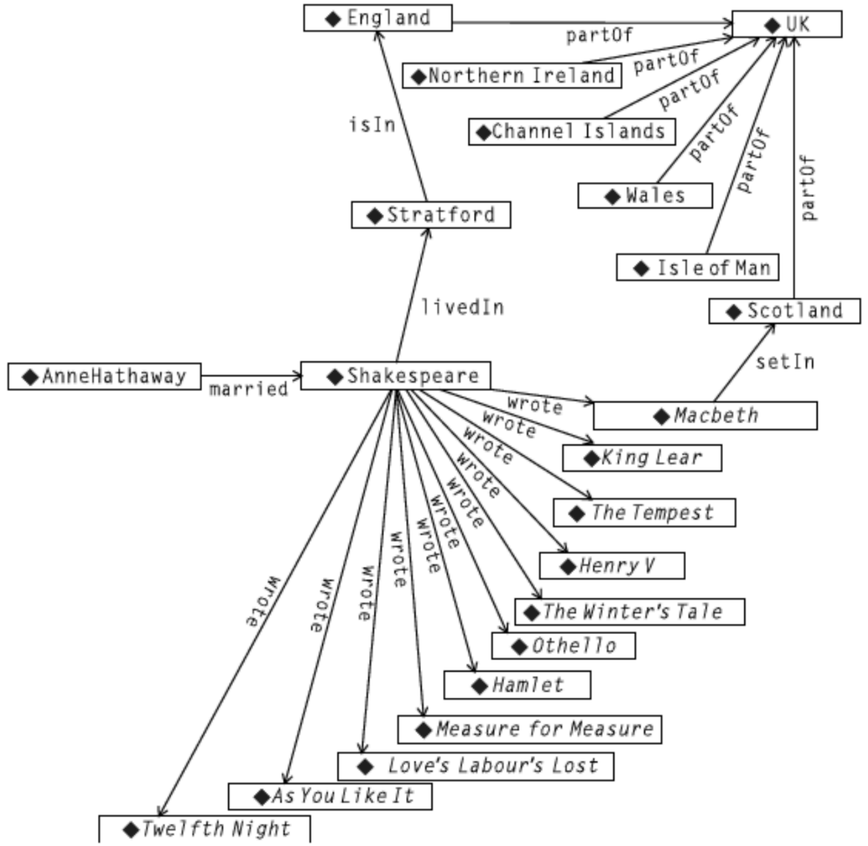


FIGURE 3.6

Combined graph of all triples about Shakespeare and the United Kingdom.

Merging data from multiple sources

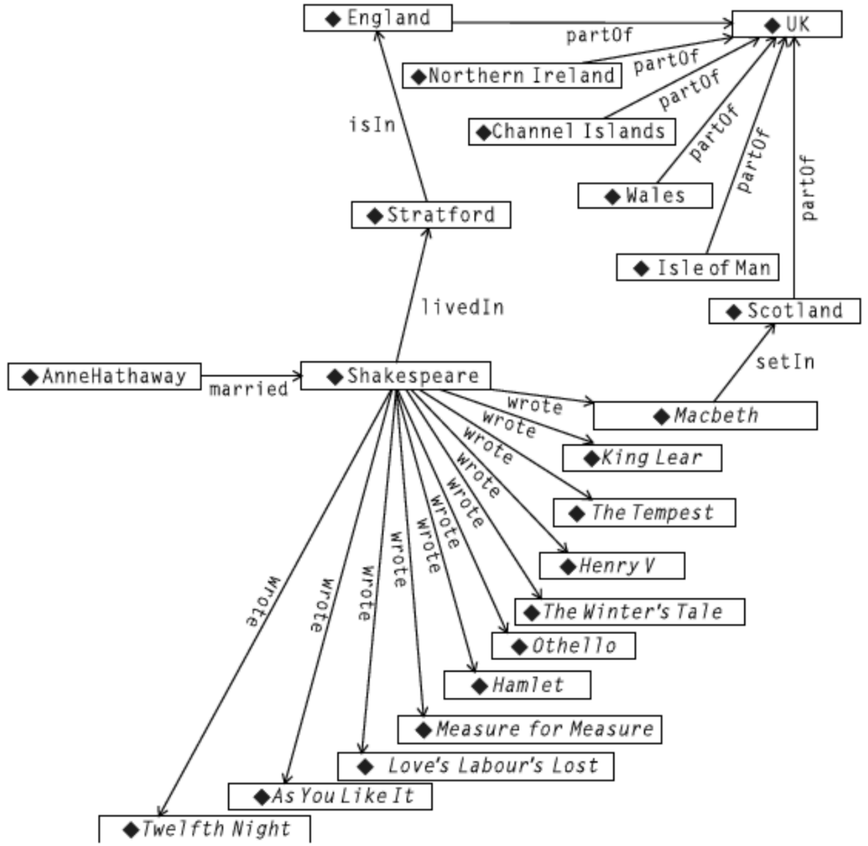


FIGURE 3.6

Combined graph of all triples about Shakespeare and the United Kingdom.

(As long as we know which nodes in the graph of each source match)

Merging data from multiple sources

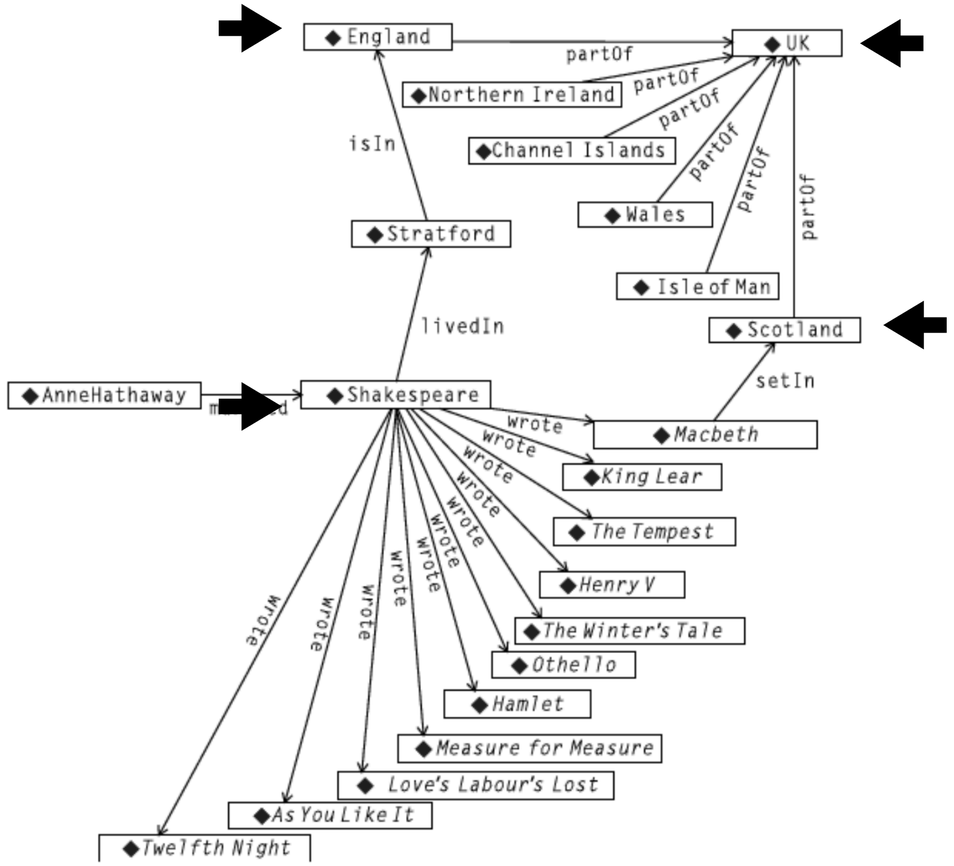


FIGURE 3.6

Combined graph of all triples about Shakespeare and the United Kingdom.

(As long as we know which nodes in the graph of each source match)

When are two resources the same?

When are two resources the same?

- A given name in natural language (e.g. “Washington”) may refer to many different concepts: the president, the state, the capital of the US…

When are two resources the same?

- A given name in natural language (e.g. “Washington”) may refer to many different concepts: the president, the state, the capital of the US…
- Within RDF, any identifiable concept (node in the graph) is called a “resource”

When are two resources the same?

- A given name in natural language (e.g. “Washington”) may refer to many different concepts: the president, the state, the capital of the US…
- Within RDF, any identifiable concept (node in the graph) is called a “resource”
- Two resources are the same if they share the same **Uniform Resource Identifier (URI)**

URIs vs URL

URIs vs URL

- **URI:** Uniform Resource *Identifier*

URIs vs URL

- **URI:** Uniform Resource *Identifier*
  - Focus on the model

URIs vs URL

- **URI:** Uniform Resource *Identifier*
  - Focus on the model

URIs vs URL

- **URI:** Uniform Resource *Identifier*
  - Focus on the model

- **URL:** Uniform Resource *Locator*

URIs vs URL

- **URI:** Uniform Resource *Identifier*
  - Focus on the model

- **URL:** Uniform Resource *Locator*
  - Focus on retrieval

URIs vs URL

- **URI:** Uniform Resource *Identifier*
  - Focus on the model

- **URL:** Uniform Resource *Locator*
  - Focus on retrieval

URIs vs URL

- **URI:** Uniform Resource *Identifier*
  - Focus on the model

- **URL:** Uniform Resource *Locator*
  - Focus on retrieval

- URL is a special case of URI (every URL is an URI but not vice-versa)

URIs example and abreviation

URIs example and abreviation

- A fully expressed URI looks like an URL. Example:

URIs example and abreviation

- A fully expressed URI looks like an URL. Example:

- https://dbpedia.org/page/William_Shakespeare is the URI of the DBPedia entry on William Shakespeare

URIs example and abreviation

- A fully expressed URI looks like an URL. Example:
- https://dbpedia.org/page/William_Shakespeare is the URI of the DBPedia entry on William Shakespeare
- For ease of readability, it is also common to express it with a *qname* , consisting of a namespace and identifier

URIs example and abreviation

URIs example and abreviation

- The binding between a namespace and the URI is local to each application.

URIs example and abreviation

- The binding between a namespace and the URI is local to each application.
- Ex:

URIs example and abreviation

- The binding between a namespace and the URI is local to each application.
- Ex:

after declaring (in turtle notation)

URIs example and abreviation

- The binding between a namespace and the URI is local to each application.
- Ex:

after declaring (in turtle notation)

```
@prefix dbr:<http://dbpedia.org/resource/> .
```

URIs example and abreviation

- The binding between a namespace and the URI is local to each application.
- Ex:

after declaring (in turtle notation)

```
@prefix dbr:<http://dbpedia.org/resource/> .
```

URIs example and abreviation

- The binding between a namespace and the URI is local to each application.
- Ex:

after declaring (in turtle notation)

```
@prefix dbr:<http://dbpedia.org/resource/> .
```

https://dbpedia.org/page/William_Shakespeare becomes
dbr:William_Shakespeare

URIs example and abreviation

- The binding between a namespace and the URI is local to each application.
- Ex:

after declaring (in turtle notation)

```
@prefix dbr:<http://dbpedia.org/resource/> .
```

https://dbpedia.org/page/William_Shakespeare becomes
dbr:William_Shakespeare

URIs example and abreviation

- The W3C also defines a few standard namespaces with important predicates:

***rdf*:** Indicates identifiers used in RDF. The set of identifiers defined in the standard is quite small and is used to define types and properties in RDF. The global URI for the *rdf* namespace is *http://www.w3.org/1999/02/22-rdf-syntax-ns#.*

***rdfs*:** Indicates identifiers used for the RDF Schema language, RDFS. The scope and semantics of the symbols in this namespace are the topics of future chapters. The global URI for the *rdfs* namespace is *http://www.w3.org/2000/01/rdf-schema#.*

***owl*:** Indicates identifiers used for the Web Ontology Language, OWL. The scope and semantics of the symbols in this namespace are the topics of future chapters. The global URI for the *owl* namespace is *http://www.w3.org/2002/07/owl#.*

See:
https://www.w3.org/1999/02/22-rdf-syntax-ns
https://www.w3.org/2000/01/rdf-schema#
https://www.w3.org/2002/07/owl#Thing

Important RDF resources

rdf:type

- Provides an elementary typing system
- Types themselves can also have types

Table 3.9 Using `rdf:type` to Describe Playwrights

Subject	Predicate	Object
lit:Shakespeare	rdf:type	lit:Playwright
lit:Ibsen	rdf:type	lit:Playwright
lit:Simon	rdf:type	lit:Playwright
lit:Miller	rdf:type	lit:Playwright
lit:Marlowe	rdf:type	lit:Playwright
lit:Wilder	rdf:type	lit:Playwright

Table 3.10 Defining Types of Names

Subject	Predicate	Object
lit:Playwright	rdf:type	bus:Profession
bus:Profession	rdf:type	hr:Compensation

Important RDF resources

rdf:property

- Indicates that something is to be used as a predicate rather than as a subject or object
- Typically not enforced by inference engines
- Still useful for reasonin, semantic clarity and documentation

Table 3.11 `rdf:Property` Assertions for Tables 3.5 to 3.8

Subject	Predicate	Object
lit:wrote	rdf:type	rdf:Property
geo:partOf	rdf:type	rdf:Property
bio:married	rdf:type	rdf:Property
bio:livedIn	rdf:type	rdf:Property
bio:livedWith	rdf:type	rdf:Property
geo:isIn	rdf:type	rdf:Property

RDF and Tabular Data

How to convert tabular data to RDF?

Table 3.12 Sample Tabular Data for Triples

Product						
ID	**Model Number**	**Division**	**Product Line**	**Manufacture Location**	**SKU**	**Available**
1	ZX-3	Manufacturing support	Paper machine	Sacramento	FB3524	23
2	ZX-3P	Manufacturing support	Paper machine	Sacramento	KD5243	4
3	ZX-3S	Manufacturing support	Paper machine	Sacramento	IL4028	34
4	B-1430	Control engineering	Feedback line	Elizabeth	KS4520	23
5	B-1430X	Control engineering	Feedback line	Elizabeth	CL5934	14
6	B-1431	Control engineering	Active sensor	Seoul	KK3945	0
7	DBB-12	Accessories	Monitor	Hong Kong	ND5520	100
8	SP-1234	Safety	Safety valve	Cleveland	HI4554	4
9	SPX-1234	Safety	Safety valve	Cleveland	OP5333	14

RDF and Tabular Data

RDF and Tabular Data

- The whole database becomes a namespace (in the example, ‘mfg:’)

RDF and Tabular Data

- The whole database becomes a namespace (in the example, ‘mfg:’)
- Each table becomes a type (in the example, ‘mfg:Product’)

RDF and Tabular Data

- The whole database becomes a namespace (in the example, ‘mfg:’)
- Each table becomes a type (in the example, ‘mfg:Product’)
- Each line gets each own URI

RDF and Tabular Data

- The whole database becomes a namespace (in the example, ‘mfg:’)
- Each table becomes a type (in the example, ‘mfg:Product’)
- Each line gets each own URI
  - e.g. mfg:Product 1, mfg:Product2, etc

RDF and Tabular Data

Table 3.14 Triples Representing Type of Information from Table 3.12

Subject	Predicate	Object
mfg:Product1	rdf:type	mfg:Product
mfg:Product2	rdf:type	mfg:Product
mfg:Product3	rdf:type	mfg:Product
mfg:Product4	rdf:type	mfg:Product
mfg:Product5	rdf:type	mfg:Product
mfg:Product6	rdf:type	mfg:Product
mfg:Product7	rdf:type	mfg:Product
mfg:Product8	rdf:type	mfg:Product
mfg:Product9	rdf:type	mfg:Product

RDF and Tabular Data

RDF and Tabular Data

- For each line, we state that element to be of the appropriate type

RDF and Tabular Data

- For each line, we state that element to be of the appropriate type
  - e.g. mfg:Product1 rdf:type mfg:product

RDF and Tabular Data

- For each line, we state that element to be of the appropriate type
  - e.g. mfg:Product1 rdf:type mfg:product
- Each column becomes a property, each cell becomes a triple

RDF and Tabular Data

- For each line, we state that element to be of the appropriate type
  - e.g. mfg:Product1 rdf:type mfg:product
- Each column becomes a property, each cell becomes a triple
  - e.g. mfg:Product1 mfg:Product_ModelNo “ZX-3”

RDF and Tabular Data

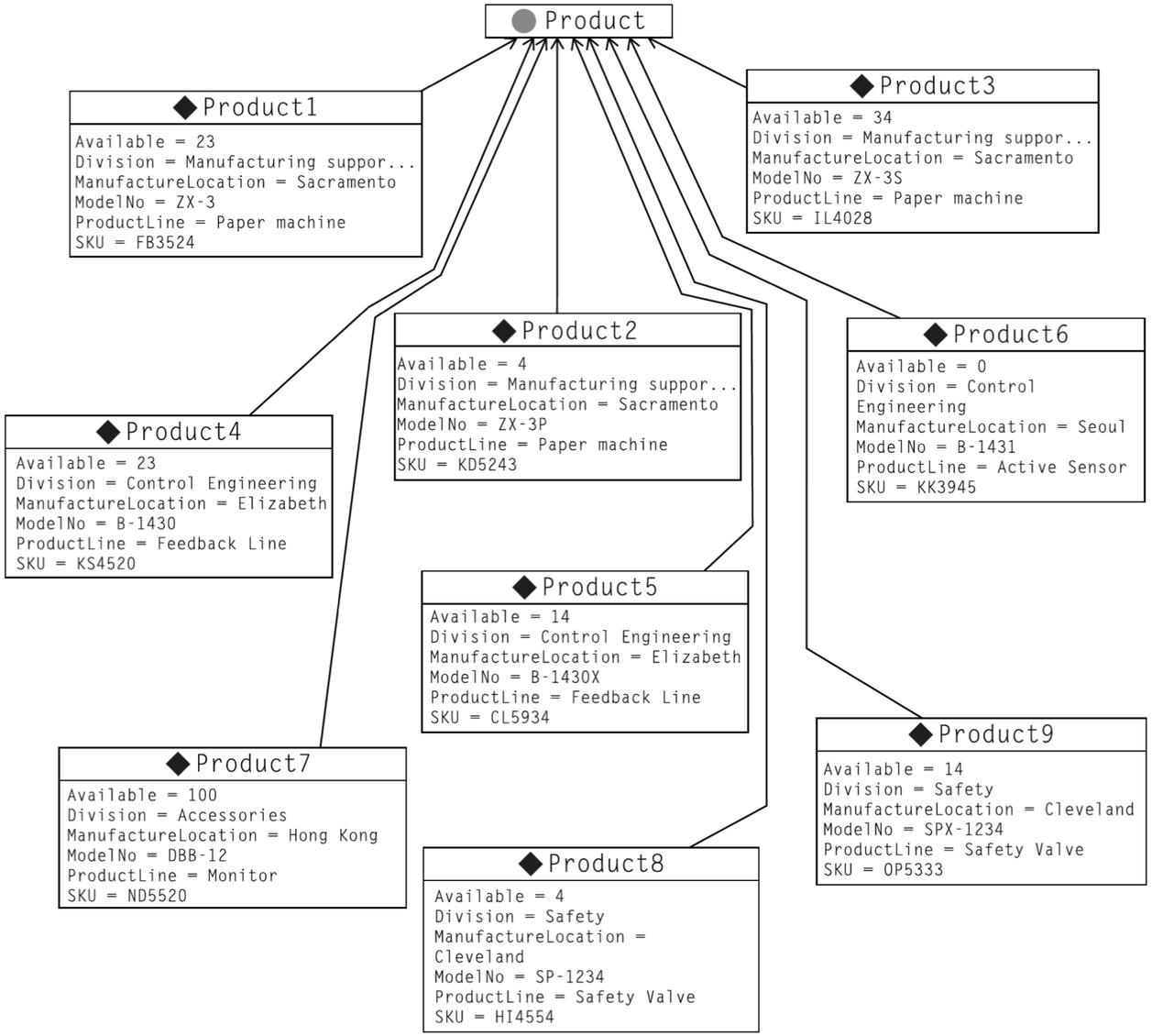
- For each line, we state that element to be of the appropriate type
  - e.g. mfg:Product1 rdf:type mfg:product
- Each column becomes a property, each cell becomes a triple
  - e.g. mfg:Product1 mfg:Product_ModelNo “ZX-3”
- Object of each triple may be a literal data type (Integer, String) rather than a resource

RDF and Tabular Data

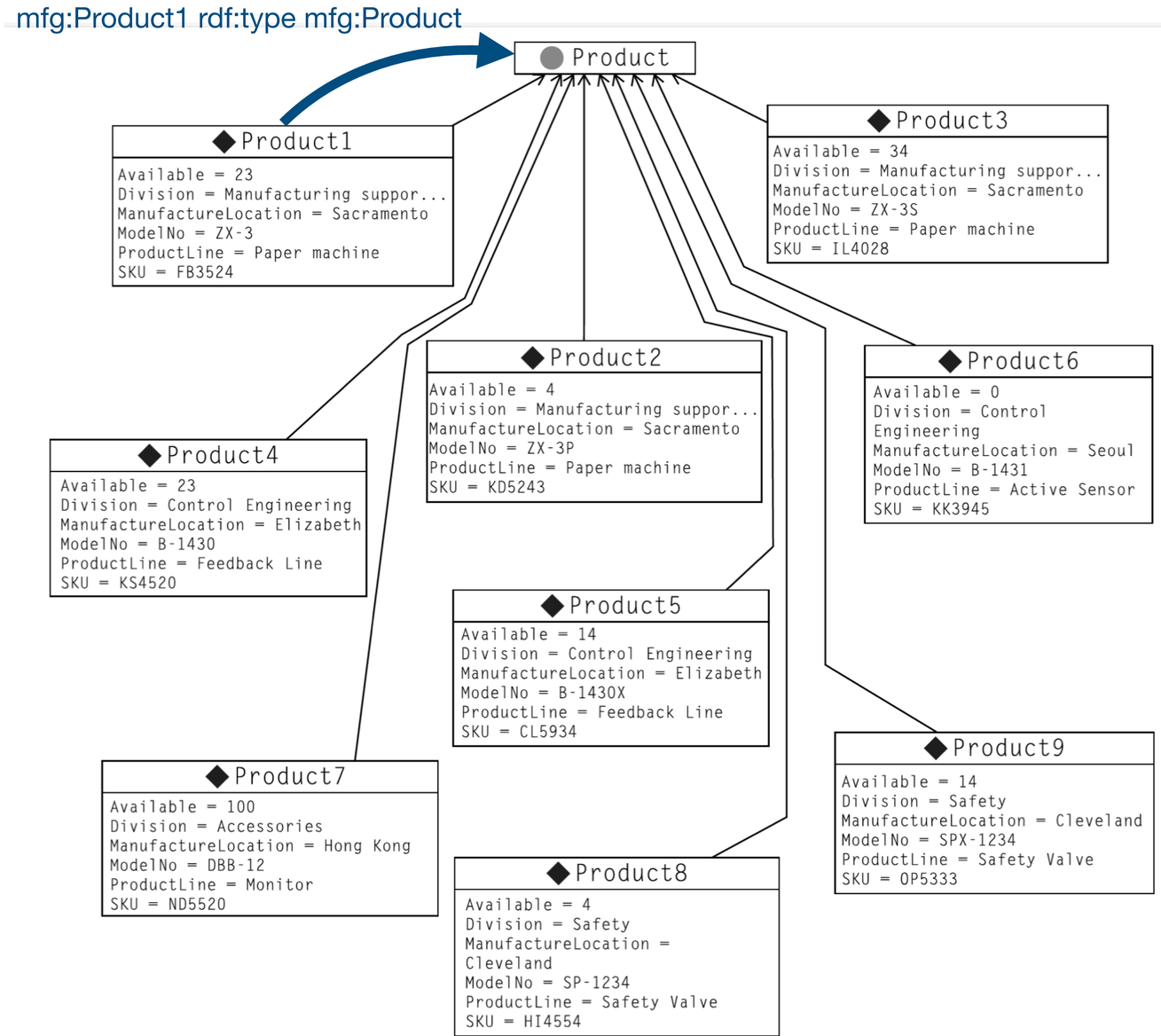
Table 3.13 Triples Representing Some of the Data in Table 3.12

Subject	Predicate	Object
mfg:Product1	mfg:Product_ID	1
mfg:Product1	mfg:Product_ModelNo	ZX-3
mfg:Product1	mfg:Product_Division	Manufacturing support
mfg:Product1	mfg:Product_Product_Line	Paper machine
mfg:Product1	mfg:Product_Manufacture_Location	Sacramento
mfg:Product1	mfg:Product_SKU	FB3524
mfg:Product1	mfg:Product_Available	23
mfg:Product2	mfg:Product_ID	2
mfg:Product2	mfg:Product_ModelNo	ZX-3P
mfg:Product2	mfg:Product_Division	Manufacturing support
mfg:Product2	mfg:Product_Product_Line	Paper machine
mfg:Product2	mfg:Product_Manufacture_Location	Sacramento
mfg:Product2	mfg:Product_SKU	KD5243
mfg:Product2	mfg:Product_Available	4…

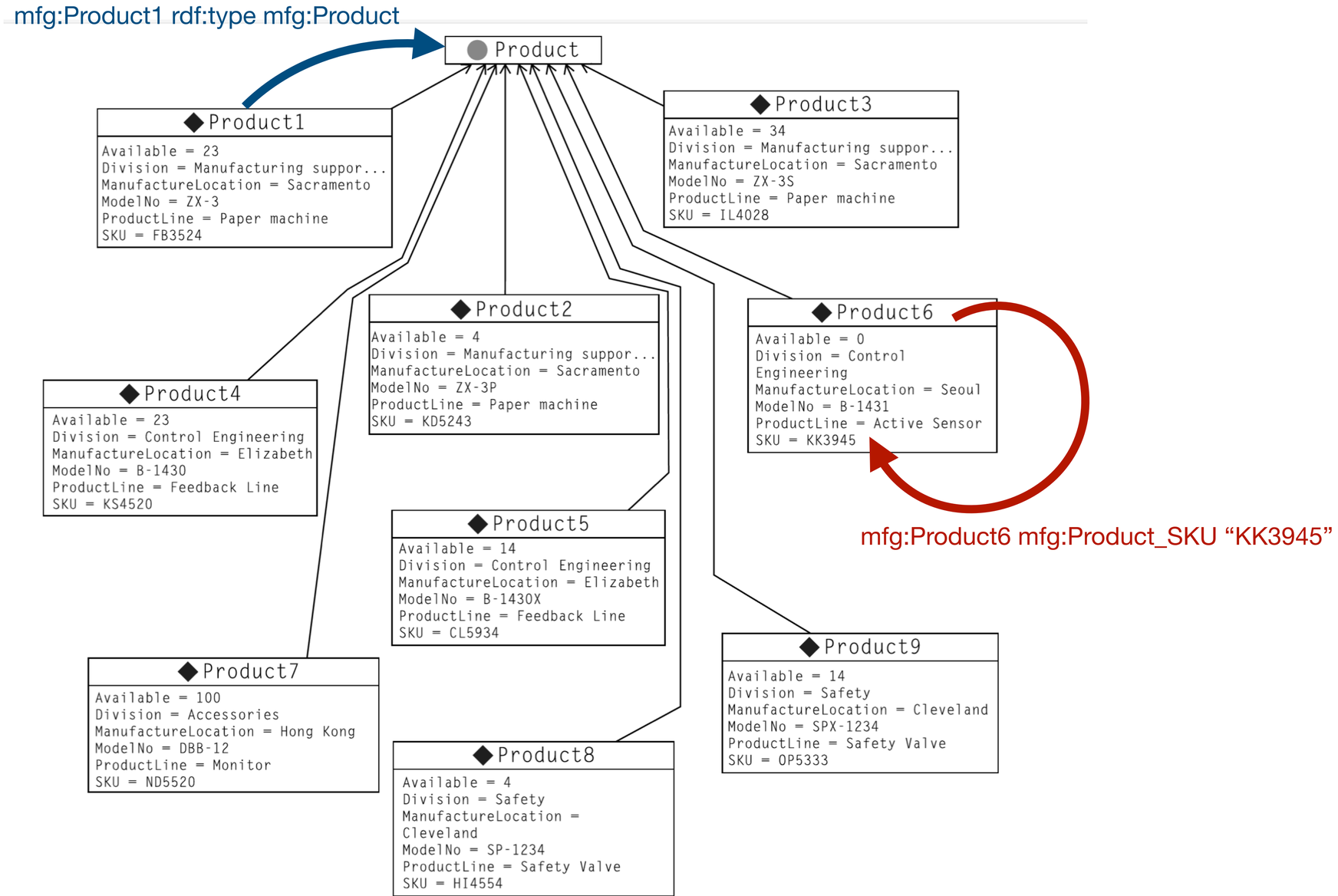
RDF and Tabular Data



RDF and Tabular Data



RDF and Tabular Data



What about higher-order relationships?

What about higher-order relationships?

- How do we state the following with triplets?

What about higher-order relationships?

- How do we state the following with triplets?
  - “Shakespeare wrote Hamlet”

What about higher-order relationships?

- How do we state the following with triplets?
  - “Shakespeare wrote Hamlet”
  - “Shakespeare wrote Hamlet in 1604”

What about higher-order relationships?

- How do we state the following with triplets?
  - “Shakespeare wrote Hamlet”
  - “Shakespeare wrote Hamlet in 1604”
  - “Wikipedia states that Shakespeare wrote Hamlet in 1604”

What about higher-order relationships?

What about higher-order relationships?

- “Shakespeare wrote Hamlet”

What about higher-order relationships?

- “Shakespeare wrote Hamlet”
  - Can be represented in rdf similarly to English

What about higher-order relationships?

- “Shakespeare wrote Hamlet”
  - Can be represented in rdf similarly to English
    - :Shakespeare :wrote :Hamlet

What about higher-order relationships?

What about higher-order relationships?

- “Shakespeare wrote Hamlet in 1604”

What about higher-order relationships?

- “Shakespeare wrote Hamlet in 1604”
  - Implicit reification

What about higher-order relationships?

- “Shakespeare wrote Hamlet in 1604”
  - Implicit reification
    - :n1 :author :Shakespeare

What about higher-order relationships?

- “Shakespeare wrote Hamlet in 1604”
  - Implicit reification
    - :n1 :author :Shakespeare
    - :n1 :title :Hamlet

What about higher-order relationships?

- “Shakespeare wrote Hamlet in 1604”
  - Implicit reification
    - :n1 :author :Shakespeare
    - :n1 :title :Hamlet
    - :n1 :publicationDate 1604

What about higher-order relationships?

- “Shakespeare wrote Hamlet in 1604”
  - Implicit reification
    - :n1 :author :Shakespeare
    - :n1 :title :Hamlet
    - :n1 :publicationDate 1604
  - Reification is implicit because the identifier n1 still represents Hamlet (an object in the domain)

What about higher-order relationships?

What about higher-order relationships?

- “Wikipedia states Shakespeare wrote Hamlet in 1604”

What about higher-order relationships?

- “Wikipedia states Shakespeare wrote Hamlet in 1604”
- Consider the simpler “Wikipedia states Shakespeare wrote Hamlet"

What about higher-order relationships?

- “Wikipedia states Shakespeare wrote Hamlet in 1604”
- Consider the simpler “Wikipedia states Shakespeare wrote Hamlet"
  - Explicit reification

What about higher-order relationships?

- “Wikipedia states Shakespeare wrote Hamlet in 1604”
- Consider the simpler “Wikipedia states Shakespeare wrote Hamlet"
  - Explicit reification
    - :n2 rdf:subject :Shakespeare

What about higher-order relationships?

- “Wikipedia states Shakespeare wrote Hamlet in 1604”
- Consider the simpler “Wikipedia states Shakespeare wrote Hamlet"
  - Explicit reification
    - :n2 rdf:subject :Shakespeare
    - :n2 rdf:predicate :wrote

What about higher-order relationships?

- “Wikipedia states Shakespeare wrote Hamlet in 1604”
- Consider the simpler “Wikipedia states Shakespeare wrote Hamlet"
  - Explicit reification
    - :n2 rdf:subject :Shakespeare
    - :n2 rdf:predicate :wrote
    - :n2 rdf:object :Hamlet

What about higher-order relationships?

- “Wikipedia states Shakespeare wrote Hamlet in 1604”
- Consider the simpler “Wikipedia states Shakespeare wrote Hamlet"
  - Explicit reification
    - :n2 rdf:subject :Shakespeare
    - :n2 rdf:predicate :wrote
    - :n2 rdf:object :Hamlet
    - :n2 :attributedTo :Wikipedia

What about higher-order relationships?

- “Wikipedia states Shakespeare wrote Hamlet in 1604”
- Consider the simpler “Wikipedia states Shakespeare wrote Hamlet"
  - Explicit reification
    - :n2 rdf:subject :Shakespeare
    - :n2 rdf:predicate :wrote
    - :n2 rdf:object :Hamlet
    - :n2 :attributedTo :Wikipedia
  - Reification is explicit because n2 represents a statement (not an object)

What about higher-order relationships?

- “Wikipedia states Shakespeare wrote Hamlet in 1604”
- Consider the simpler “Wikipedia states Shakespeare wrote Hamlet"
  - Explicit reification
    - :n2 rdf:subject :Shakespeare
    - :n2 rdf:predicate :wrote
    - :n2 rdf:object :Hamlet
    - :n2 :attributedTo :Wikipedia
  - Reification is explicit because n2 represents a statement (not an object)
  - Similarly, we could construct another statement n3 (“Wikipedia states Hamlet was written in 1604”)

Modern uses of Semantic Web Technologies

Google Knowledge Graph

Google Knowledge Graph

24 languages

Article Talk | Read Edit View history Tools

From Wikipedia, the free encyclopedia

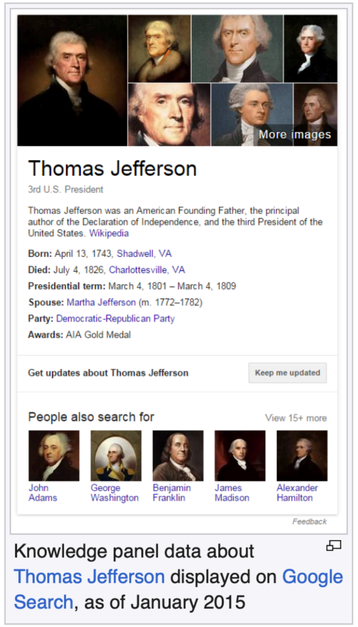
"Knowledge Graph" redirects here. For the general concept in information science, see Knowledge graph. For other uses, see Knowledge Graph (disambiguation).

A request that this article title be changed to *Knowledge Graph* is under discussion.
Please *do not move* this article until the discussion is closed.

The **Google Knowledge Graph** is a knowledge base from which Google serves relevant information in an infobox beside its search results. This allows the user to see the answer in a glance, as an instant answer. The data is generated automatically from a variety of sources, covering places, people, businesses, and more.[1][2]

The information covered by Google's Knowledge Graph grew quickly after launch, tripling its data size within seven months (covering 570 million entities and 18 billion facts[3]). By mid-2016, Google reported that it held 70 billion facts[4] and answered "roughly one-third" of the 100 billion monthly searches they handled. By May 2020, this had grown to 500 billion facts on 5 billion entities.[5]

There is no official documentation of how the Google Knowledge Graph is implemented.[6] According to Google, its information is retrieved from many sources, including the *CIA World Factbook* and Wikipedia.[7] It is used to answer direct spoken questions in Google Assistant[8][9] and Google Home voice queries.[10] It has been criticized for providing answers with neither source attribution nor citations.[11]



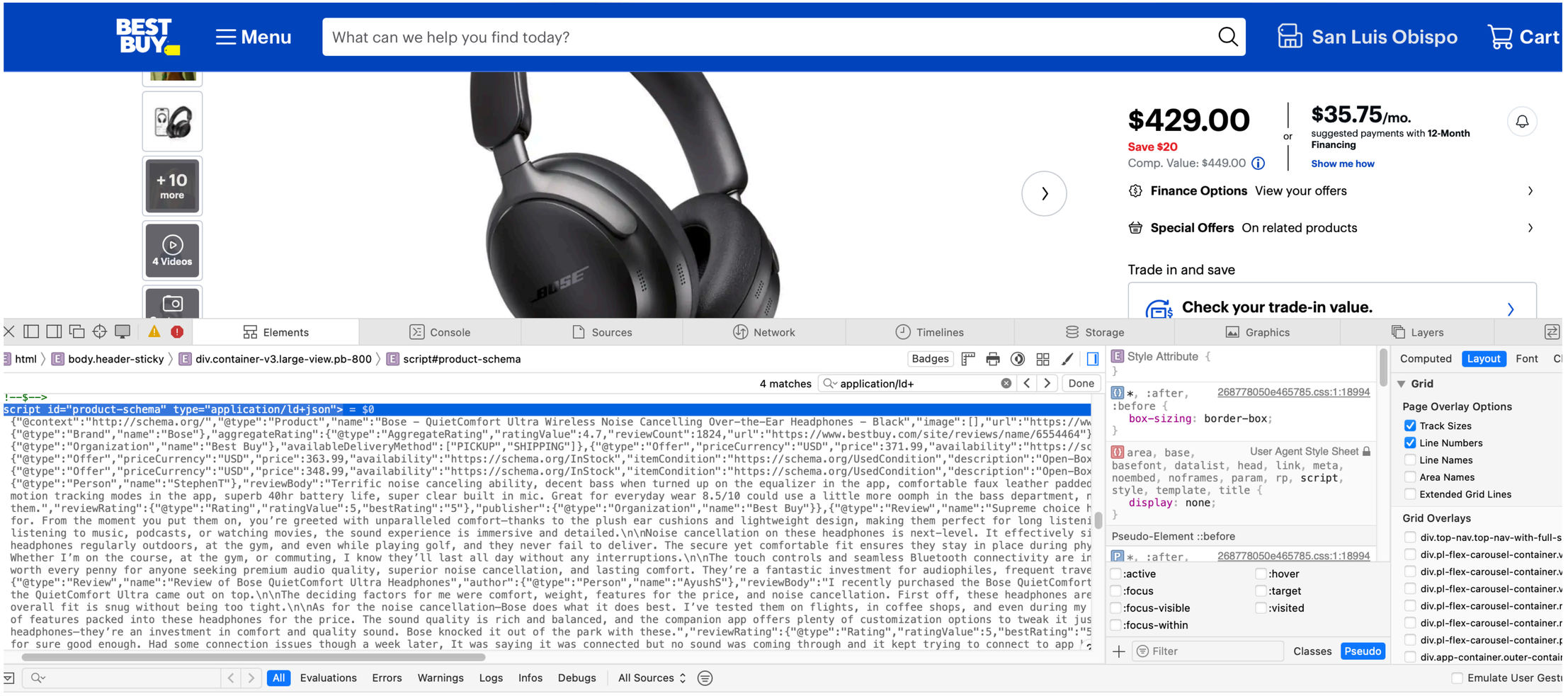
Knowledge panel data about Thomas Jefferson displayed on Google Search, as of January 2015

History [edit]

Google announced its Knowledge Graph on May 16, 2012, as a way to significantly enhance the value of information returned by Google searches.[7] Initially available only in English, it was expanded in December 2012 to Spanish, French, German, Portuguese, Japanese, Russian and Italian.[12] Bengali support was added in March 2017.[13]

Source: Wikipedia (access May 14, 2025)

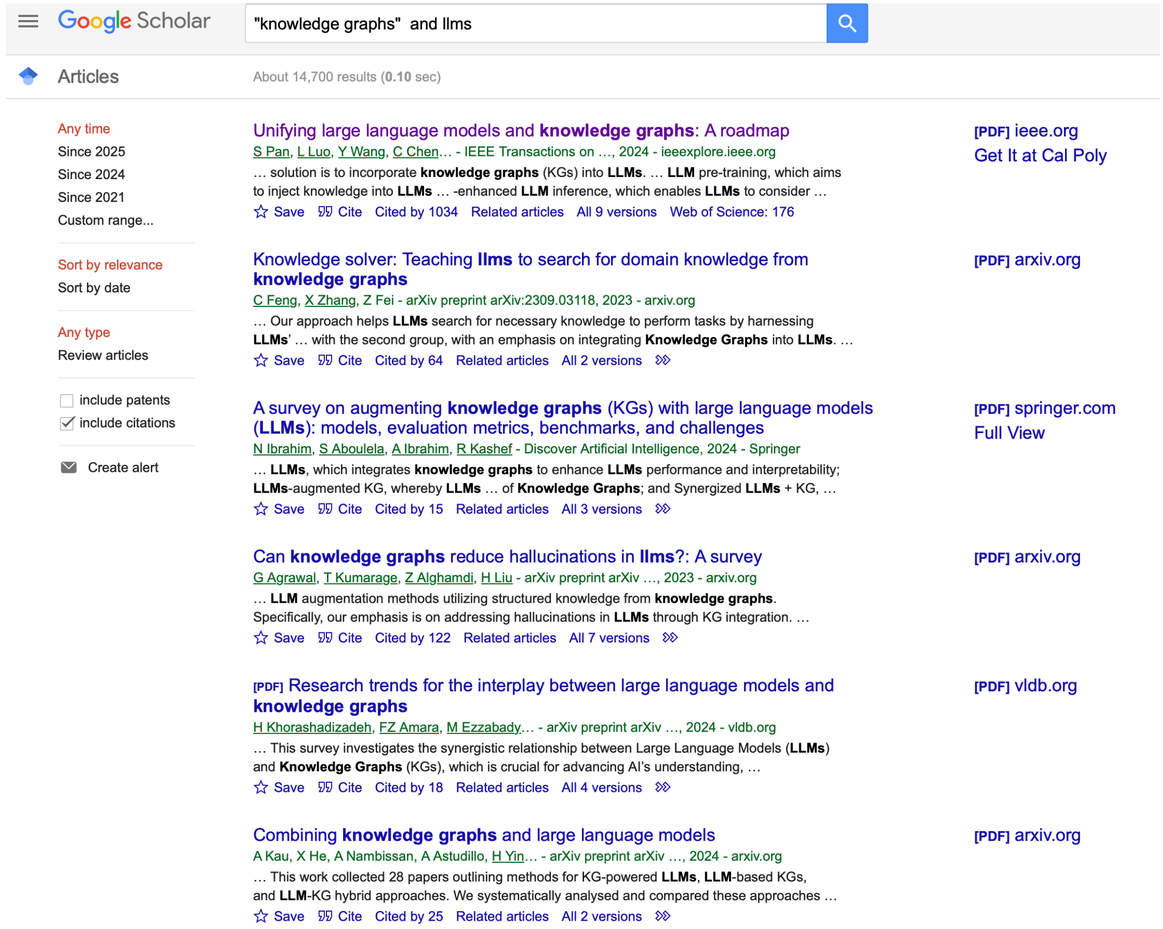
Tagging in online shopping



Source: BestBuy (access May 14, 2025)

LLMs and Knowledge Graphs

Modern research on integration (both ways) between KGs and LLMs



Source: Google Scholar (access May 14, 2025)

Other large projects

Other large projects

- dbpedia.org - open source project extracting structured content from Wikimedia. One of the largest triple stores

Other large projects

- dbpedia.org - open source project extracting structured content from Wikimedia. One of the largest triple stores
- Wikidata Query Service - a SPARQL endpoint for queries on wikidata

Other large projects

- dbpedia.org - open source project extracting structured content from Wikimedia. One of the largest triple stores
- Wikidata Query Service - a SPARQL endpoint for queries on wikidata
- schema.org - A community-maintained ontology founded by Google, Microsoft, Yahoo and Yandex

Other large projects

- dbpedia.org - open source project extracting structured content from Wikimedia. One of the largest triple stores
- Wikidata Query Service - a SPARQL endpoint for queries on wikidata
- schema.org - A community-maintained ontology founded by Google, Microsoft, Yahoo and Yandex
- LOD (Linked Open Data) Cloud - lod-cloud.net

Other large projects

- dbpedia.org - open source project extracting structured content from Wikimedia. One of the largest triple stores
- Wikidata Query Service - a SPARQL endpoint for queries on wikidata
- schema.org - A community-maintained ontology founded by Google, Microsoft, Yahoo and Yandex
- LOD (Linked Open Data) Cloud - lod-cloud.net
- Biomedical databases:

Other large projects

- dbpedia.org - open source project extracting structured content from Wikimedia. One of the largest triple stores
- Wikidata Query Service - a SPARQL endpoint for queries on wikidata
- schema.org - A community-maintained ontology founded by Google, Microsoft, Yahoo and Yandex
- LOD (Linked Open Data) Cloud - lod-cloud.net
- Biomedical databases:
  - https://geneontology.org/

Turtle notation (serialization)

Serialization

Serialization

- How to represent triples on text (or in a file)?

Serialization

- How to represent triples on text (or in a file)?
- Fully writing down each triple (N-Triples) is clunky and not very readable

Serialization

- How to represent triples on text (or in a file)?
- Fully writing down each triple (N-Triples) is clunky and not very readable
- Turtle format is particularly suited for human readability

Serialization

- How to represent triples on text (or in a file)?
- Fully writing down each triple (N-Triples) is clunky and not very readable
- Turtle format is particularly suited for human readability
  - Our focus here

Serialization

- How to represent triples on text (or in a file)?
- Fully writing down each triple (N-Triples) is clunky and not very readable
- Turtle format is particularly suited for human readability
  - Our focus here
- Other formats, such as RDF/XML are more appropriate for web applications

Serialization

- How to represent triples on text (or in a file)?
- Fully writing down each triple (N-Triples) is clunky and not very readable
- Turtle format is particularly suited for human readability
  - Our focus here
- Other formats, such as RDF/XML are more appropriate for web applications
  - Full specification available: https://www.w3.org/TR/rdf-syntax-grammar/

Serialization

- How to represent triples on text (or in a file)?
- Fully writing down each triple (N-Triples) is clunky and not very readable
- Turtle format is particularly suited for human readability
  - Our focus here
- Other formats, such as RDF/XML are more appropriate for web applications
  - Full specification available: https://www.w3.org/TR/rdf-syntax-grammar/
- Most triple stores can provide triples in multiple formats

Turtle

Turtle

- Preambles binds local names and global URIs

Turtle

- Preambles binds local names and global URIs

```
@prefix mfg:
<http://www.WorkingOntologist.com/Examples/Chapter3/Manufac
     turing#>
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

Turtle

- Preambles binds local names and global URIs

```
@prefix mfg:
<http://www.WorkingOntologist.com/Examples/Chapter3/Manufac
     turing#>
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

- Remaining triples can be expressed using the local names

Turtle

- Preambles binds local names and global URIs

```
@prefix mfg:
<http://www.WorkingOntologist.com/Examples/Chapter3/Manufac
     turing#>
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

- Remaining triples can be expressed using the local names
  - Triples usually terminate with a period

Turtle

- Preambles binds local names and global URIs

```
@prefix mfg:
<http://www.WorkingOntologist.com/Examples/Chapter3/Manufac
     turing#>
@prefix rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
```

- Remaining triples can be expressed using the local names
  - Triples usually terminate with a period

```
mfg:Product1 rdf:type mfg:Product .
```

Turtle

- Commas ‘,’ separate triples with repeat subjects AND predicate

Turtle

- Commas ‘,’ separate triples with repeat subjects AND predicate

```
lit:Shakespeare b:hasChild b:Susanna, b:Judith, b:Hamnet.
```

There are actually three triples represented here—namely:

```
lit:Shakespeare b:hasChild b:Susanna.
lit:Shakespeare b:hasChild b:Judith.
lit:Shakespeare b:hasChild b:Hamnet.
```

Turtle

Turtle

- rdf:type is often abbreviated by *a*

Turtle

- rdf:type is often abbreviated by *a*
  - Rather than say “X has type Y”, say “X is a Y”

Turtle

- rdf:type is often abbreviated by *a*
  - Rather than say “X has type Y”, say “X is a Y”

```
mfg:Product1 rdf:type mfg:Product.
lit:Shakespeare rdf:type lit:Playwright.
```

Turtle

- rdf:type is often abbreviated by *a*
  - Rather than say “X has type Y”, say “X is a Y”

```
mfg:Product1 rdf:type mfg:Product.
lit:Shakespeare rdf:type lit:Playwright.
```

```
mfg:Product1 a mfg:Product.
lit:Shakespeare a lit:Playwright.
```

Turtle

Blank Nodes (“bnodes”)

Turtle

Blank Nodes (“bnodes”)

- What if we don’t know the subject of a triple?

Turtle

Blank Nodes (“bnodes”)

- What if we don’t know the subject of a triple?
  - e.g. existential quantifier in FOL

Turtle

Blank Nodes (“bnodes”)

- What if we don’t know the subject of a triple?
  - e.g. existential quantifier in FOL
- We can always create a “dummy” identifiar

Turtle

Blank Nodes (“bnodes”)

- What if we don’t know the subject of a triple?
  - e.g. existential quantifier in FOL
- We can always create a “dummy” identifiar
- Alternatively, we can express with bracket notation:

Turtle

Blank Nodes (“bnodes”)

- What if we don’t know the subject of a triple?
  - e.g. existential quantifier in FOL
- We can always create a “dummy” identifiar
- Alternatively, we can express with bracket notation:

```
lit:Sonnet78 lit:hasInspiration [a :Woman;
                                bio:livedIn geo:England].
```

Turtle

Blank Nodes (“bnodes”)

- What if we don’t know the subject of a triple?
  - e.g. existential quantifier in FOL
- We can always create a “dummy” identifiar
- Alternatively, we can express with bracket notation:

```
lit:Sonnet78 lit:hasInspiration [a :Woman;
                                bio:livedIn geo:England].
```

- Due to lack of a UID, need to copy the whole bracketed statement

Summary

- RDF expresses data as triples
- Two elements have the same identity if they share the same URI
- RDF triples can be viewed as a graph
- To merge data, simply add up all triples from all sources
  - No need to worry about missing entries or matching up columns from tables

Key Concepts

RDF (Resource Description Framework)—This distributes data on the Web.
Triple—The fundamental data structure of RDF. A triple is made up of a subject, predicate, and object.
Graph—A nodes-and-links structural view of RDF data.
Merging—The process of treating two graphs as if they were one.
URI (Uniform Resource Indicator)—A generalization of the URL (Uniform Resource Locator), which is the global name on the Web.
namespace—A set of names that belongs to a single authority. Namespaces allow different agents to use the same word in different ways.
qname—An abbreviated version of a URI, it is made up of a namespace identifier and a name, separated by a colon.
rdf:type—The relationship between an instance and its type.
rdf:Property—The type of any property in RDF.
Reification—The practice of making a statement about another statement. It is done in RDF using `rdf:subject`, `rdf:predicate`, and `rdf:object`.
N-Triples, Turtle, RDF/XML—The serialization syntaxes for RDF.
Blank nodes—RDF nodes that have no URI and thus cannot be referenced globally. They are used to stand in for anonymous entities.