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

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

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

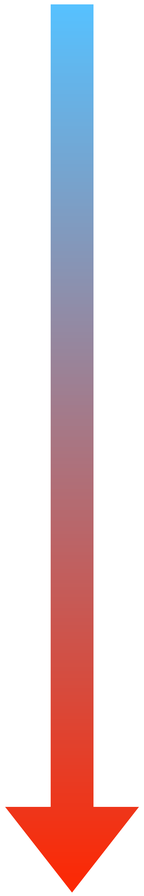
Main considerations for semantic web

- 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

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)

How to distribute data?

Combines the downsides of both previous approaches!

Table 3.1 Tabular Data about Elizabethan Literature and Mu

ID	Title	Author
1	*As You Like It*	Shakespeare
2	*Hamlet*	Shakespeare
3	*Othello*	Shakespeare
4	"Sonnet 78"	Shakespeare
5	*Astrophil and Stella*	Sir Phillip Sidney
6	*Edward II*	Christopher Marlowe
7	*Hero and Leander*	Christopher Marlowe
8	*Greensleeves*	Henry VIII Rex

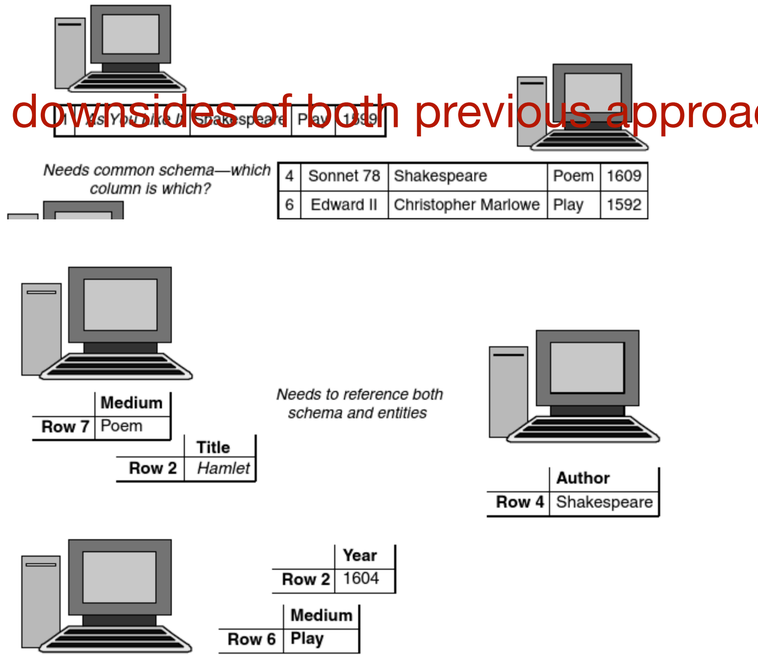


FIGURE 3.3

Distributing data across the Web, cell by cell.

But is the choice for RDF (AAA slogan)

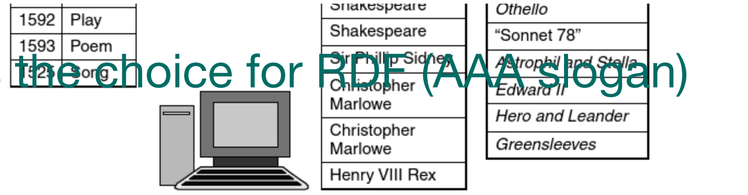


FIGURE 3.2

Distributing data across the Web, column by column.

Triplets 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

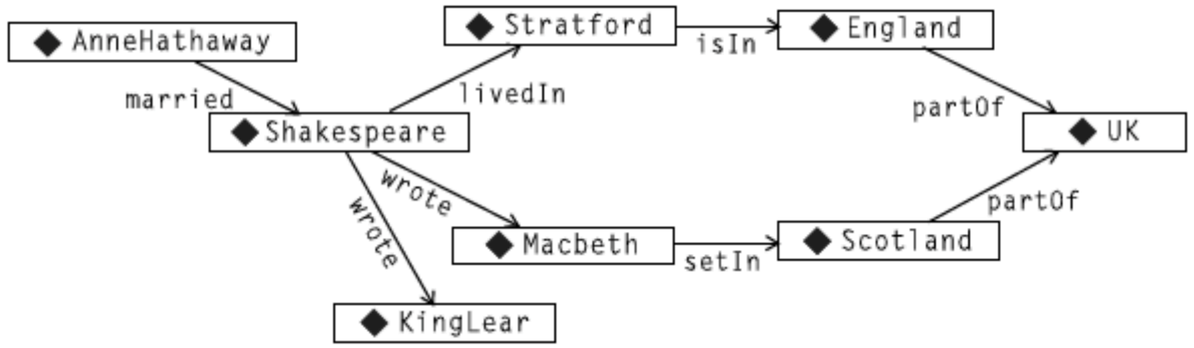


FIGURE 3.4

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*

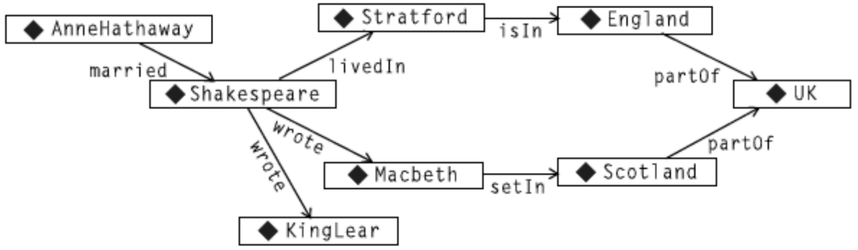
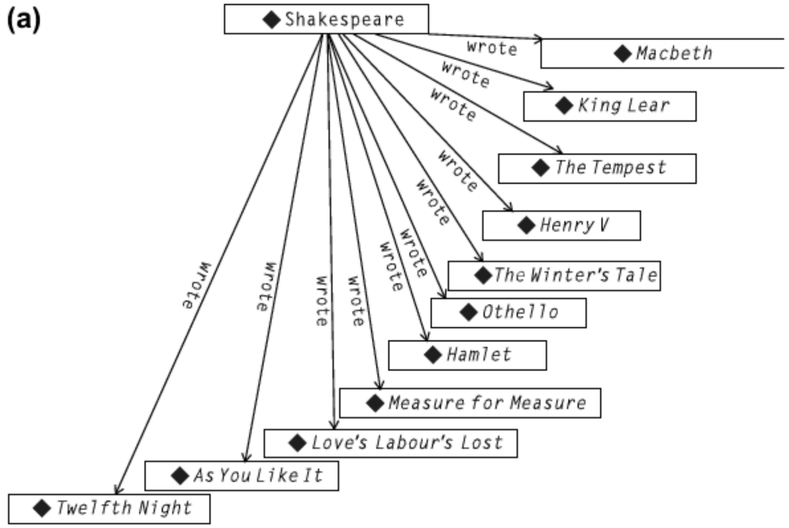


FIGURE 3.4

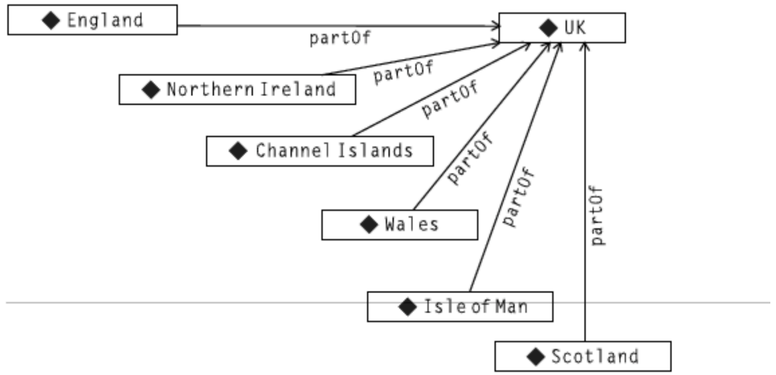
Graph display of triples from Table 3.3. Eight triples appear as eight labeled edges.

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

(a)



(b)



Merging data from multiple sources

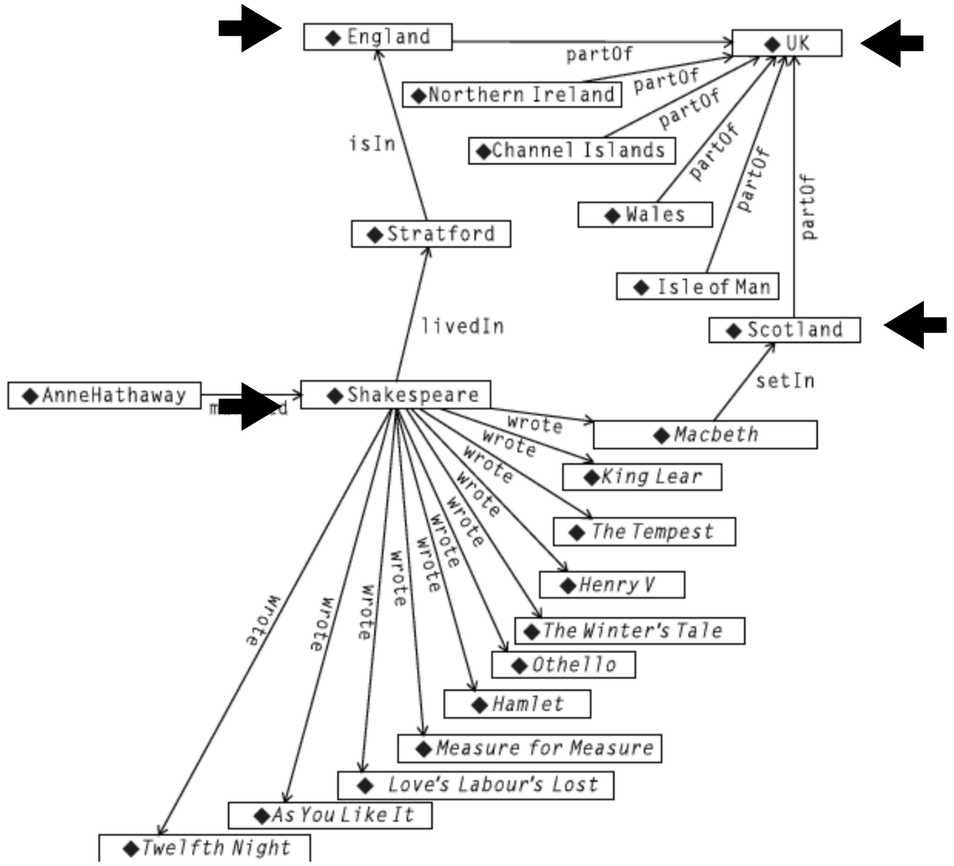


FIGURE 3.6

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?

- 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 is called a “resource”
- Two resources are the same if they share the same **Uniform Resource Identifier (URI)**

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

- 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

- 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 pre

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 predicates

rdf:property

- Indicates that something is to be used as a predicate rather than as a subject or object

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

- 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

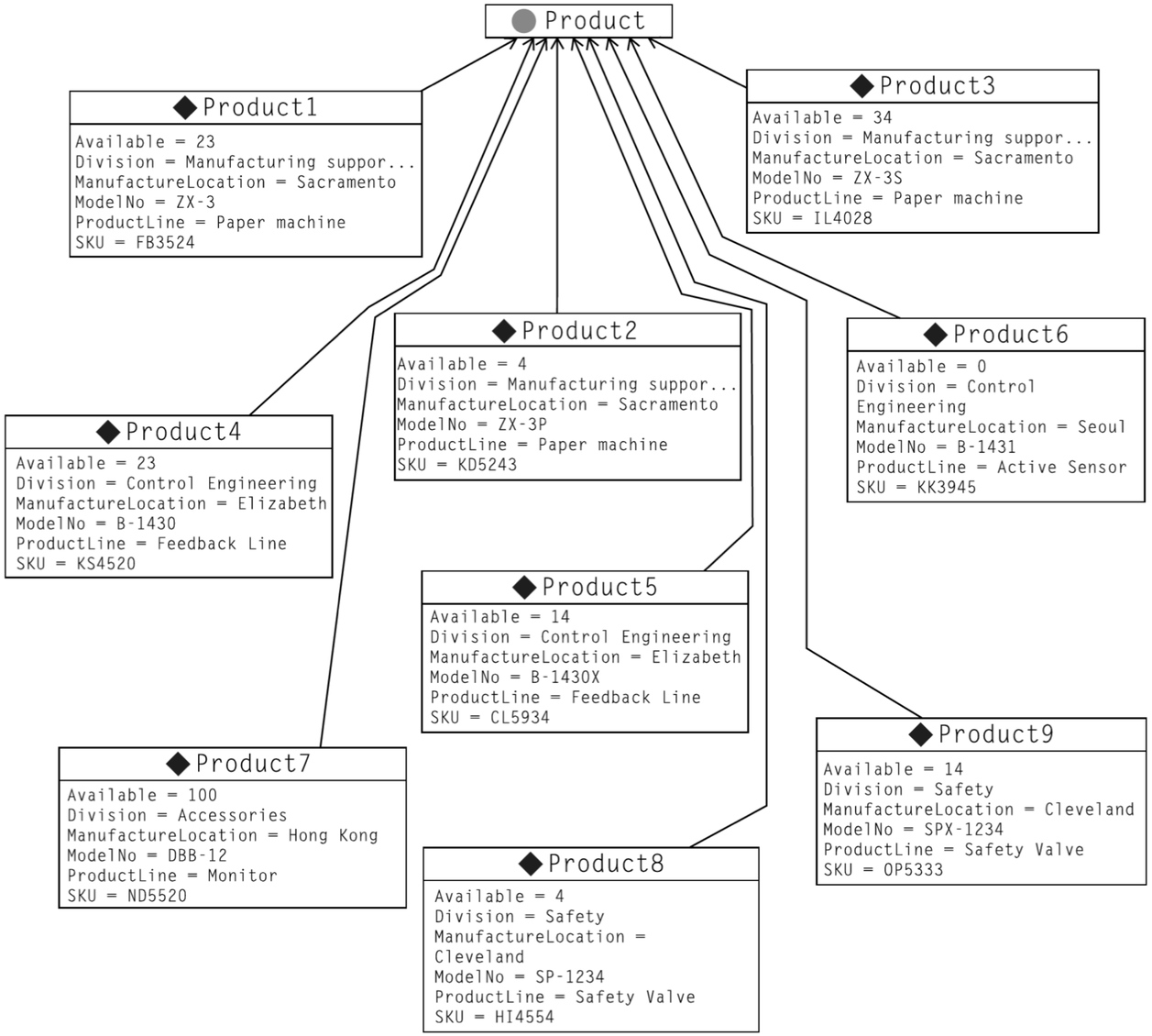
- For each line, we state that element to be of the appropriate type
  - e.g. mfg:Product1 rdf:type mfg:product
- Each column becomes a property, each cell becomes a triple
  - e.g. mfg:Product1 mfg:Product_ModelNo “ZX-3”
- Object of each triple may be a literal data type (Integer, String) rather than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

RDF and Tabular Data

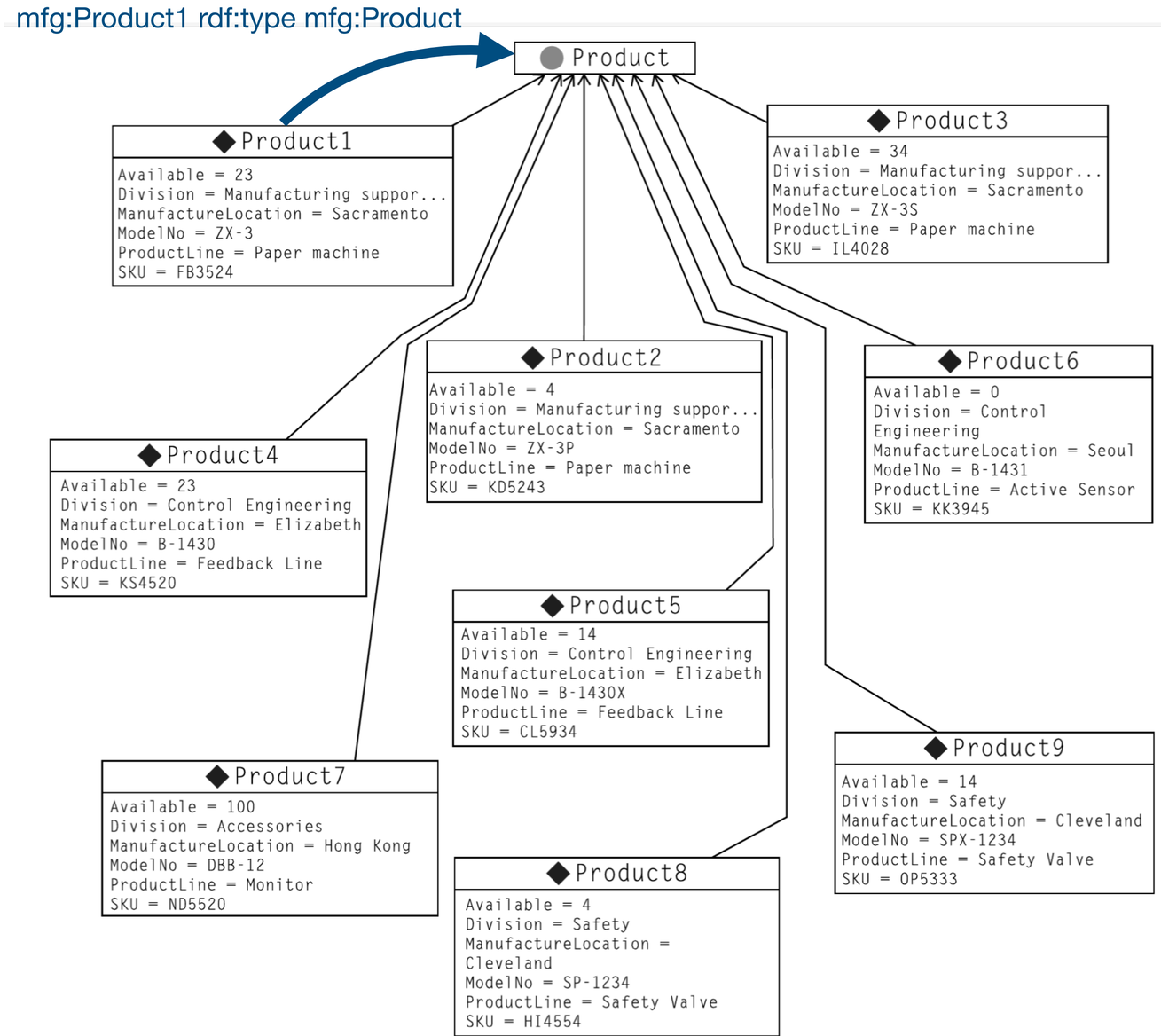
Table 3.13 Triples Representing Some of the Data in Table 3.12

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…

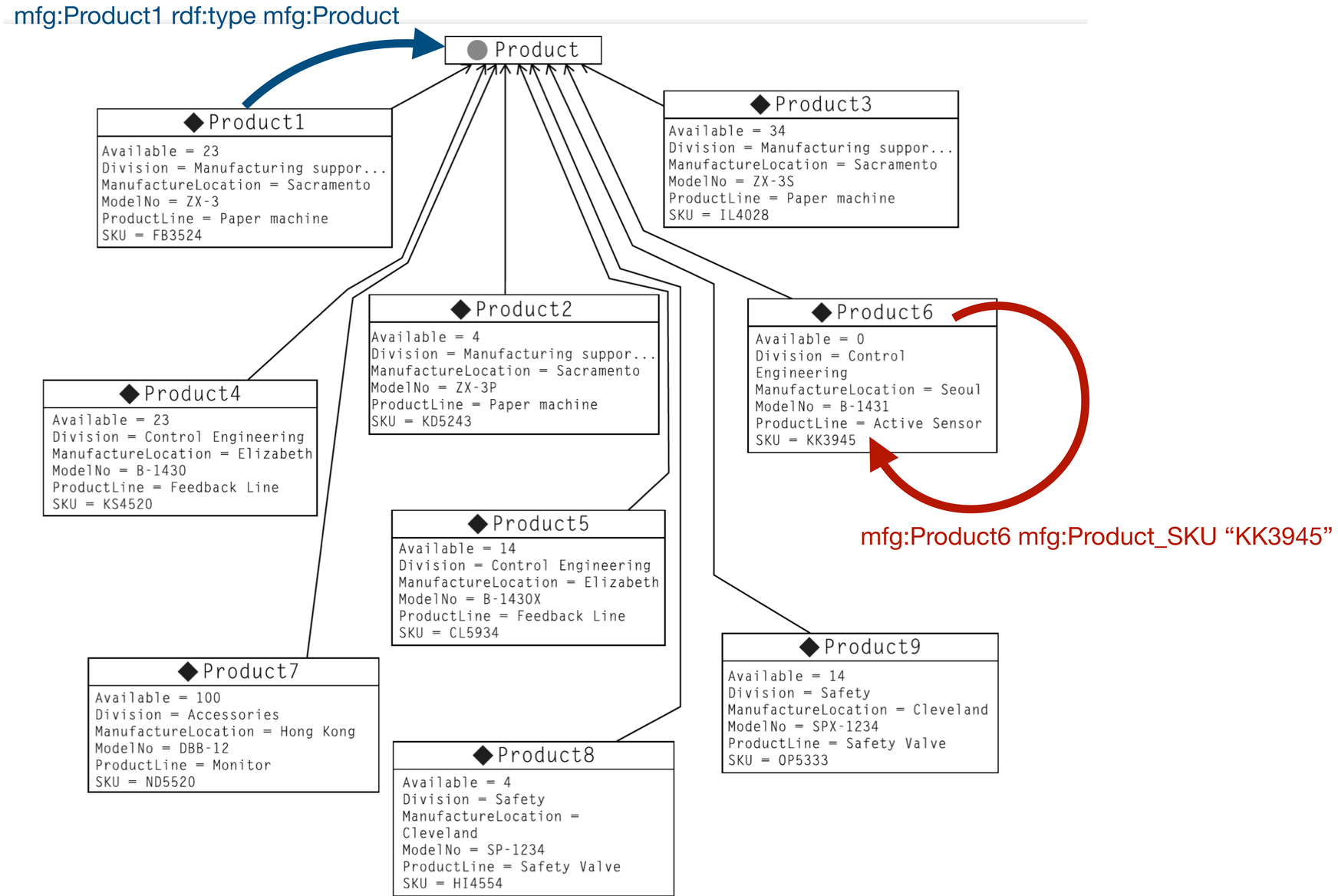
RDF and Tabular Data



RDF and Tabular Data



RDF and Tabular Data



What about higher-order relationships?

- 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?

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

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?

- “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”)

Turtle notation (serialization)

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/

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

```
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

- 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”)

- 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.