Knowledge Graphs definitions

- Knowledge base

  = technology used to store complex
  structured/unstructured **information**

- stores answers to questions or solutions to problems

- Vs. Database:

  collection of **data** representing facts

Knowledge Graphs Intro

- What is a knowledge graph:
  = a **network** of real-world entities (objects, events, concepts) and their **relationships**. Commonly stored in a graph database and visualized as a graph structure (a.k.a. semantic network)

- What is the Semantic Web:
  = an extension of the current web in which information is given **well-defined meaning**, better enabling computers and people to work in cooperation.

How to make sense of it

- Go from ANY (informal) representation to a **formal** model
- Connect information, but adhere to model (stay **consistent**)
- Further **Distribute** information (www)

Technologies

- Semantic technologies enable people to:
  - Create data stores
  - Build vocabularies
  - Write rules for handling data
- We will be looking at RDF, SPARQL and OWL
- We want to give the data further meaning!

Why is Meaning so important?

- Once a computer ‘understands‘ what a thing (person, event, place etc.) is, it can help user interact with these:
- I understand a date! => add to calendar
- I like this type of music! => here are some suggestions

- Search engine:
  difference between plain **keywords** and their actual **meaning** when browsing would be a game-changer

Non-unique Naming Assumption

- Train
  = working out
  = public transportation
  = machine learning

- It is evident that you can have multiple names/definitions for a thing
- The computer needs to know this,
  and *by needs to know*, we mean **EXPLICITLY**
- This is handled with uniform resource identifiers
- AND VERY STRICT DEFINITIONS

RDF & URIs

- The RDF graph is based on the idea that every data item should have a unique (Web?) identifier **(Uniform Resource Identifier),**

and that

- **every** data item *can be* connected to every other item.

- A URI is different from a URL (Uniform Resource Locator) in that a URI identifies a resource and differentiates it from others (may refer to either a name, for example); a URL may refer only to actual Web locations

Resource Description Framework

- A model for representing **metadata**
- A model for encoding **semantic relationships** between items of data so that these relationships can be interpreted computationally.
- A general method to decompose knowledge into **small pieces** with **rules about the meaning** of those pieces.
- A method to describe facts in a short form.
- Everything is a Resource in the form of a URI*

*there tends to be an(or more) exception(s)
– keep it in the back of your minds and let me know if it comes up

RDF Pros and Cons

+ Suitable for machines

+ Parsing is easy

~ Hard to validate (semantics?, OWA?)

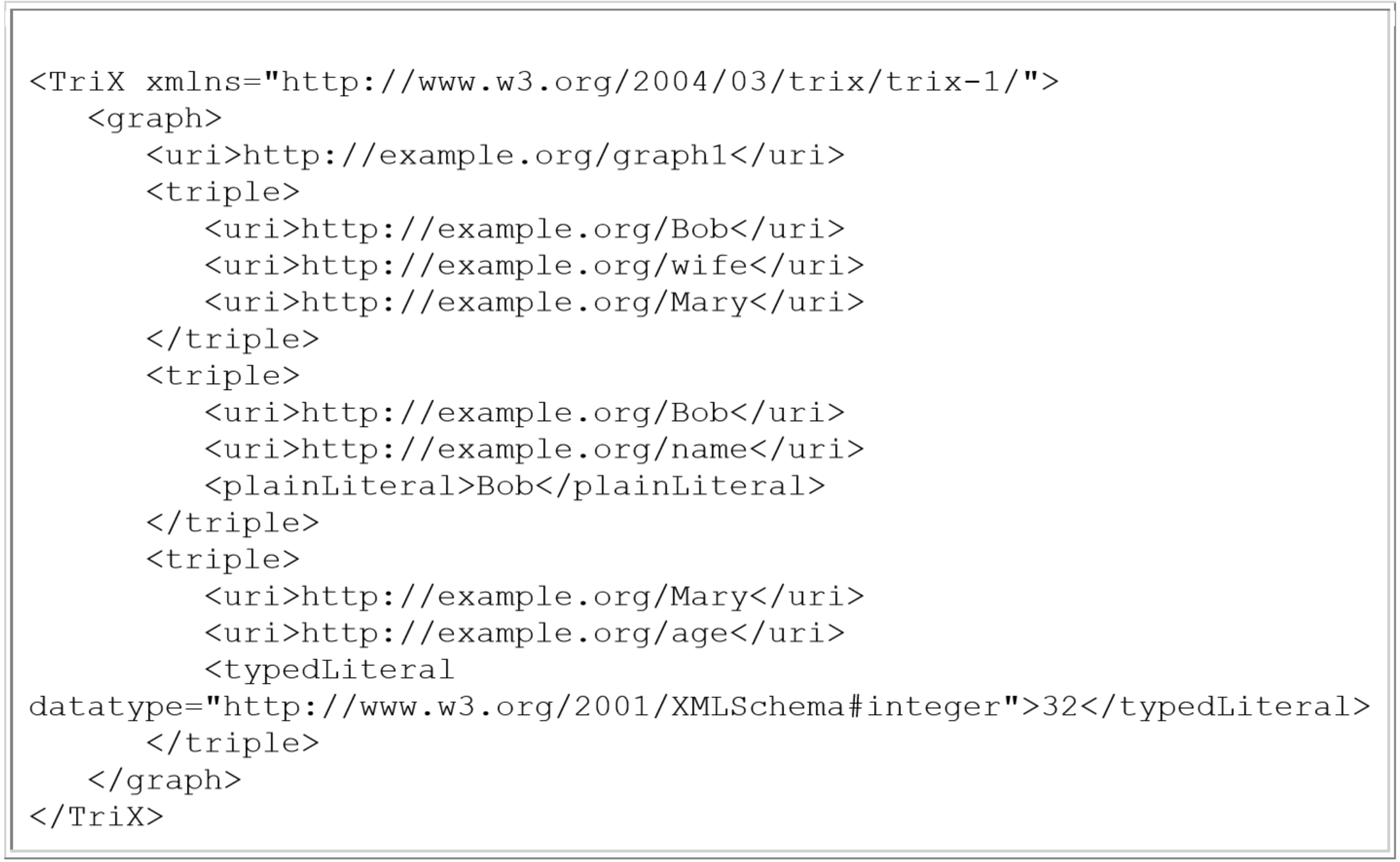
- Difficult for humans to see the pattern
(subject-predicate-object triples)

TriX Notation

- Added the ability to name graphs, noting that in practice this is already widely used
- https://www.hpl.hp.com/techreports/2004/HPL-2004-56.pdf

TriX example

- Any Observations?



```
<TriX xmlns="http://www.w3.org/2004/03/trix/trix-1/">
   <graph>
      <uri>http://example.org/graph1</uri>
      <triple>
         <uri>http://example.org/Bob</uri>
         <uri>http://example.org/wife</uri>
         <uri>http://example.org/Mary</uri>
      </triple>
      <triple>
         <uri>http://example.org/Bob</uri>
         <uri>http://example.org/name</uri>
         <plainLiteral>Bob</plainLiteral>
      </triple>
      <triple>
         <uri>http://example.org/Mary</uri>
         <uri>http://example.org/age</uri>
         <typedLiteral
datatype="http://www.w3.org/2001/XMLSchema#integer">32</typedLiteral>
      </triple>
   </graph>
</TriX>
```

* URI for graph Is optional

Use of meta data
(any hints from the TriX example?)

- Semantic Web data formats were designed from the ground up as purpose-built languages for

  metadata

  (a way to accurately describe data by using more data)

- In business software systems, these new formats provide a way to more easily exchange data across systems, and new ways to model complex data environments that can be more simply maintained over time

N-Triples

- As close to raw RDF triples as possible
- Uses fully unabbreviated URIs (yes, you can have “Literals”)
- URIs written between angle brackets (< and >)
- Three resources are expressed in subject/predicate/object order, followed by a period (.)
- For example,
  <http://www....org/Examples/Chapter3Manufacture.rdf#Product1>
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  <http://www....org/Examples/Chapter3Manufacture.rdf#Product>
  .

N3 (Notation 3 RDF)

- Compact serialization of RDF
- Combines ntriples and qnames.
- Example:

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

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

N3 - The semicolon

- Often multiple triples share a common subject. N3 provides for a compact representation of such data.
- It begins with the first triple in subject/predicate/object order but does not terminate (with a period)
- Instead, it uses a semicolon (;) to indicate that another triple with the **same subject** follows. For that triple, only the predicate and object need to be specified

```
mfg:Product1 rdf:type mfg:Product;
   mfg:Product_Division "Manufacturing support";
   mfg:Product_ID "1";
   mfg:Product_Manufacture_Location "Sacramento";
   mfg:Product_ModelNo "ZX-3";
   mfg:Product_Product_Line "Paper Machine";
   mfg:Product_SKU "FB3524";
   mfg:Product_Available "23."
```

Turtles

- Turtle is similar to N-triples (and N3 for that matter), but even more compact
- Uses @prefix to define the prefix and later on uses qualified names

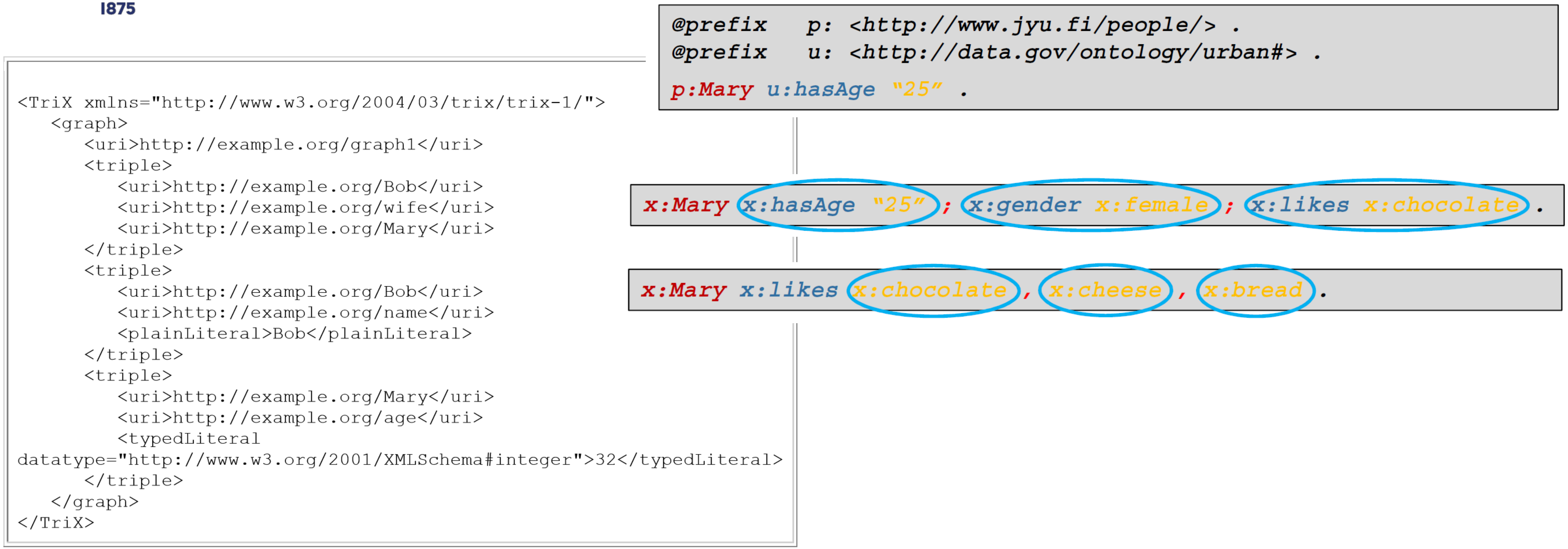
```
@prefix   p: <http://www.jyu.fi/people/> .
@prefix   u: <http://data.gov/ontology/urban#> .
p:Mary u:hasAge "25" .
```

- Abbreviated form of triples:
- – Semicolon (;) to separate statements about the same subject
- – Comma (,) about the same subject **with the same predicate**

```
x:Mary x:hasAge "25" ; x:gender x:female ; x:likes x:chocolate .
```

```
x:Mary x:likes x:chocolate , x:cheese , x:bread .
```

Is the conversion easy? Discuss!



```
@prefix   p: <http://www.jyu.fi/people/> .
@prefix   u: <http://data.gov/ontology/urban#> .
p:Mary u:hasAge "25" .
```

```
x:Mary x:hasAge "25" ; x:gender x:female ; x:likes x:chocolate .
```

```
x:Mary x:likes x:chocolate , x:cheese , x:bread .
```

```
<TriX xmlns="http://www.w3.org/2004/03/trix/trix-1/">
   <graph>
      <uri>http://example.org/graph1</uri>
      <triple>
         <uri>http://example.org/Bob</uri>
         <uri>http://example.org/wife</uri>
         <uri>http://example.org/Mary</uri>
      </triple>
      <triple>
         <uri>http://example.org/Bob</uri>
         <uri>http://example.org/name</uri>
         <plainLiteral>Bob</plainLiteral>
      </triple>
      <triple>
         <uri>http://example.org/Mary</uri>
         <uri>http://example.org/age</uri>
         <typedLiteral
datatype="http://www.w3.org/2001/XMLSchema#integer">32</typedLiteral>
      </triple>
   </graph>
</TriX>
```