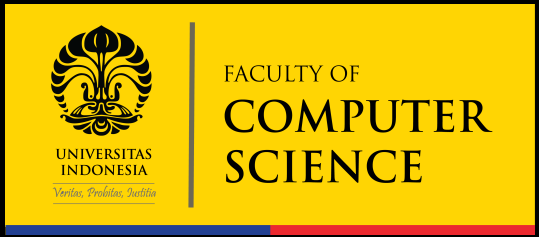


Semantic Web 03: RDF and the Issue of Identity

Adila Krisnadhi – adila@cs.ui.ac.id Faculty of Computer Science, Universitas Indonesia

Outline

Persistent identifiers: Motivation

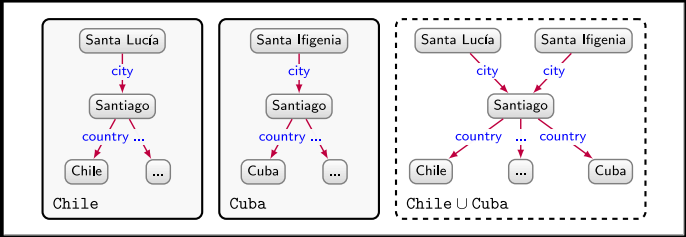
- Suppose we obtain two different graphs: one describes tourism in Chile, while the other describes tourism in Cuba.
- We can merge both graphs together, but ...

Persistent identifiers: Motivation

- Suppose we obtain two different graphs: one describes tourism in Chile, while the other describes tourism in Cuba.
- We can merge both graphs together, but … there might be a problem: both Chile and Cuba have a city named Santiago!

Persistent identifiers: Motivation

- Suppose we obtain two different graphs: one describes tourism in Chile, while the other describes tourism in Cuba.
- We can merge both graphs together, but … there might be a problem: both Chile and Cuba have a city named Santiago!



Persistent identifiers: Motivation

- Using an **ambiguous** node may lead to naming clash.

Persistent identifiers: Motivation

- Using an **ambiguous** node may lead to naming clash.
- Solution: use long lasting, globally unique, persistent identifiers (PIDs).

Persistent identifiers: Motivation

- Using an **ambiguous** node may lead to naming clash.
- Solution: use long lasting, globally unique, persistent identifiers (PIDs).
  - Digital Object Identifiers (DOIs) for papers,

Persistent identifiers: Motivation

- Using an **ambiguous** node may lead to naming clash.
- Solution: use long lasting, globally unique, persistent identifiers (PIDs).
  - Digital Object Identifiers (DOIs) for papers,
  - ORCID IDs for researchers,

Persistent identifiers: Motivation

- Using an **ambiguous** node may lead to naming clash.
- Solution: use long lasting, globally unique, persistent identifiers (PIDs).
  - Digital Object Identifiers (DOIs) for papers,
  - ORCID IDs for researchers,
  - International Standard Book Numbers (ISBNs) for books,

Persistent identifiers: Motivation

- Using an **ambiguous** node may lead to naming clash.
- Solution: use long lasting, globally unique, persistent identifiers (PIDs).
  - Digital Object Identifiers (DOIs) for papers,
  - ORCID IDs for researchers,
  - International Standard Book Numbers (ISBNs) for books,
  - Alpha-2 codes for countries,

Persistent identifiers: Motivation

- Using an **ambiguous** node may lead to naming clash.
- Solution: use long lasting, globally unique, persistent identifiers (PIDs).
  - Digital Object Identifiers (DOIs) for papers,
  - ORCID IDs for researchers,
  - International Standard Book Numbers (ISBNs) for books,
  - Alpha-2 codes for countries,
  - for Semantic Web resources?

Can we use URL?

- For the Web, we already have Uniform Resource Locators (URLs):
  - identifies the location of **information resources** (i.e., web documents) such as webpages.

Can we use URL?

- For the Web, we already have Uniform Resource Locators (URLs):
  - identifies the location of **information resources** (i.e., web documents) such as webpages.

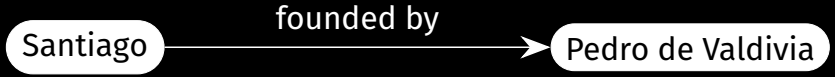
- However, URLs are not sufficient ...

Can we use URL?

- For the Web, we already have Uniform Resource Locators (URLs):
  - identifies the location of **information resources** (i.e., web documents) such as webpages.
- However, URLs are not sufficient ...
- Consider as an example, we would like to represent/store the information that “Santiago in Chile was founded by Pedro de Valdivia”, leading to the following graph:

Can we use URL?

- For the Web, we already have Uniform Resource Locators (URLs):
  - identifies the location of **information resources** (i.e., web documents) such as webpages.
- However, URLs are not sufficient ...
- Consider as an example, we would like to represent/store the information that "Santiago in Chile was founded by Pedro de Valdivia", leading to the following graph:



Santiago was founded by Pedro de Valdivia

Suppose the relevant URLs are as follows:

- URL for Santiago of Chile's **webpage** in Wikidata:
  `https://www.wikidata.org/wiki/Q2887`
- URL for Pedro de Valdivia's **webpage** in Wikidata:
  `https://www.wikidata.org/wiki/Q203534`
- URL for the **webpage** of the 'founded by' relation in Wikidata:
  `https://www.wikidata.org/wiki/Property:P112`

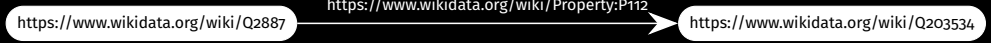
Then we have something like:

Santiago was founded by Pedro de Valdivia

Suppose the relevant URLs are as follows:

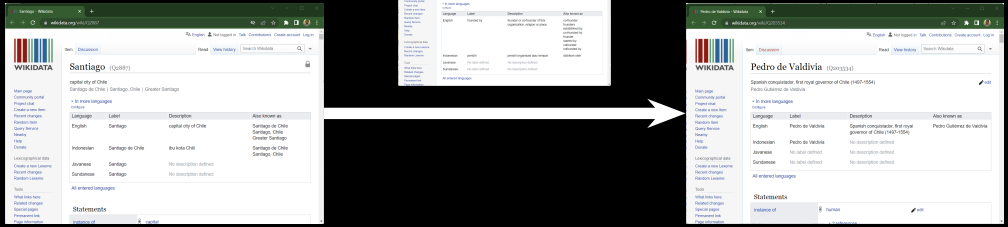
- URL for Santiago of Chile's **webpage** in Wikidata:
  `https://www.wikidata.org/wiki/Q2887`
- URL for Pedro de Valdivia's **webpage** in Wikidata:
  `https://www.wikidata.org/wiki/Q203534`
- URL for the **webpage** of the 'founded by' relation in Wikidata:
  `https://www.wikidata.org/wiki/Property:P112`

Then we have something like:

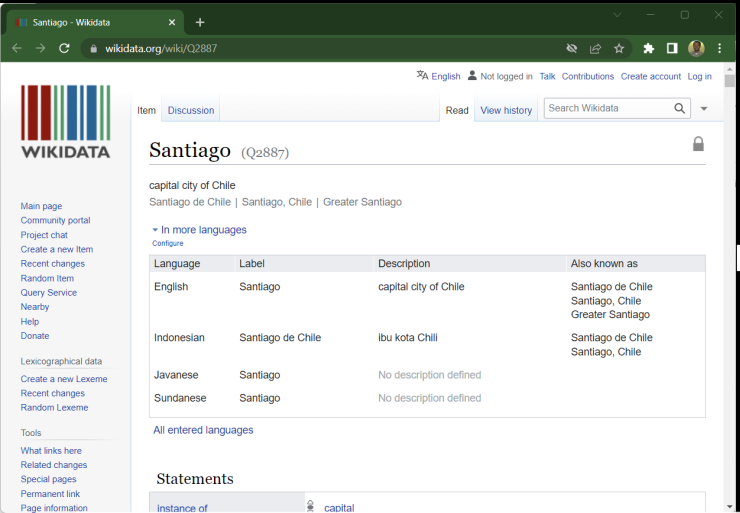


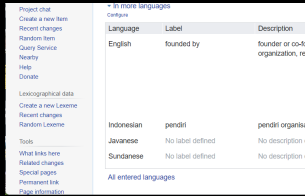
Santiago was founded by Pedro de Valdivia (cont.)

which actually represents the following relationship **between the two webpages** (not between a city and a person):



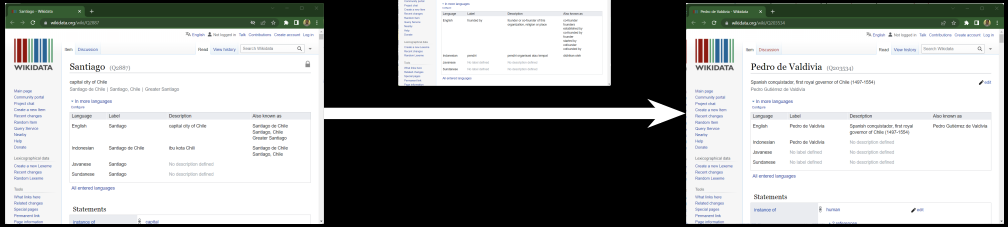
Santiago was founded by Pedro de Valdivia (cont.)



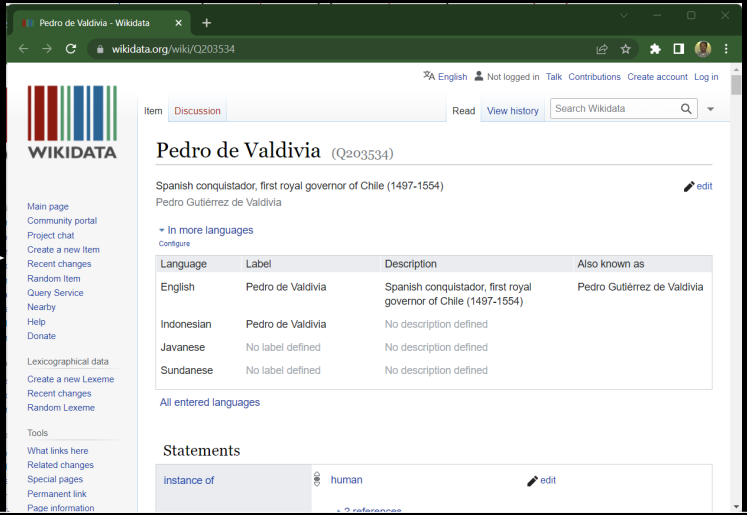


Santiago was founded by Pedro de Valdivia (cont.)

which actually represents the following relationship **between the two webpages** (not between a city and a person):

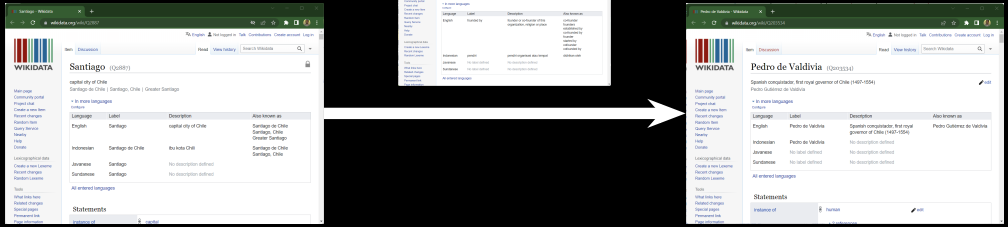


Santiago was founded by Pedro de Valdivia (cont.)



Santiago was founded by Pedro de Valdivia (cont.)

which actually represents the following relationship **between the two webpages** (not between a city and a person):



- What does it mean to have a webpage as a relation?
- Relationship is ambiguous: was Pedro de Valdivia the founder of the webpage or the city?

IRI

- For the Semantic Web, we extend the idea of URLs to the so-called Internationalized Resource Identifier (IRI) — used to be called Uniform Resource Identifier (URI).

IRI

- For the Semantic Web, we extend the idea of URLs to the so-called Internationalized Resource Identifier (IRI) — used to be called Uniform Resource Identifier (URI).
- An IRI is first and foremost an identifier, but in many cases, it is also a location on the Web.

IRI

- For the Semantic Web, we extend the idea of URLs to the so-called Internationalized Resource Identifier (IRI) — used to be called Uniform Resource Identifier (URI).
- An IRI is first and foremost an identifier, but in many cases, it is also a location on the Web.
- IRIs look similar to URLs; in fact, every URL is an IRI, because every URL **is** an identifier of some web document.

IRI

- For the Semantic Web, we extend the idea of URLs to the so-called Internationalized Resource Identifier (IRI) — used to be called Uniform Resource Identifier (URI).
- An IRI is first and foremost an identifier, but in many cases, it is also a location on the Web.
- IRIs look similar to URLs; in fact, every URL is an IRI, because every URL **is** an identifier of some web document.
- IRIs also include identifiers assigned to non-information resources, e.g., people, cities, organizations, etc. For example,

IRI

- For the Semantic Web, we extend the idea of URLs to the so-called Internationalized Resource Identifier (IRI) — used to be called Uniform Resource Identifier (URI).
- An IRI is first and foremost an identifier, but in many cases, it is also a location on the Web.
- IRIs look similar to URLs; in fact, every URL is an IRI, because every URL **is** an identifier of some web document.
- IRIs also include identifiers assigned to non-information resources, e.g., people, cities, organizations, etc. For example,
  - For Santiago the city, Wikidata uses `https://www.wikidata.org/entity/Q2887`

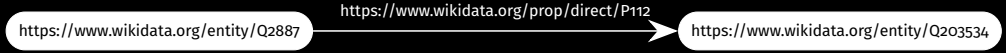
IRI

- For the Semantic Web, we extend the idea of URLs to the so-called Internationalized Resource Identifier (IRI) — used to be called Uniform Resource Identifier (URI).
- An IRI is first and foremost an identifier, but in many cases, it is also a location on the Web.
- IRIs look similar to URLs; in fact, every URL is an IRI, because every URL **is** an identifier of some web document.
- IRIs also include identifiers assigned to non-information resources, e.g., people, cities, organizations, etc. For example,
  - For Santiago the city, Wikidata uses `https://www.wikidata.org/entity/Q2887`
  - For Pedro de Valdivia the person, Wikidata uses `https://www.wikidata.org/wiki/Q203534`

IRI

- For the Semantic Web, we extend the idea of URLs to the so-called Internationalized Resource Identifier (IRI) — used to be called Uniform Resource Identifier (URI).
- An IRI is first and foremost an identifier, but in many cases, it is also a location on the Web.
- IRIs look similar to URLs; in fact, every URL is an IRI, because every URL **is** an identifier of some web document.
- IRIs also include identifiers assigned to non-information resources, e.g., people, cities, organizations, etc. For example,
  - For Santiago the city, Wikidata uses `https://www.wikidata.org/entity/Q2887`
  - For Pedro de Valdivia the person, Wikidata uses `https://www.wikidata.org/wiki/Q203534`
  - For the 'founded by' relationship, Wikidata uses `https://www.wikidata.org/prop/direct/P112`

The relationship between Santiago the city and Pedro de Valdivia is represented by:



IRI namespace

- The IRI scheme used to distinguish information and non-information resources is left to the information/data owner/maintainer so long as it conforms to the general scheme for IRI (see next slide).

IRI namespace

- The IRI scheme used to distinguish information and non-information resources is left to the information/data owner/maintainer so long as it conforms to the general scheme for IRI (see next slide).
- The prefix part of each IRI is known as a namespace, and we can define an abbreivation for them, e.g.,

IRI namespace

- The IRI scheme used to distinguish information and non-information resources is left to the information/data owner/maintainer so long as it conforms to the general scheme for IRI (see next slide).
- The prefix part of each IRI is known as a namespace, and we can define an abbreivation for them, e.g.,
  - `https://www.wikidata.org/entity/` is abbreviated as `wd:`

IRI namespace

- The IRI scheme used to distinguish information and non-information resources is left to the information/data owner/maintainer so long as it conforms to the general scheme for IRI (see next slide).
- The prefix part of each IRI is known as a namespace, and we can define an abbreivation for them, e.g.,
  - `https://www.wikidata.org/entity/` is abbreviated as `wd:`
  - `https://www.wikidata.org/prop/direct/` is abbreviated as `wdt:`

IRI namespace

- The IRI scheme used to distinguish information and non-information resources is left to the information/data owner/maintainer so long as it conforms to the general scheme for IRI (see next slide).
- The prefix part of each IRI is known as a namespace, and we can define an abbreivation for them, e.g.,
  - `https://www.wikidata.org/entity/` is abbreviated as `wd:`
  - `https://www.wikidata.org/prop/direct/` is abbreviated as `wdt:`
- So, “Santiago was founded by Pedro Valdivia” can be represented by:

IRI standard according to RFC 3987

`IRI ::= scheme:[//authority]path[?query][#fragment]`

- All parts outside ':', '/', '?', and '#' may use Unicode characters.
- Parts in square brackets are optional.
- scheme: the scheme classifying the type of IRI
  - E.g., `http`, `https`, `ftp`, `mailto`, `urn`, ...
  - Standardized by IANA (Internet Assigned Numbers Authority)
  - Case insensitive if using US-ASCII
- authority: host name, optionally with user and/or port info
  - E.g., `kgbook.org`, `john@example.com`, `example.org:8080`
  - host names using US-ASCII are case-insensitive.

IRI standard according to RFC 3987 (cont.)

- path: main part of IRIs organized hierarchically
  - E.g., `/etc/passwd`, `this/path/with/-:_ /is/../okay`
  - Case-sensitive, if using US-ASCII, and may be empty (e.g., in email address)
- query: optional part of the IRI that provides additional non-hierarchical information such as providing HTTP GET parameters
  - E.g., `q=Semantic+Web+book`
  - Case-sensitive if using US-ASCII
- fragment: provides a second level of identifying resources (different fragments mean different names even if they lead to the same document when retrieved in browser)
  - E.g., `#section1`
  - Case-sensitive if using US-ASCII

IRI examples

- https://en.wikipedia.org/wiki/Indonesia#History
- https://remote-lib.ui.ac.id:2196/doi/fullHtml/10.1145/3293318
- https://arxiv.org/pdf/1806.06478.pdf
- https://www.google.com/search?client=opera&q=knowledge+graph+zero+shot+learning&sourceid=opera&ie=UTF-8&oe=UTF-8
- mailto:adila@cs.ui.ac.id

IRI as identifier

- Using `http` or `https` for the scheme is preferred, because http/https IRIs can be resolved via the Web protocol, i.e., the HTTP protocol.
- If two IRIs are the same in all parts except that one uses `http` and the other uses `https`, then they are considered the same.
- IRIs should be persistent:
  - should always be live and point to the same resource forever
  - the query part should be empty.
- Persistent URL (PURL) services may be used to ensure persistence.
  - They offer redirects from a fixed central server to a particular location, which may be changed over time.
  - See `http://www.purlz.org/` and `https://w3id.org/` for more details

IRI is globally unique?

- The scheme part is standardized and a data owner picks one for the host machine in which the data would be hosted.

IRI is globally unique?

- The scheme part is standardized and a data owner picks one for the host machine in which the data would be hosted.
- The authority part, i.e., host name corresponds to a particular data owner (person, organization, etc.) who holds the only authority over the host machine (i.e., no other data owner in the world holds the authority for that machine).

IRI is globally unique?

- The scheme part is standardized and a data owner picks one for the host machine in which the data would be hosted.
- The authority part, i.e., host name corresponds to a particular data owner (person, organization, etc.) who holds the only authority over the host machine (i.e., no other data owner in the world holds the authority for that machine).
- The data owner also holds the only authority on how the rest of the IRI is defined.

IRI is globally unique?

- The scheme part is standardized and a data owner picks one for the host machine in which the data would be hosted.
- The authority part, i.e., host name corresponds to a particular data owner (person, organization, etc.) who holds the only authority over the host machine (i.e., no other data owner in the world holds the authority for that machine).
- The data owner also holds the only authority on how the rest of the IRI is defined.
  - Uniqueness is ensured if the data owner ensures that each resource (entity or relation) is assigned a unique combination of path (and fragment if employed).

IRI is globally unique?

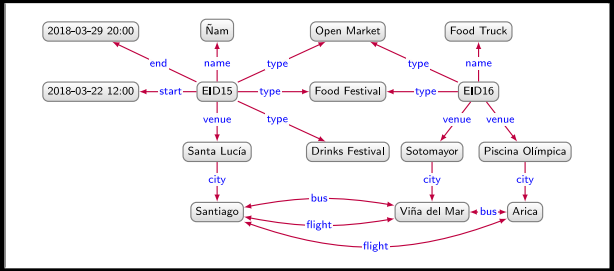
- The scheme part is standardized and a data owner picks one for the host machine in which the data would be hosted.
- The authority part, i.e., host name corresponds to a particular data owner (person, organization, etc.) who holds the only authority over the host machine (i.e., no other data owner in the world holds the authority for that machine).
- The data owner also holds the only authority on how the rest of the IRI is defined.
  - Uniqueness is ensured if the data owner ensures that each resource (entity or relation) is assigned a unique combination of path (and fragment if employed).
- Hence, an IRI is globally unique: each IRI corresponds to exactly one resource (entity/relation).

IRI is globally unique?

- The scheme part is standardized and a data owner picks one for the host machine in which the data would be hosted.
- The authority part, i.e., host name corresponds to a particular data owner (person, organization, etc.) who holds the only authority over the host machine (i.e., no other data owner in the world holds the authority for that machine).
- The data owner also holds the only authority on how the rest of the IRI is defined.
  - Uniqueness is ensured if the data owner ensures that each resource (entity or relation) is assigned a unique combination of path (and fragment if employed).
- Hence, an IRI is globally unique: each IRI corresponds to exactly one resource (entity/relation).
  - But, this does not prevent two IRIs to refer to the same resource.

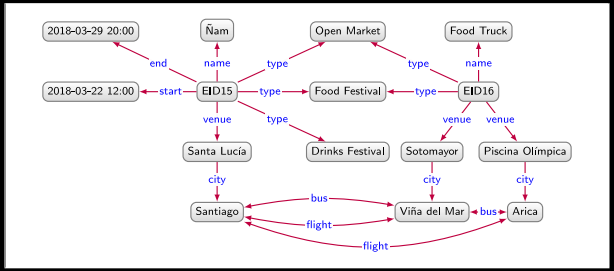
Outline

Motivation



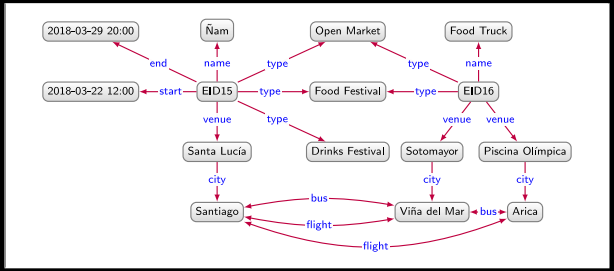
How should we assign a global/persistent identifier for the two date-times on the left?

Motivation



How should we assign a global/persistent identifier for the two date-times on the left? Assigning IRIs doesn't make sense because their syntactic form:

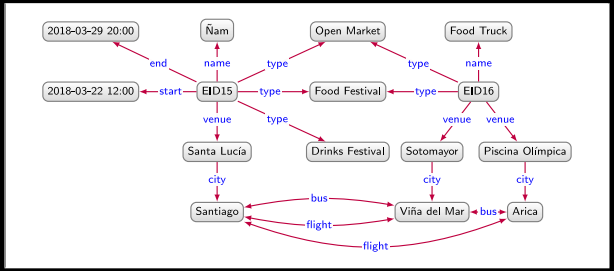
Motivation



How should we assign a global/persistent identifier for the two date-times on the left?
Assigning IRIs doesn't make sense because their syntactic form:

- directly indicates what they refer to: specific date and times in March 2020.

Motivation



How should we assign a global/persistent identifier for the two date-times on the left?
Assigning IRIs doesn't make sense because their syntactic form:

- directly indicates what they refer to: specific date and times in March 2020.
- is recognizable by machines — with appropriate software, we could sort them, extract the year, etc.

Datatypes

- Solution: take the node's value as a literal with a certain datatype.

Datatypes

- Solution: take the node's value as a literal with a certain datatype.
  - e.g.,, RDF writes the node (2018-03-22 12:00) as a literal string with a datatype

    `"2018-03-22T12:00:00"^^xsd:dateTime`

    In RDF, such a node is **not** allowed to have an outgoing edge.

Datatypes

- Solution: take the node's value as a literal with a certain datatype.
  - e.g.,, RDF writes the node (2018-03-22 12:00) as a literal string with a datatype

    `"2018-03-22T12:00:00"^^xsd:dateTime`

    In RDF, such a node is **not** allowed to have an outgoing edge.
  - `xsd:dateTime` is an IRI denoting the datatype. Other IRI datatypes include `xsd:string`, `xsd:decimal`, etc.

Datatypes

- Solution: take the node's value as a literal with a certain datatype.
  - e.g.,, RDF writes the node (2018-03-22 12:00) as a literal string with a datatype

    `"2018-03-22T12:00:00"^^xsd:dateTime`

    In RDF, such a node is **not** allowed to have an outgoing edge.
  - `xsd:dateTime` is an IRI denoting the datatype. Other IRI datatypes include `xsd:string`, `xsd:decimal`, etc.
- Similar notions are also used in property graphs (though maybe not using IRIs).

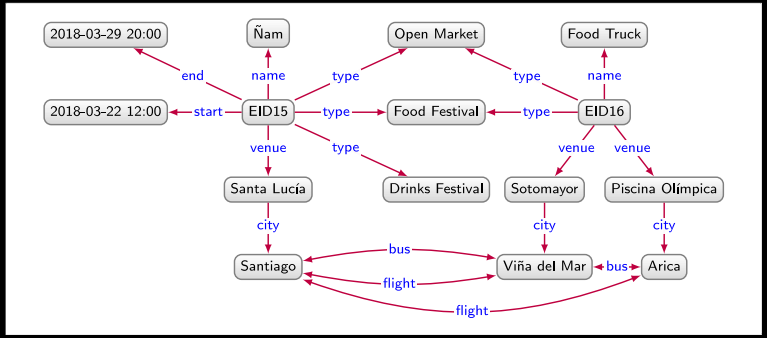
Outline

RDF

- Resource Description Framework (RDF): W3C-standardized DELG data model.
  - RDF 1.0 in 2004
  - RDF 1.1 in 2014 (see `https://www.w3.org/TR/rdf11-primer/`)
- An RDF triple is a statement of the form (s, p, o) where
  - s, called the subject of the triple, is either a IRI or a blank node;
  - p, called the predicate of the triple, is an IRI; and
  - o, called the object of the triple, is an IRI or a blank node or a literal.

  The literal can be assigned a datatype explicitly. Otherwise, the literal is typed `xsd:string` by default if not given explicitly.
- An RDF graph is a set of RDF triples.

Graph example



To model the above graph in RDF, we need to decide which nodes are literal (and their type) and which should be given an IRI.

Which nodes are literal?

- Nodes containing “basic/primitive” data values should be modeled as literals, e.g., strings, numbers, booleans, etc.
- In the previous examples, 4 nodes are literals:
  - two dates (given `xsd:dateTime` as type):

```
"2018-03-29T20:00:00"^^xsd:dateTime
"2018-03-22T12:00:00"^^xsd:dateTime
```

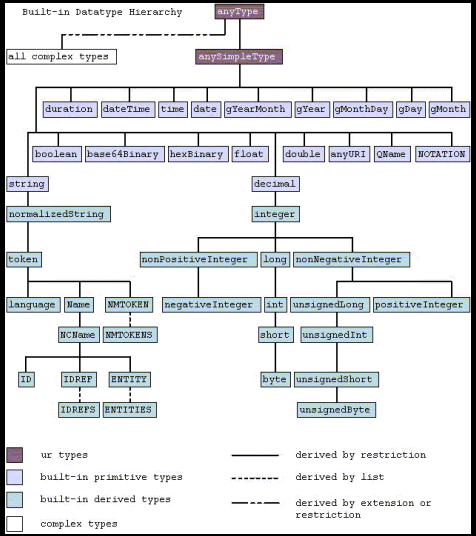
  - two names (can be given type `xsd:string`):

```
"Food Truck"
"Food Truck"^^xsd:string
"Ñam"
"Ñam"^^xsd:string
```

How to give an IRI

- Pick one or more namespaces for your resources.
- Common practice is to set up different IRI patterns for relations, types, and the rest of the entities.
- For some nodes/relations/datatypes, if possible, it's better to reuse standard vocabulary terms. Some vocabulary terms have a standardized/community-agreed semantic that may be useful.
  - `http://www.w3.org/1999/02/22-rdf-syntax-ns#type` or usually abbreviated `rdf:type` for the 'type' relation.
  - Many datatypes are defined as standard in the XML schema namespace, e.g., `http://www.w3.org/2001/XMLSchema#decimal` abbreviated `xsd:decimal` `http://www.w3.org/2001/XMLSchema#string` abbreviated `xsd:string`
  - See `http://prefix.cc` to obtain some well-known namespace IRIs.

XML schema type hierarchy

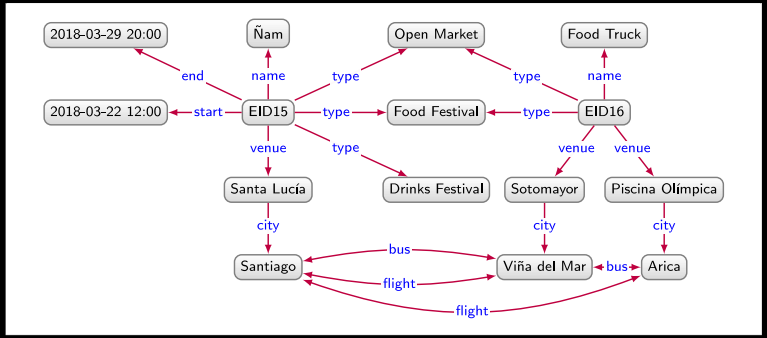


How to give an IRI (2)

For our example,

- we use the following IRI namespaces:
  - For instance nodes: `http://example.org/data/` abbreviated with `ex:`
  - For type nodes and relations (other than the 'type' relation): `http://example.org/vocab#` abbreviated with `exv:`
- We simply take the node and property IDs in the graph to complete their identifier; white spaces are omitted.

RDF graph example



So for the graph above, we can write an RDF graph using N-triples, Turtle, RDF/XML, and JSON-LD in the following.

RDF graph example in N-triples

N-triples format:

- List all triples one by one in arbitrary order.
- Each triple ends with a period.
- All IRIs (including datatype IRIs) are written in full.
- Unicode characters are by default written in an escape sequence form.
- Typical file extension: .nt

RDF graph example in N-triples (cont.)

```
<http://example.org/data/Vi\u00F1adelMar> <http://example.org/vocab#bus>
    <http://example.org/data/Arica> .
<http://example.org/data/EID15> <http://example.org/vocab#venue>
    <http://example.org/data/SantaLuc\u00EDa> .
<http://example.org/data/SantaLuc\u00EDa> <http://example.org/vocab#city>
    <http://example.org/data/Santiago> .
<http://example.org/data/Arica> <http://example.org/vocab#bus>
    <http://example.org/data/Vi\u00F1adelMar> .
<http://example.org/data/Sotomayor> <http://example.org/vocab#city>
    <http://example.org/data/Vi\u00F1adelMar> .
<http://example.org/data/EID16> <http://example.org/vocab#venue>
    <http://example.org/data/Sotomayor> .
<http://example.org/data/EID16> <http://example.org/vocab#name>
    "Food Truck"^^<http://www.w3.org/2001/XMLSchema#string> .
<http://example.org/data/PiscinaOl\u00EDmpica>
    <http://example.org/vocab#city> <http://example.org/data/Arica> .
<http://example.org/data/Santiago> <http://example.org/vocab#flight>
    <http://example.org/data/Vi\u00F1adelMar> .
```

RDF graph example in N-triples (cont.)

```
<http://example.org/data/EID16> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://example.org/vocab#FoodFestival> .
<http://example.org/data/EID15> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://example.org/vocab#FoodFestival> .
<http://example.org/data/EID15> <http://example.org/vocab#name>
    "\u00D1am" .
<http://example.org/data/EID16> <http://example.org/vocab#venue>
    <http://example.org/data/PiscinaOl\u00EDmpica> .
<http://example.org/data/Vi\u00F1adelMar> <http://example.org/vocab#bus>
    <http://example.org/data/Santiago> .
<http://example.org/data/Arica> <http://example.org/vocab#flight>
    <http://example.org/data/Santiago> .
<http://example.org/data/EID15> <http://example.org/vocab#end>
    "2018-03-29T20:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
<http://example.org/data/Santiago> <http://example.org/vocab#bus>
    <http://example.org/data/Vi\u00F1adelMar> .
<http://example.org/data/EID15> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://example.org/vocab#DrinksFestival> .
```

RDF graph example in N-triples (cont.)

```
<http://example.org/data/EID16> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://example.org/vocab#OpenMarket> .
<http://example.org/data/Santiago> <http://example.org/vocab#flight>
    <http://example.org/data/Arica> .
<http://example.org/data/EID15> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
    <http://example.org/vocab#OpenMarket> .
<http://example.org/data/Vi\u00F1adelMar> <http://example.org/vocab#flight>
    <http://example.org/data/Santiago> .
<http://example.org/data/EID15> <http://example.org/vocab#start>
    "2018-03-22T12:00:00"^^<http://www.w3.org/2001/XMLSchema#dateTime> .
```

RDF graph example in Turtle

Turtle format:

- Triples (and triple groups) are ended by a period.
- Triples sharing the same subject can be grouped using semicolons.
- Triples sharing the same subject and predicate can be grouped using commas.
- Namespace prefixes can be used; defined using special syntax.
- Unicode characters need not be esacped.
- Typical file extension: .ttl
- Format conversion can be done using online tools, e.g., `https://issemantic.net/rdf-converter` (also has RDF visualizer) or `https://www.easyrdf.org/converter`. Libraries such as Jena also supports it

RDF graph example in Turtle (cont.)

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix ex: <http://example.org/data/> .
@prefix exv: <http://example.org/vocab#> .

ex:EID15 rdf:type exv:OpenMarket, exv:FoodFestival, exv:DrinksFestival ;
    exv:name "Ñam" ;
    exv:start "2018-03-22T12:00:00"^^xsd:dateTime ;
    exv:end "2018-03-29T20:00:00"^^xsd:dateTime ;
    exv:venue ex:SantaLucía .

ex:SantaLucía exv:city ex:Santiago .

ex:EID16 rdf:type exv:OpenMarket, exv:FoodFestival ;
    exv:name "Food Truck"^^xsd:string ;
    exv:venue ex:Sotomayor, ex:PiscinaOlímpica .
```

RDF graph example in Turtle (cont.)

```
ex:Sotomayor exv:city ex:ViñadelMar .
ex:PiscinaOlímpica exv:city ex:Arica .

ex:Santiago exv:bus ex:ViñadelMar ;
     exv:flight ex:ViñadelMar, ex:Arica .

ex:ViñadelMar exv:bus ex:Santiago, ex:Arica ;
     exv:flight ex:Santiago .

ex:Arica exv:bus ex:ViñadelMar ;
     exv:flight ex:Santiago .
```

RDF graph example in RDF/XML

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:exv="http://example.org/vocab#"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <rdf:Description rdf:about="http://example.org/data/EID15">
    <rdf:type rdf:resource="http://example.org/vocab#OpenMarket"/>
    <exv:start rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
      2018-03-22T12:00:00</exv:start>
    <exv:name>Ñam</exv:name>
    <exv:venue rdf:resource="http://example.org/data/SantaLucía"/>
    <exv:end rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">
      2018-03-29T20:00:00</exv:end>
    <rdf:type rdf:resource="http://example.org/vocab#DrinksFestival"/>
    <rdf:type rdf:resource="http://example.org/vocab#FoodFestival"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/ViñadelMar">
```

RDF graph example in RDF/XML (cont.)

```
    <exv:bus rdf:resource="http://example.org/data/Santiago"/>
    <exv:flight rdf:resource="http://example.org/data/Santiago"/>
    <exv:bus rdf:resource="http://example.org/data/Arica"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/Arica">
    <exv:bus rdf:resource="http://example.org/data/ViñadelMar"/>
    <exv:flight rdf:resource="http://example.org/data/Santiago"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/EID16">
    <rdf:type rdf:resource="http://example.org/vocab#FoodFestival"/>
    <exv:name rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
      Food Truck</exv:name>
    <exv:venue rdf:resource="http://example.org/data/Sotomayor"/>
    <exv:venue rdf:resource="http://example.org/data/PiscinaOlímpica"/>
    <rdf:type rdf:resource="http://example.org/vocab#OpenMarket"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/Santiago">
```

RDF graph example in RDF/XML (cont.)

```
    <exv:bus rdf:resource="http://example.org/data/ViñadelMar"/>
    <exv:flight rdf:resource="http://example.org/data/Arica"/>
    <exv:flight rdf:resource="http://example.org/data/ViñadelMar"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/PiscinaOlímpica">
    <exv:city rdf:resource="http://example.org/data/Arica"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/Sotomayor">
    <exv:city rdf:resource="http://example.org/data/ViñadelMar"/>
  </rdf:Description>
  <rdf:Description rdf:about="http://example.org/data/SantaLucía">
    <exv:city rdf:resource="http://example.org/data/Santiago"/>
  </rdf:Description>
</rdf:RDF>
```

RDF graph example in JSON-LD

JSON-LD format:

- Abstract structure is similar to Turtle, but written using combination of lists and dictionaries.
- No IRI abbreviation.
- Nodes with IRI are indicated by `"@id"` key.
- Type relation is indicated by a special key: `"@type"`.
- Literal values is indicated by a special key: `"@value"`.

RDF graph example in JSON-LD (cont.)

```
[
  {
    "@id": "http://example.org/data/Santiago",
    "http://example.org/vocab#bus": [
      {
        "@id": "http://example.org/data/ViñadelMar"
      }
    ],
    "http://example.org/vocab#flight": [
      {
        "@id": "http://example.org/data/Arica"
      },
      {
        "@id": "http://example.org/data/ViñadelMar"
      }
    ]
  },
```

RDF graph example in JSON-LD (cont.)

```
{
  "@id": "http://example.org/data/SantaLucía",
  "http://example.org/vocab#city": [
    {
      "@id": "http://example.org/data/Santiago"
    }
  ]
},
{
  "@id": "http://example.org/data/EID15",
  "@type": [
    "http://example.org/vocab#OpenMarket",
    "http://example.org/vocab#DrinksFestival",
    "http://example.org/vocab#FoodFestival"
  ],
  "http://example.org/vocab#end": [
    {
```

RDF graph example in JSON-LD (cont.)

```
      "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
      "@value": "2018-03-29T20:00:00"
    }
  ],
  "http://example.org/vocab#name": [
    {
      "@value": "Ñam"
    }
  ],
  "http://example.org/vocab#start": [
    {
      "@type": "http://www.w3.org/2001/XMLSchema#dateTime",
      "@value": "2018-03-22T12:00:00"
    }
  ],
  "http://example.org/vocab#venue": [
    {
```

RDF graph example in JSON-LD (cont.)

```
        "@id": "http://example.org/data/SantaLucía"
      }
    ]
  },
  {
    "@id": "http://example.org/data/EID16",
    "@type": [
      "http://example.org/vocab#FoodFestival",
      "http://example.org/vocab#OpenMarket"
    ],
    "http://example.org/vocab#name": [
      {
        "@value": "Food Truck"
      }
    ],
    "http://example.org/vocab#venue": [
      {
```

RDF graph example in JSON-LD (cont.)

```
      "@id": "http://example.org/data/Sotomayor"
    },
    {
      "@id": "http://example.org/data/PiscinaOlímpica"
    }
  ]
},
{
  "@id": "http://example.org/data/PiscinaOlímpica",
  "http://example.org/vocab#city": [
    {
      "@id": "http://example.org/data/Arica"
    }
  ]
},
{
  "@id": "http://example.org/data/Sotomayor",
```

RDF graph example in JSON-LD (cont.)

```
    "http://example.org/vocab#city": [
      {
        "@id": "http://example.org/data/ViñadelMar"
      }
    ]
  },
  {
    "@id": "http://example.org/data/Arica",
    "http://example.org/vocab#bus": [
      {
        "@id": "http://example.org/data/ViñadelMar"
      }
    ],
    "http://example.org/vocab#flight": [
      {
        "@id": "http://example.org/data/Santiago"
      }
```

RDF graph example in JSON-LD (cont.)

```
    ]
  },
  {
    "@id": "http://example.org/data/ViñadelMar",
    "http://example.org/vocab#bus": [
      {
        "@id": "http://example.org/data/Santiago"
      },
      {
        "@id": "http://example.org/data/Arica"
      }
    ],
    "http://example.org/vocab#flight": [
      {
        "@id": "http://example.org/data/Santiago"
      }
    ]
```

RDF graph example in JSON-LD (cont.)

```
  }
]
```

What about **blank node**?
(allowed as the subject and the object of an RDF triple)

Outline

Blank nodes = anonymous nodes

- When modeling incomplete information or when “reifying” a relation into a node, we sometimes need to state that there must exist a node in the graph without being able or required to given an explicit identity to the node.

Blank nodes = anonymous nodes

- When modeling incomplete information or when “reifying” a relation into a node, we sometimes need to state that there must exist a node in the graph without being able or required to given an explicit identity to the node.

- In such a case, it may be useful to use the so-called blank nodes, which is a node in the graph to which an IRI is not assigned.

Blank nodes: Example

- Consider two co-located events `chile:EID42` and `chile:EID43` whose venue has yet to be announced.

Blank nodes: Example

- Consider two co-located events `chile:EID42` and `chile:EID43` whose venue has yet to be announced.
- Option 1: drop the ‘venue’ edge, but

Blank nodes: Example

- Consider two co-located events `chile:EID42` and `chile:EID43` whose venue has yet to be announced.
- Option 1: drop the ‘venue’ edge, but we can lose the information that these events have a venue and that both events have the same venue.

Blank nodes: Example

- Consider two co-located events `chile:EID42` and `chile:EID43` whose venue has yet to be announced.
- Option 1: drop the ‘venue’ edge, but we can lose the information that these events have a venue and that both events have the same venue.
- Option 2: create a fresh IRI representing the venue, but

Blank nodes: Example

- Consider two co-located events `chile:EID42` and `chile:EID43` whose venue has yet to be announced.
- Option 1: drop the ‘venue’ edge, but we can lose the information that these events have a venue and that both events have the same venue.
- Option 2: create a fresh IRI representing the venue, but this becomes semantically indistinguishable from there being a known venue (though right now it should been unknown).

Blank nodes: Example

- Consider two co-located events `chile:EID42` and `chile:EID43` whose venue has yet to be announced.
- Option 1: drop the ‘venue’ edge, but we can lose the information that these events have a venue and that both events have the same venue.
- Option 2: create a fresh IRI representing the venue, but this becomes semantically indistinguishable from there being a known venue (though right now it should been unknown).
- Option 3: use a blank node

Blank nodes: Example (contd.)

Using blank nodes, we can model the two co-located events as:

- Edges capture the meaning that there exists a common venue for `chile:EID42` and `chile:EID43` without identifying it.
- Can be written in Turtle (assuming namespace prefixes defined) as:

```
chile:EID42 chile:venue _:b1 .
chile:EID43 chile:venue _:b1 .
```

where `_:b1` is the blank node identifier.

Identity of blank nodes

- Does blank nodes have an identifier?

Identity of blank nodes

- Does blank nodes have an identifier? Yes. (See the N-triple version of the previous example).

Identity of blank nodes

- Does blank nodes have an identifier? Yes. (See the N-triple version of the previous example).
- How does the identifier differ from the standard IRIs?

Identity of blank nodes

- Does blank nodes have an identifier? Yes. (See the N-triple version of the previous example).
- How does the identifier differ from the standard IRIs?
  - They differ in scope: blank node identifiers are unique within a single RDF document.

Identity of blank nodes

- Does blank nodes have an identifier? Yes. (See the N-triple version of the previous example).
- How does the identifier differ from the standard IRIs?
  - They differ in scope: blank node identifiers are unique within a single RDF document.
  - Blank node identifiers cannot be referred to from outside the document
  - Re-writing/reloading the graph (by machines) may change blank node identifiers.

Many-value relation representation with blank nodes

Model the following in RDF: “Chutney has 1 lb. green Mango and 1 tsp. Cayenne pepper as ingredients.”

Many-value relation representation with blank nodes

Attempt 1:

```
@prefix ex: <http://example.org/> .

ex:Chutney ex:hasIngredient "1 lb. green mango",
                            "1 tsp. Cayenne pepper"
```

Many-value relation representation with blank nodes

Attempt 1:

```
@prefix ex: <http://example.org/> .

ex:Chutney ex:hasIngredient "1 lb. green mango",
                            "1 tsp. Cayenne pepper"
```

Can we query all recipes containing green mango?

Many-value relation representation with blank nodes

Attempt 2:

```
@prefix ex: <http://example.org/> .
ex:Chutney ex:ingredient ex:greenMango;
           ex:amount "1 lb." ;
           ex:ingredient ex:CayennePepper;
           ex:amount "1tsp." .
```

Many-value relation representation with blank nodes

Attempt 2:

```
@prefix ex: <http://example.org/> .
ex:Chutney ex:ingredient ex:greenMango;
           ex:amount "1 lb." ;
           ex:ingredient ex:CayennePepper;
           ex:amount "1tsp." .
```

Can we unambiguously obtain the amount of Cayenne pepper used by Chutney?

Many-value relation representation with blank nodes

Attempt 3:

```
@prefix ex: <http://example.org/> .

ex:Chutney ex:ingredient ex:greenMango,
                         ex:CayennePepper .
ex:greenMango ex:amount "1 lb." ;
ex:CayennePepper ex:amount "1 tsp."
```

Many-value relation representation with blank nodes

Attempt 3:

```
@prefix ex: <http://example.org/> .

ex:Chutney ex:ingredient ex:greenMango,
                         ex:CayennePepper .
ex:greenMango ex:amount "1 lb." ;
ex:CayennePepper ex:amount "1 tsp."
```

Can we unambiguously obtain the amount of Cayenne pepper used by Chutney?

Many-value relation representation with blank nodes

Attempt 4 without blank nodes:

```
@prefix ex: <http://example.org/> .

ex:Chutney ex:hasIngredient ex:ingredient1, ex:ingredient2 .
ex:ingredient1 ex:ingredient ex:greenMango;
               ex:amount "1 lb." .
ex:ingredient2 ex:ingredient ex:CayennePaper ;
               ex:amount "1 tsp." .
```

Many-value relation representation with blank nodes

Attempt 4 with blank nodes:

```
@prefix ex: <http://example.org/> .

ex:Chutney ex:hasIngredient _:b1, _:b2 .
_:b1 ex:ingredient ex:greenMango;
     ex:amount "1 lb." .
_:b2 ex:ingredient ex:CayennePaper ;
     ex:amount "1 tsp." .
```

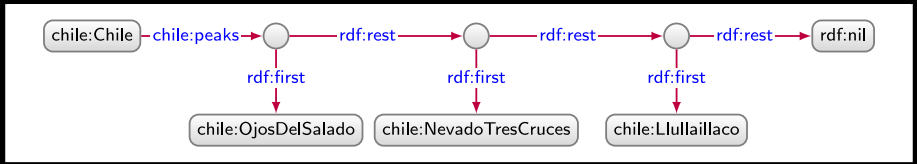
Many-value relation representation with blank nodes

Attempt 4 with blank nodes using square bracket syntax:

```
@prefix ex: <http://example.org/> .

ex:Chutney ex:hasIngredient
             [ ex:ingredient ex:greenMango ;
               ex:amount "1 lb." ],
             [ ex:ingredient ex:CayennePaper ;
               ex:amount "1 tsp." ].
```

RDF list representation with blank nodes



can be compactly represented in Turtle using parentheses notation:

```
chile:Chile chile:peaks
       (chile:OjosDelSalado chile:NevadoTresCruces chile:Llullailaco) .
```

Further remarks on blank nodes

- Existential nodes/blank nodes can be convenient, but also complicate operations on graphs.
- In some cases, we may want to replace blank nodes with nodes with canonical IRIs.
- Some researchers also suggest to minimize the use of blank nodes in the graph.

Outline

Lexicalization

- Global identifiers (IRIs) may sometimes have a human-interpretable form, e.g., `chile:Santiago`.
- But, the identifier strings themselves do not carry any formal semantic significance
  - It's perfectly acceptable to simply use random string as identifier as long as its use is unambiguous.
- Real world examples: in Wikidata, the identifier for Eswatini is `wd:Q1050`
  - No need to choose between languages for creating IRIS, e.g., `wd:Eswatini` (English), `wd:eSwatini` (Swahili), or `wd:Esuatini` (Spanish).

Lexicalization (cont.)

- Since identifiers can be arbitrary, it is common to add edges that provide human-interpretable label for nodes (possibly with language tags), e.g.,

```
wd:Q1050 rdfs:label "Swatini"
wd:Q1050 rdfs:label "Swatini"@en
wd:Q1050 rdfs:label "eSwatini"@sw
wd:Q1050 rdfs:label "Esuaatini"@es
```

Lexicalization (cont.)

- Other linguistic information could also be added, such as aliases (using `skos:altLabel` property) or comments (using `rdfs:comment` property)

- Benefits of linguistic information through such metadata:
  - can help user identify which real-world entity a node in KG actually references;
  - enables cross-referencing with text corpora to find documents that provide details about an entity;
  - can help user interfaces in displaying the data.