Maintaining Temporal Perspective

Ian Emmons and Doug Reid
{ iemmons, dreid }@bbn.com
STIDS 2010
http://asio.bbn.com/2010/10/stids/time.pdf

Raytheon
BBN Technologies

Outline

- Some background
  - The project and its temporal requirement
- The knowledge representation
  - How we store transaction time
- Query expansion
  - How we interpret the metadata

Project Background

- Risk analysis application for assessing risks to one particular high-value resource
- Continually gathers data from five relational databases, stores it as RDF in a triple store
- Data includes events as well as latest current state, both of which are time-sensitive
- Stores analyses as they are performed daily
- Later review of analyses is important

The Temporal Requirement

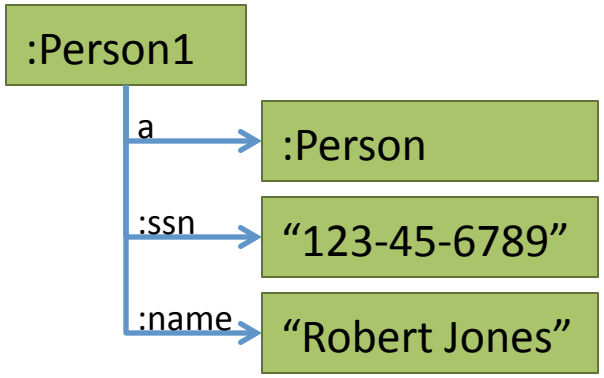
- Must maintain temporal perspective for subsequent review and inspection
  - Older analyses mean little in the context of current data
- Temporal perspective also useful in the analysis task itself
  - The age of what we know sometimes changes how we interpret it

Solution Components

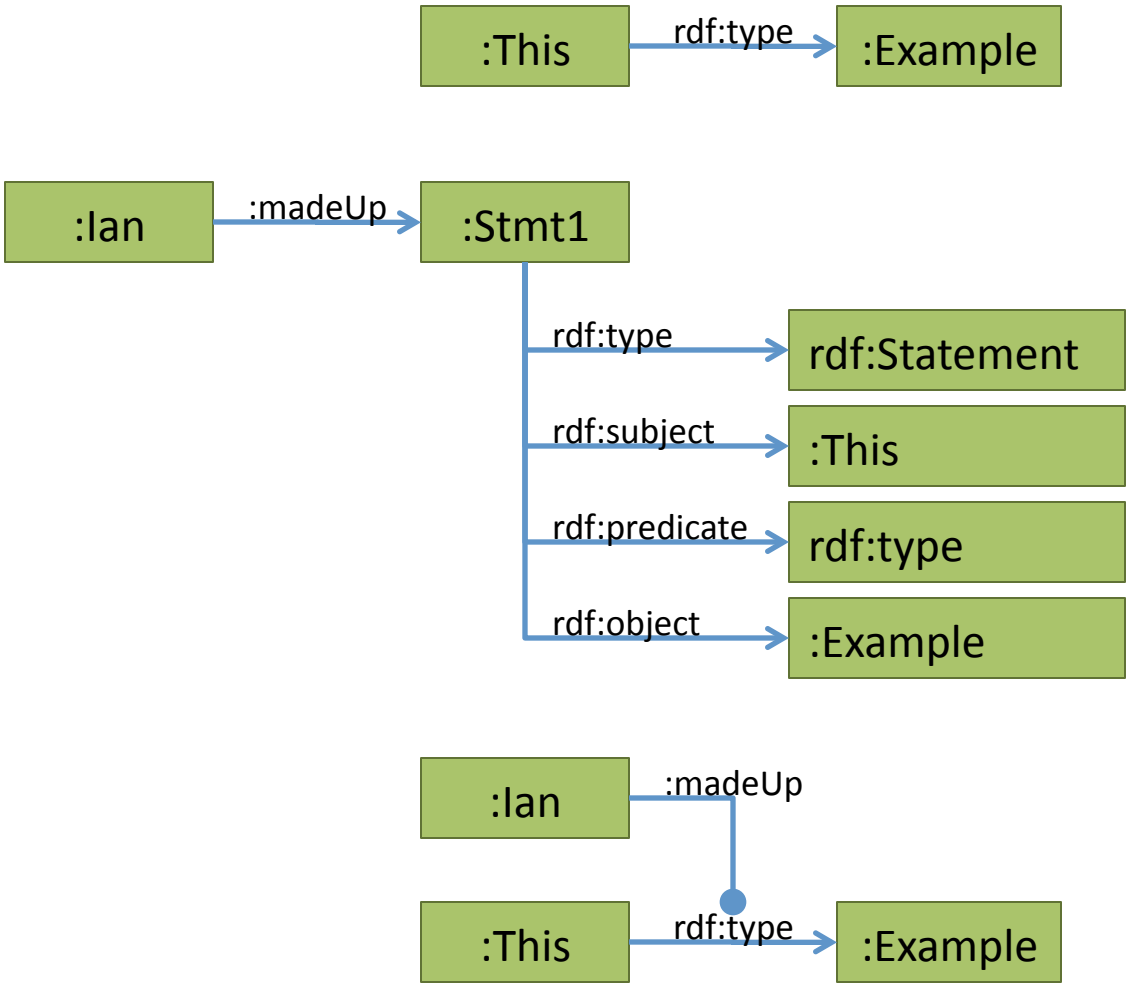
- The RDF knowledge representation
  - Associates facts with transaction times
  - Handles co-reference resolution
- Query expander
  - Transforms a time-agnostic SPARQL query and a point in time into a new, time-sensitive query
  - Transformed query yields the subset of the results of the original query that were valid at the given time

The Knowledge Representation

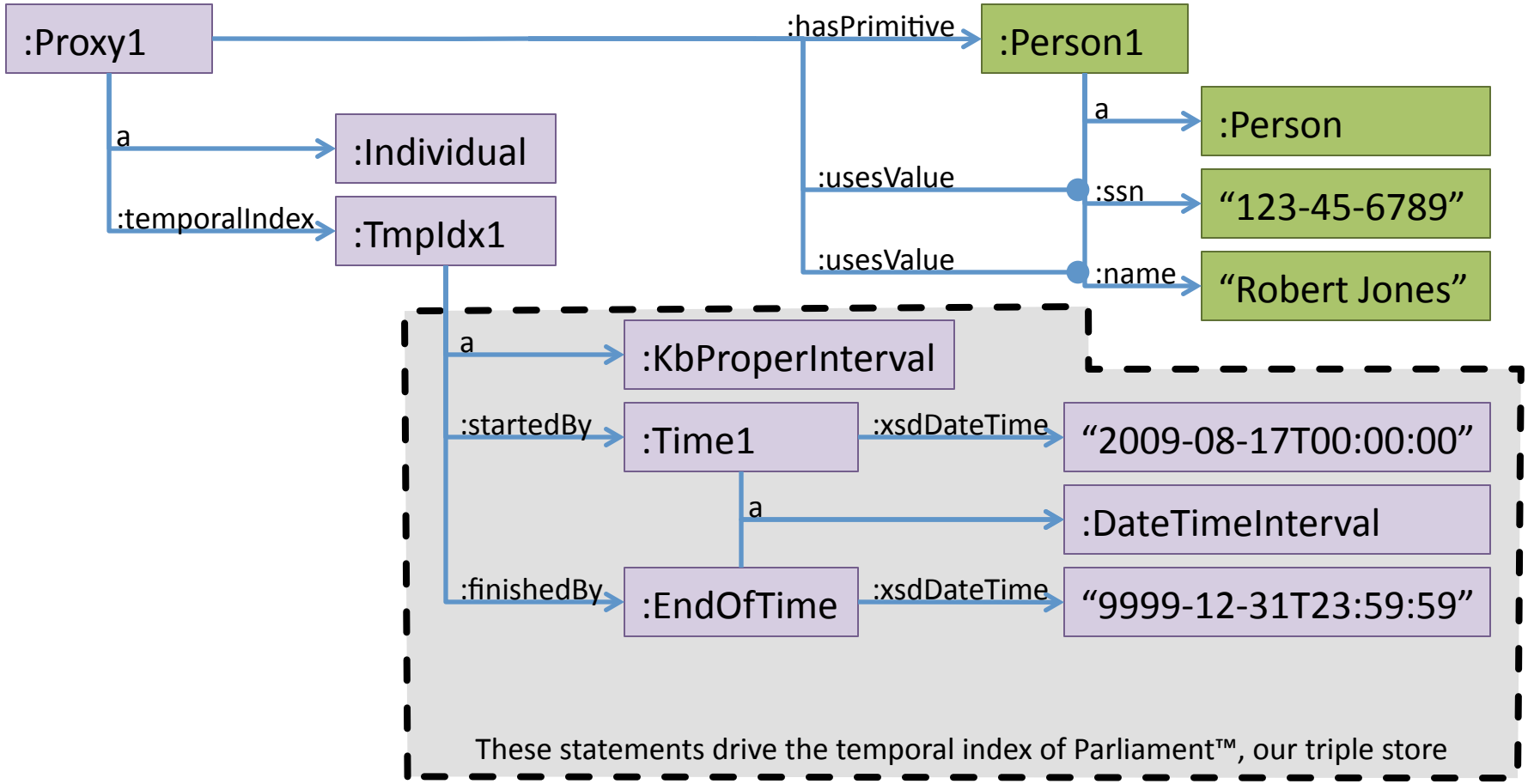
- First import has one person with type, name, and social security number
- Identifiers preceded by colons are URIs with base URI suppressed for brevity
- Items in quotes are literals



Reification

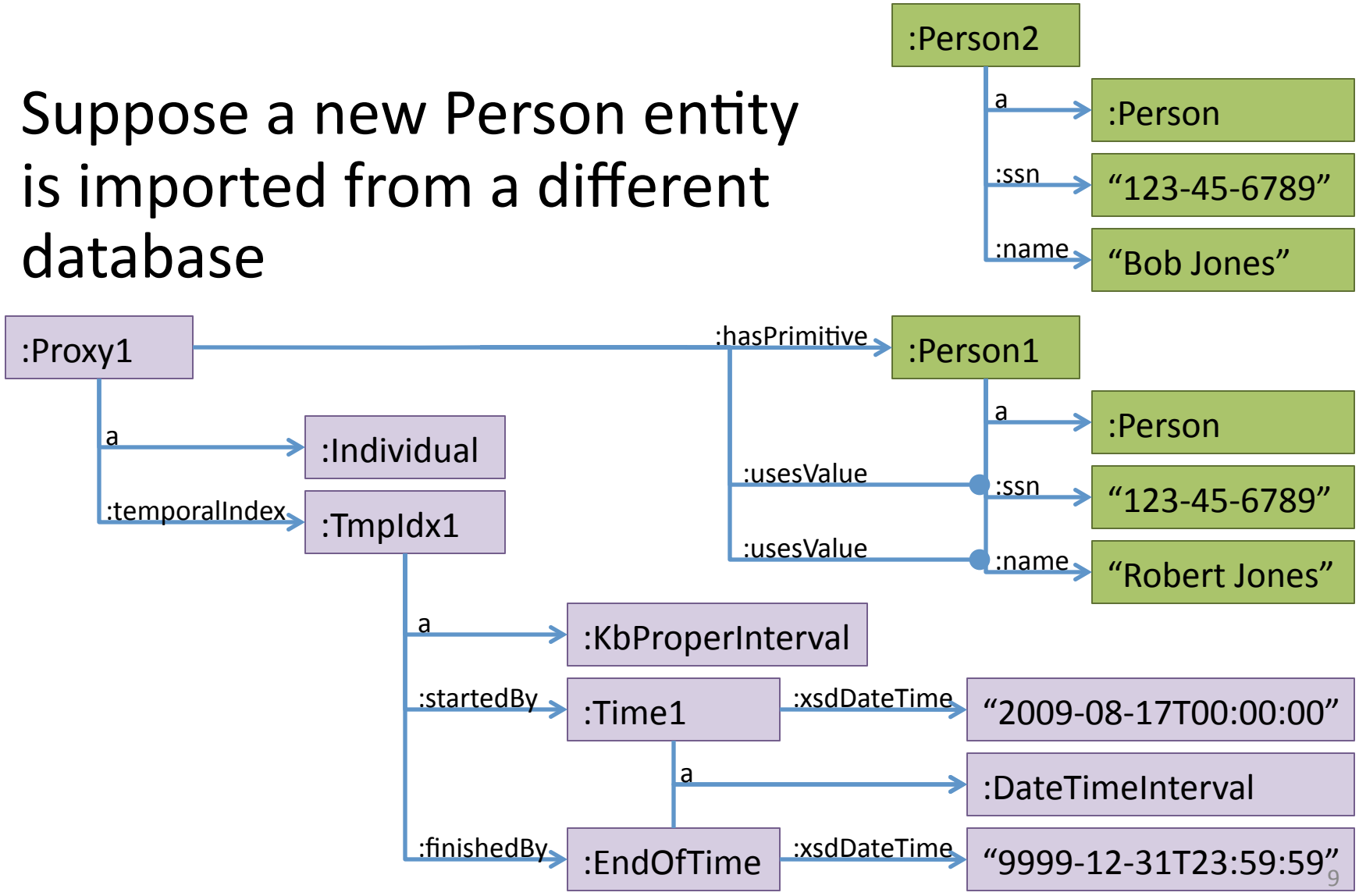


Adding a Proxy



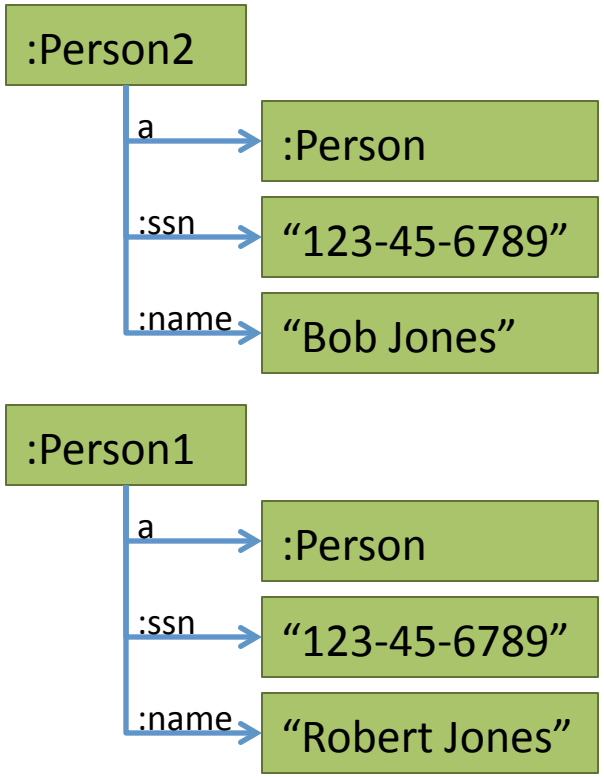
Later, More Data Added

Suppose a new Person entity is imported from a different database

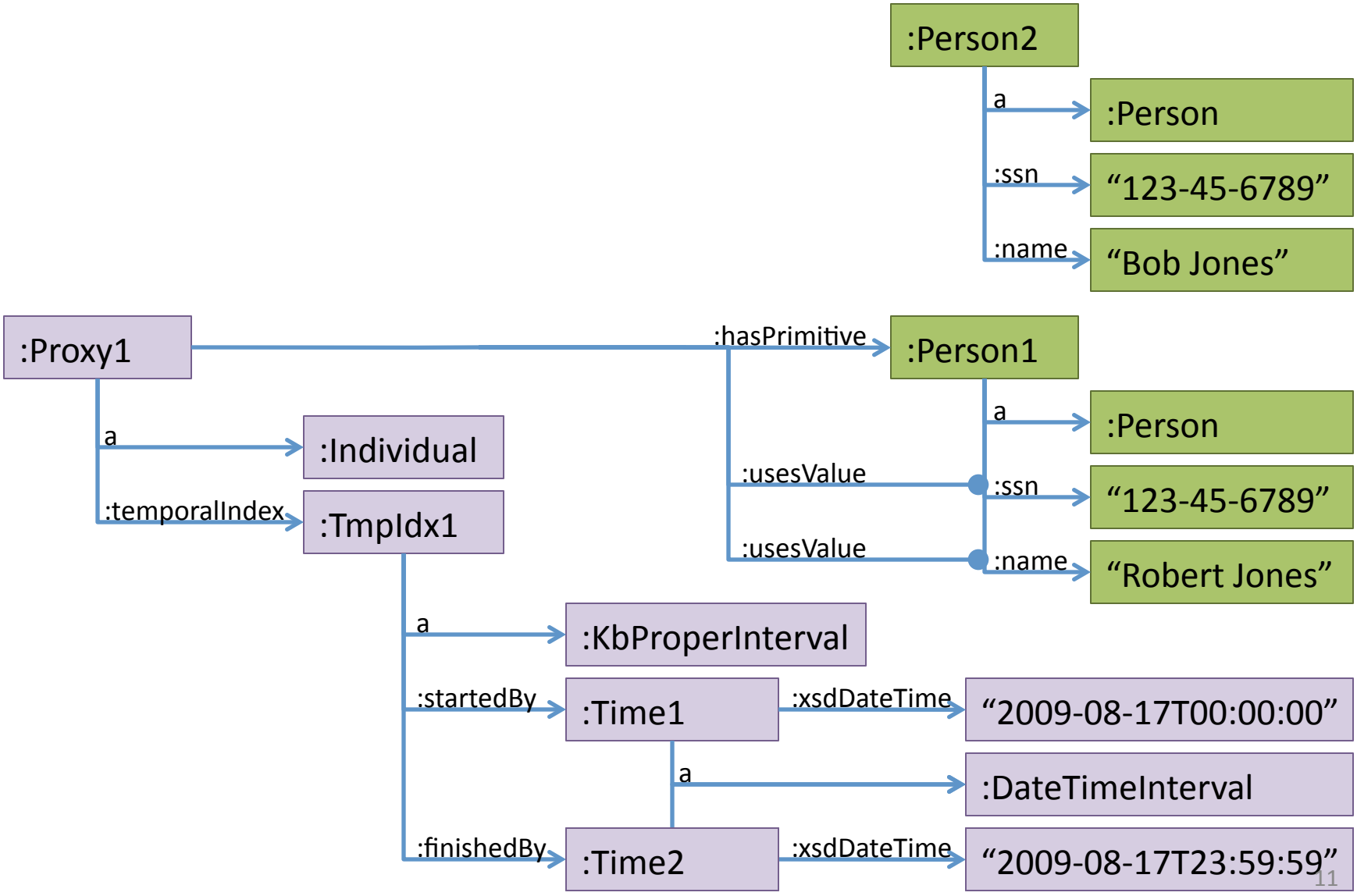


Coreference Resolution & Merging

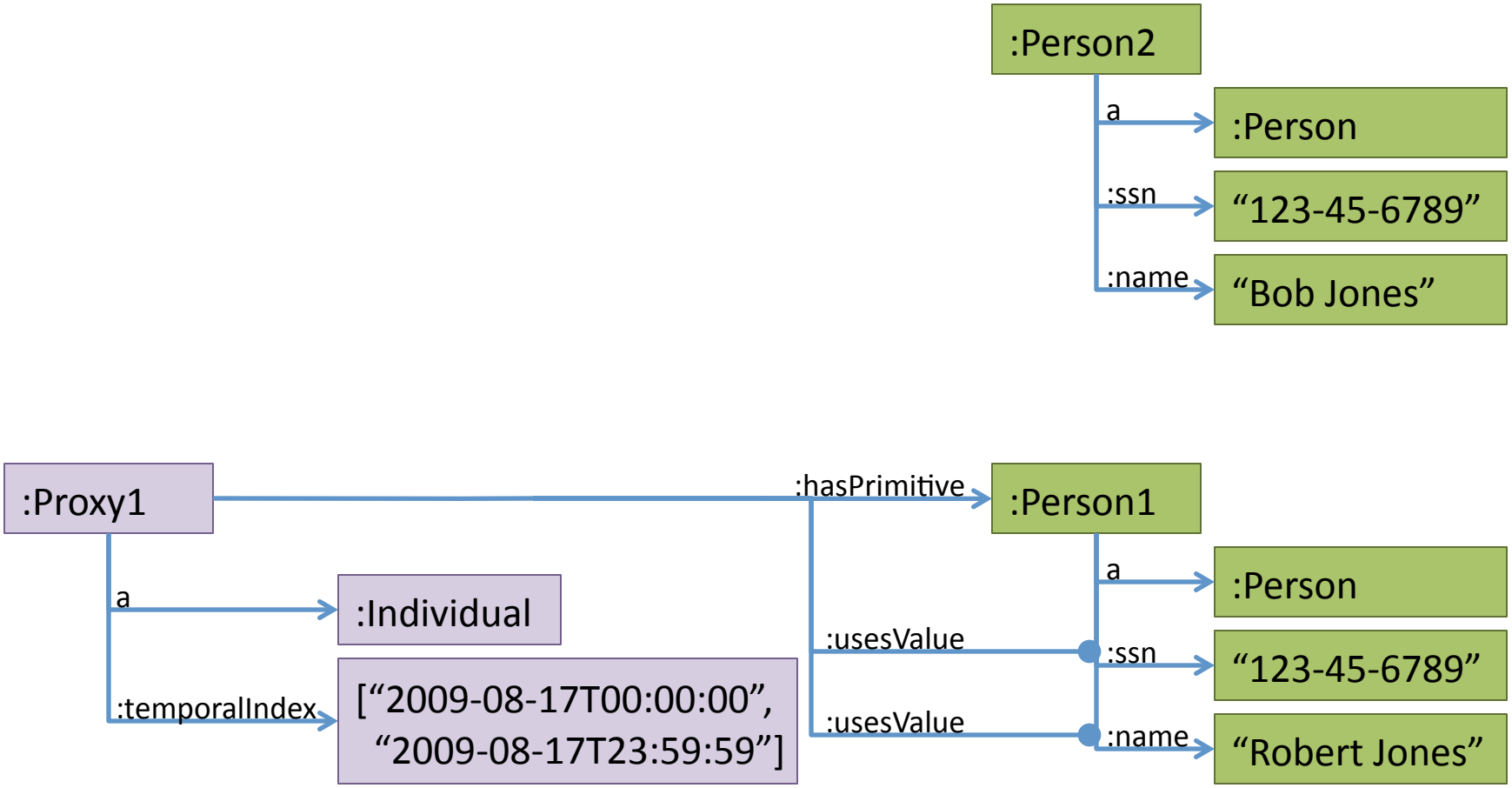
- These two data entities are "obviously" the same real-world entity
- They form a coreference that we want to resolve (merge)
- Common problem with multi-source data
- Merging should be reversible and non-destructive
- Must maintain temporal perspective



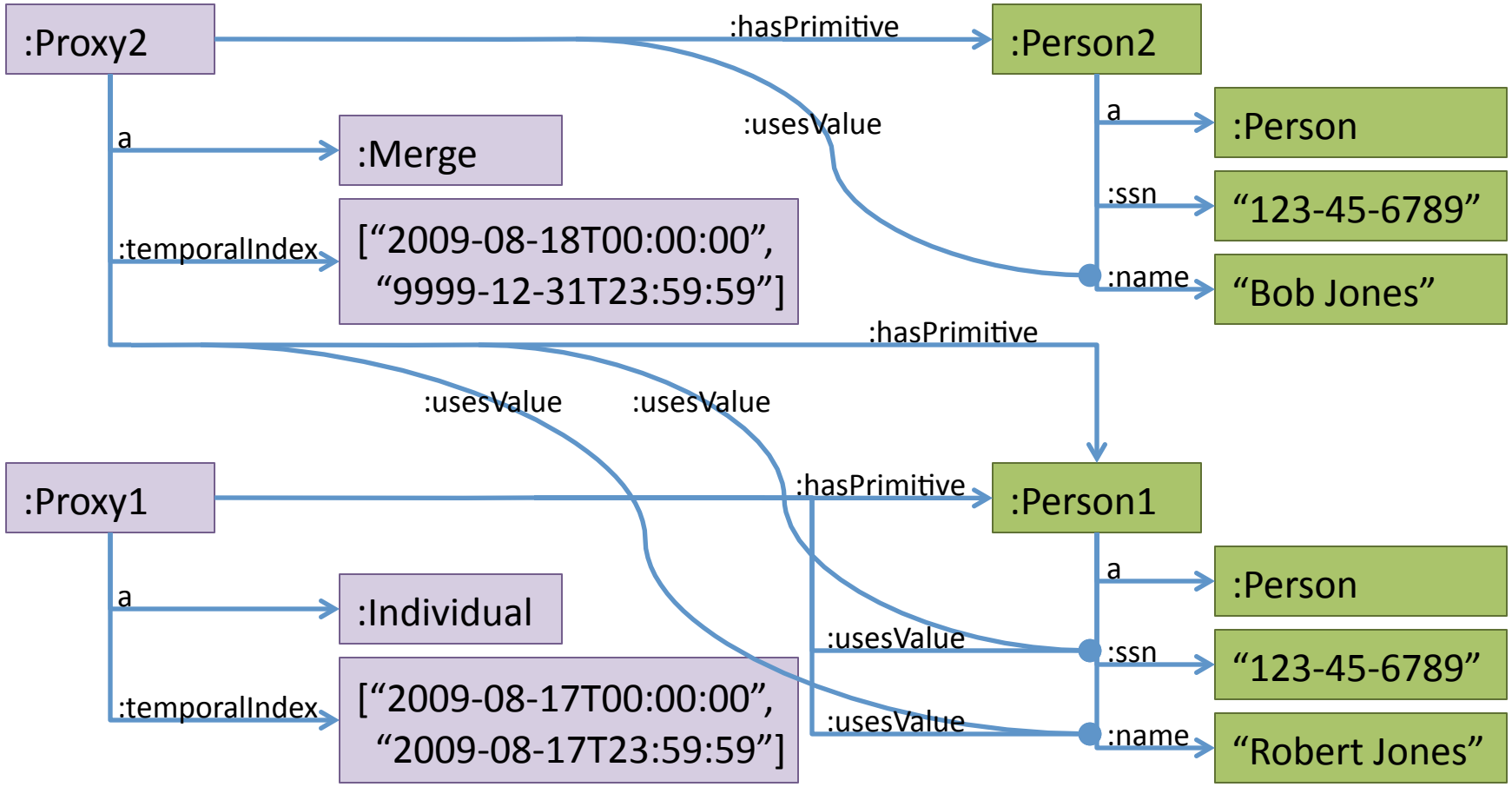
Merge Step 1: Expiring the Proxy



Temporal Index Shorthand



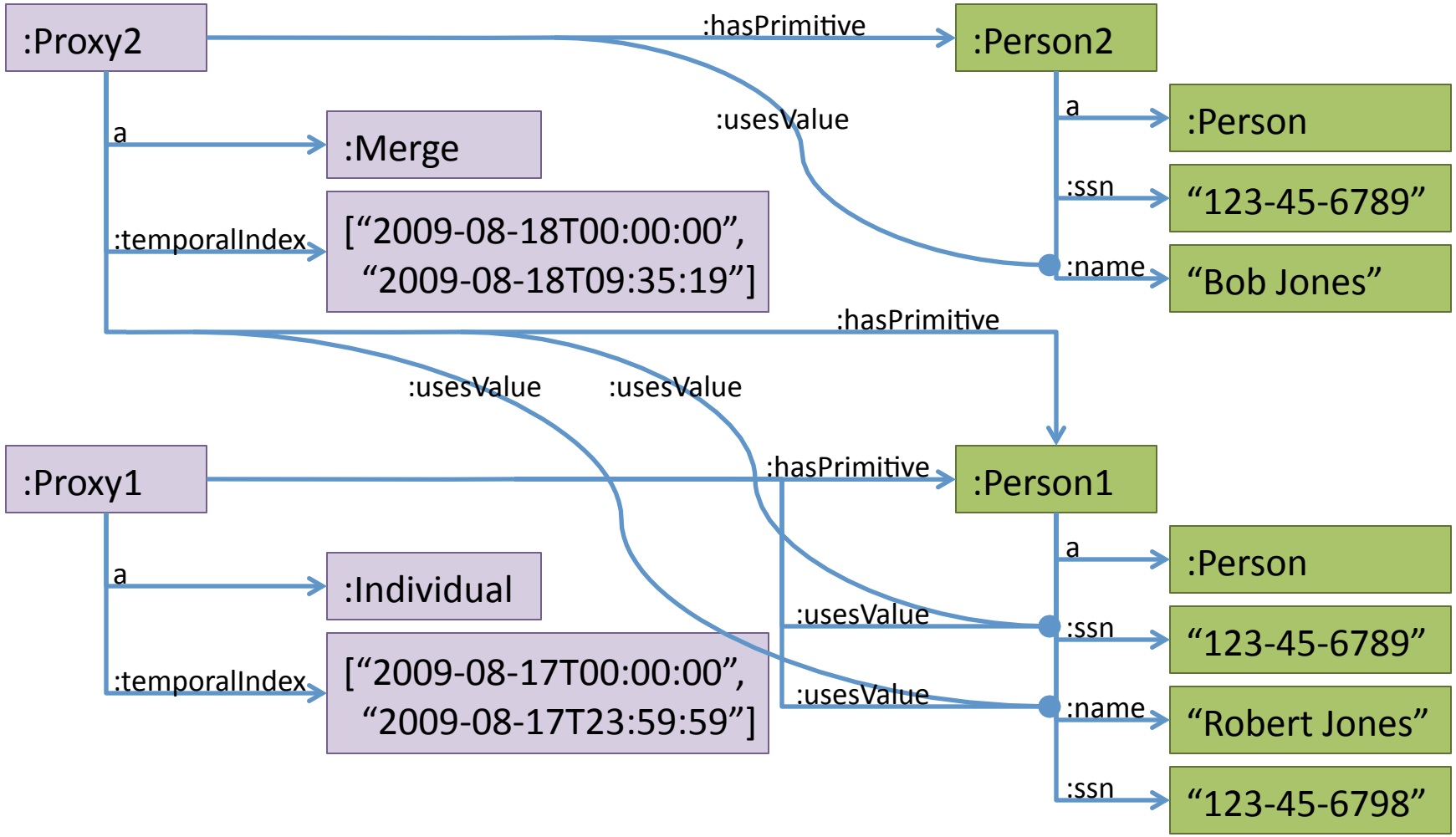
Merge Step 2: Adding a Merge Proxy



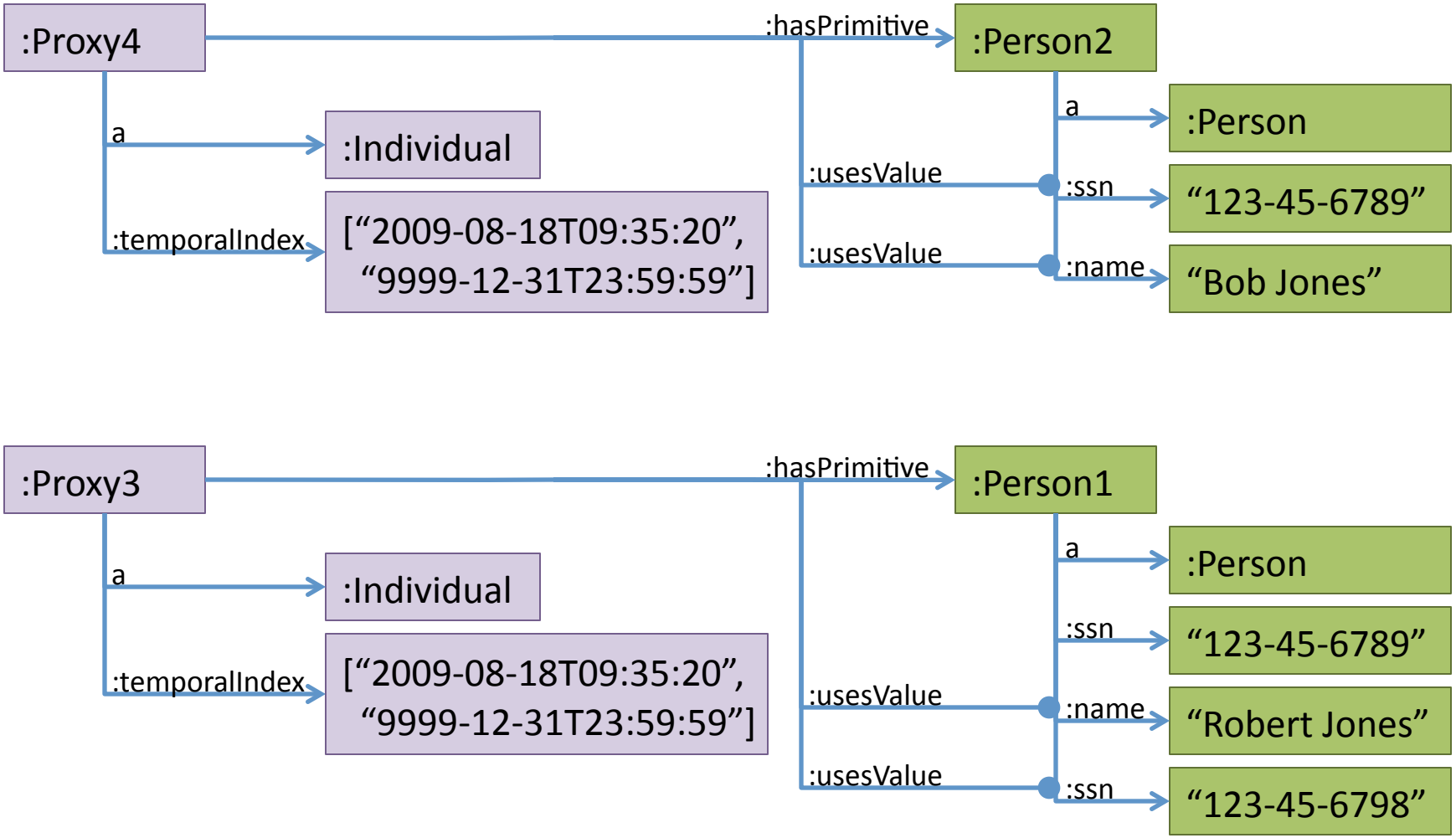
A Third Data Import

- Suppose a third import happens:
  - Re-imports the same “Robert Jones” record
  - But the SSN was changed in the interim
- The same record from the same database will:
  - Result in the same :Person1 entity (since the URI is formed from the primary key)
  - But creates a new :ssn property value

New Data ⇒ Expire a Proxy



Adding New Proxies



Query Expander

- Query expander
  - Transforms a time-agnostic SPARQL query and a point in time into a new, time-sensitive query
  - Transformed query yields the subset of the results of the original query that were valid at the given time
- Caveat: Not a result set rewriter
  - Leaves original query bindings as-is; adds new variables to result set
  - Customer wanted direct access to non-proxied entities in query results

Why Query Expansion?

- Writing temporally-annotated queries is complex and tedious
- Client-side query building should be simple
- Allows for altering the expanded query structure to exploit query optimization routines

Our Example Scenario

Date Time	Actions
2009-08-17 00:00:00	1. **:Person1** added to KB 2. **:Proxy1** created and added to KB for **:Person1**
2009-08-18 00:00:00	1. **:Person2** added to KB 2. **:Person2** discovered to be same person as **:Person1** 3. **:Proxy2** created to represent the Merge of **:Person1** and **:Person2** 4. **:Proxy1** retired
2009-08-18 09:35:20	1. **:Person1** re-imported, adding second SSN value 2. **:Person1** and **:Person2** are un-merged by Analyst 3. **:Proxy3** and :**Proxy4** created to represent the individuals **:Person1** and **:Person2** going forward 4. **:Proxy2** retired

The Resultant KB

```
:Person1 a :Person ;
  :name “Robert Jones” ;
  :ssn “123-45-6789”, “123-45-6798”.

:Proxy1 a :Individual ;
  :hasPrimitive :Person1 ;
  :usesValue
   [ :Person1 :name “Robert Jones” ] ,
   [ :Person1 :ssn “123-45-6789” ] ;
  :temporalIndex
   [ 2009-08-17T00:00:00
     .. 2009-08-17T23:59:59 ] .

:Proxy2 a :Merge ;
  :hasPrimitive :Person1, :Person2 ;
  :usesValue
   [ :Person1 :name “Robert Jones” ] ,
   [ :Person2 :name “Bob Jones” ] ,
   [ :Person1 :ssn “123-45-6789” ] ;
  :temporalIndex
   [ 2009-08-18T00:00:00
     .. 2009-08-18T09:35:19 ] .
```

```
:Person2 a :Person ;
  :name “Bob Jones” ;
  :ssn “123-45-6789”.

:Proxy3 a :Individual ;
  :hasPrimitive :Person1 ;
  :usesValue
   [ :Person1 :name “Robert Jones” ] ,
   [ :Person1 :ssn “123-45-6798” ] ;
  :temporalIndex
   [ 2009-08-18T09:35:20
     .. 9999-12-31T23:59:59 ] .

:Proxy4 a :Individual ;
  :hasPrimitive :Person2 ;
  :usesValue
   [ :Person2 :name “Bob Jones” ] ,
   [ :Person2 :ssn “123-45-6789” ] ;
  :temporalIndex
   [ 2009-08-18T09:35:20
     .. 9999-12-31T23:59:59 ] .
```

Simple Time-Agnostic Query

Query:

```
SELECT DISTINCT ?person ?name
WHERE {
   ?person a :Person .
   ?person :name ?name .
   ?person :ssn “123-45-6789” .
}
```

Results:

?person	?name
:Person1	“Robert Jones”
:Person2	“Bob Jones”

Adding Time Sensitivity…

Query:

```
SELECT DISTINCT ?person ?name
WHERE {
   ?person a :Person .
   ?person :name ?name .
   ?person :ssn “123-45-6789” .
}
```

Example Time Interval of Interest:

```
2009-08-17T12:00:00Z
(Noon, August 17, 2009)
```

Adding Time Sensitivity…

Our Steps:

1. Keep original query structure
   - Optimization hints (observed reification issues)
   - Enhanced readability
2. Add proxy representation to mirror original query structure
   - Use ontological hints
   - Proxy structure added at appropriate query depth
3. Add temporal interval information to each proxy in modified query

Adding Time Sensitivity…

```
SELECT DISTINCT ?person_proxy ?person ?name
WHERE {
   # original query block
   ?person a :Person .
   ?person :name ?name .
   ?person :ssn “123-45-6789” .
   …
}
```

Adding Time Sensitivity…

```
SELECT DISTINCT ?person_proxy ?person ?name
WHERE {
   # original query block …
   # proxy representation block
   ?person_proxy a :Proxy .
   ?person_proxy :hasPrimitive ?person .
   ?person_proxy :usesValue ?statement1 .
   ?statement1 rdf:predicate :name .
   ?statement1 rdf:object ?name .
   ?person_proxy :usesValue ?statement2 .
   ?statement2 rdf:predicate :ssn .
   ?statement2 rdf:object “123-45-6789” .
   …
}
```

Adding Time Sensitivity…

```
SELECT DISTINCT ?person_proxy ?person ?name
WHERE {
   # original query block …
   # proxy representation block …
   # temporal block
   {
      ?person_proxy :temporalIndex ?interval1 .
      ?interval1 a :ProperInterval ;
         :intervalContains ?dtInterval1 .
      ?dtInterval1 a :DateTimeInterval ;
         :xsdDateTime
            “2009-08-17T12:00:00Z”^^xsd:dateTime .
   }
}
```

Results

For 2009-08-17T12:00:00Z

?person_proxy	?person	?name
:Proxy1	:Person1	“Robert Jones”

For 2009-08-18T09:00:00Z

?person_proxy	?person	?name
:Proxy2	:Person1	“Robert Jones”
:Proxy2	:Person2	“Bob Jones”

For 2009-08-18T09:40:00Z

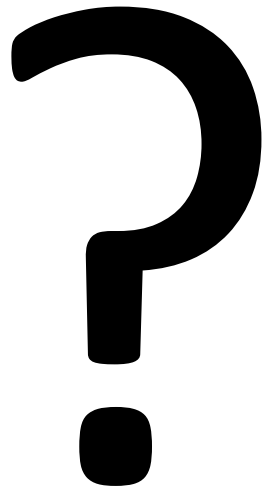
?person_proxy	?person	?name
:Proxy4	:Person2	“Bob Jones”

Benefits

- Enables analysts to explore previous analyses in the context of what was known *then* rather than what is known *now*
- Allows event-driven evolution of knowledge while preserving all previous KB states
- Provides non-destructive, reversible coreference resolution

http://asio.bbn.com/2010/10/stids/time.pdf

Questions



http://asio.bbn.com/2010/10/stids/time.pdf