Effective RDF Resource Identifiers for Integration of Structured Data Sources

Avoiding Coreferences and Other Common Pitfalls

Ian Emmons
iemmons@bbn.com

November 19, 2014

Raytheon
BBN Technologies
Introduction

Considerations for Any Data Source

IRIs for Non-RDF Sources of Record

Example Scenarios

Conclusion
Introduction

- Goal: Avoiding common pitfalls when creating International Resource Identifiers (IRIs) for the Resource Description Framework (RDF)
- Why: The act of naming is powerful and pivotal, so it’s important to not mess it up
- My background: Structured data integration
  - I don’t claim to cover the needs of unstructured data
Globally Unique Names

▶ Global Really Means Global
  ▶ The primary key of a database table need only be unique within the table, but
  ▶ A resource’s IRI must be *globally* unique

▶ Common Approaches
  ▶ Globally Unique Identifier (GUID): 16-byte number computed from attributes of the local computing environment
  ▶ Hierarchical: A hierarchy of string segments, each further narrowing the scope until a unique identifier is achieved
Hierarchical Naming with IRIs

http://org/dept/project/class/item

- Each segment carves out a subset of its predecessor’s namespace
- Scope narrows successively through the organization, department, project, class, and item names
- Segments often correspond to organizational entities with jurisdiction over that subset
Create IRIs Only With Your Segments

- Create new IRIs only with hierarchical scopes over which you have authority.
- An author in Department A should not create IRIs using Department B’s identifier segment.
  - Unless Department B has given permission.
- Otherwise you risk an ID collision.
- *Reuse of an entire IRI to identify the same entity (rather than minting a new IRI) is encouraged.*
IRI Schemes — http:

- By far the most popular
- Make unique identification easy, with extremely low cost
- Can be resolvable, if desired

http://org/dept/project/class/item
IRI Schemes — $\text{tag}$:

Similar to $http$, except:

- Explicitly non-resolvable
- Root may be either a domain name or an email address
- Formalizes the use of dates to prevent temporal collision

$\text{tag}:joe@org,2014-10-01:dept/proj/class/item$
IRI Schemes — \texttt{urn}:

- Requires registration of namespace
  - Makes it difficult to use with RDF
- But, easy conversion of GUIDs into IRIs via the already-registered \texttt{uuid} namespace:

\begin{verbatim}
urn:uuid:f81d4fae-7dec-11d0-a765-00a0c91e6bf6
\end{verbatim}

- Prepending an \texttt{http:} or \texttt{tag:} base works too:

\begin{verbatim}
http://org/etc#f81d4fae-7dec-11d0-a765-00a0c91e6bf6
\end{verbatim}
DNS Gotchas — Example Domains

- Unless your work is truly an example, avoid reserved names `example.org` and `example.com`.
- Using them temporarily lets you get your project moving quickly, but...
- It’s just begging for naming collisions.
- Replacing them later will be much more work than you think.
- Either acquire a proper domain name or use `tag:` with an email.
Contractors may be required to use the DNS name of their customer
Has the potential to violate naming authority
So be sure to coordinate with the DNS name owners within the customer organization to avoid naming collisions
DNS Gotchas — Resolvable IRIs

- Linked Open Data (LOD) IRIs are resolvable, fetching information about what they identify


- Requires DNS name to resolve to a real server
  - In DoD/IC, this means administrative effort, security authorization, and approval periods

- If web traffic must be secure, use https: scheme
- Separate dev, integration, and ops deployments mean separate IRI spaces
- Segregated networks can induce distinct DNS names, hampering data transfer across networks
Allowed Characters in Local Names

- The rule:
  - First character is a letter or underscore, and
  - Subsequent characters are letters, underscores, hyphens, periods, or digits.

- Allows XML-style QName abbreviations:
  http://org/dept/project/class/item
can be written as p:item in a document that declares the prefix p: to be the base IRI
IRIs as Content

- Don’t query by the information content of IRIs
  - E.g., regex filtering of the IRI string
- Instead, duplicate the information content in properties and query on those
IRIs for Non-RDF Sources of Record

- Semantic Web technologies are an effective approach for data integration
- Involves creating RDF representations of non-semantic sources of record
  - May store in a triple store or may generate on demand
- Data exists in multiple places and formats
- Therefore, proper identification is crucial
- Goal: Maintain a one-to-one relationship between entities in the source of record and their other identifiers
Uniqueness and Reproducibility

Two guiding principles for achieving our goal:
- **Unique**: A resource’s IRI must be globally unique, as discussed above
- **Reproducible**: Every time we form the RDF representation of an entity, the IRI we create should be same one

These correspond to the two halves of our one-to-one constraint goal:
- Uniqueness says no two data items may share the same identifier
- Reproducibility says no data item should have two distinct identifiers

Uniqueness is clear, but why reproducibility?
Example System

- Consider a system that queries a Relational Database (RDB) and then translates the result set into RDF
- For each row in the result set:
  - Create an IRI to represent the real-world entity represented by the row
  - Transform each column into a property of that resource
  - Foreign key columns become object properties
  - Other columns become datatype properties
Example Query

Suppose we issue the following query:

```sql
select employee_id, fname, lname, ssn
from Employee where employee_id < 40
```

and the result set looks like so:

<table>
<thead>
<tr>
<th>employee_id</th>
<th>fname</th>
<th>lname</th>
<th>ssn</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Robert</td>
<td>Smith</td>
<td>123-45-6789</td>
</tr>
<tr>
<td>37</td>
<td>Alice</td>
<td>Jones</td>
<td>987-65-4321</td>
</tr>
</tbody>
</table>
Example RDF

Then the resulting RDF might look like so:

```plaintext
prefix ont: <http://example.org/ont#>
prefix id: <http://example.org/id#>

urn:uuid:138ce39f-0434-4d16-b307-82b9206142b5
  a ont:Employee ;
  ont:employeeId 12 ;
  ont:firstName "Robert" ;
  ont:lastName "Smith" ;
  ont:ssn "123-45-6789" .

urn:uuid:1e036a52-7e1e-4a33-a48f-03837634f776
  a ont:Employee ;
  ont:employeeId 37 ;
  ont:firstName "Alice" ;
  ont:lastName "Jones" ;
  ont:ssn "987-65-4321" .
```
Second Example Query Result

Suppose a second query returns row 37. The same code will translate the new result set into RDF:

```
urn:uuid:f4139560-8c48-4b4c-a860-5d1bb9e02bdf
  a ont:Employee ;
  ont:employeeId 37 ;
  ont:firstName "Alice" ;
  ont:lastName "Jones" ;
  ont:ssn "987-65-4321" .
```

- Exactly as before, but the IRI differs
- We now have two distinct employees named Alice Jones with employee number 37
- We have created a coreference
- The reproducibility principle attempts to prevent this
Forming a Better IRI

- We can avoid the coreference by identifying employee 37 with the IRI `id:employee37`
- Using the primary key ensures that every time we encounter row 37, we will form the same IRI
- What if we encounter Alice Jones in a different context, in an RDB table at the Internal Revenue Service (IRS)?
  - Alice won’t be associated with the number 37, because that is an internal implementation detail of her employer’s database
- To avoid a coreference now, we might turn to Alice’s Social Security Number (SSN): `id:ssn-987-65-4321`
Generalizing

- For reproducibility, the information in a resource’s IRI should be *semantically intrinsic to the thing being identified*
- Ideally, whether we encounter the entity in the original context or in a different one
- The GUID is a poor choice, because it has no semantic content whatsoever
- The employee number is better, but it is semantically connected to Alice only within the context of the employee database
- The SSN is better still
The Caveat

- If we encounter the entity in *very* different contexts, there may be no identifying information held in common

- E.g., a third database from the immigration agency of a foreign government
  - The record of Alice’s visit while on vacation will not contain her SSN

- Thus, the uniqueness principle is a hard requirement, but the reproducibility principle is more of a guideline to strive for
  - Usually requires some carefully chosen compromises
Derivation from One Source of Record

- When the RDF is always derived from a consistent source of record, we can use its identifier, e.g., its primary key
- If that key is a GUID, then this is one occasion when using a GUID in your IRIs is reasonable
- If there is a key with semantic meaning intrinsic to the represented entity, that may be preferable
Derivation from Multiple Sources of Record

- Entities may occur in multiple sources of record
- Sources created within the same organization may have a common identifier system
- Entities with a standardized ID may be so identified in all the sources of record
  - E.g., airplane tail numbers or merchant ship registration
- But in general, the sources of record will not contain common identifiers
  - Create IRIs for each source independently
  - Identify and merge coreferences after the fact
Flat File Sources of Record

- Well-designed flat files are as discussed above
- But flat files are often ad hoc, with little thought given to identifiers
- Try to identify a set of columns that uniquely identify the row
- I have resorted to using all of the columns
  - This may create coreferences
  - But it has the best chance of uniqueness
- Avoid using the flat file’s name or path
  - File name and location can be changed without change to the file content
  - Moving or renaming the file will create coreferences for all of the contained entities
One Source of Record with Intermediate Processing

- Consider a source of record that feeds a process
  - The process transforms the data
  - We must render the output as RDF
- Ensure that identifiers from the source are carried throughout the processing chain
  - Enables IRI construction independent of the processing steps
  - Allows a consistent IRI for an entity that passes through multiple processing chains
Sub-Row Entities

- In simple cases, each row in a database table is one RDF entity
- But ontologies have more structure than database schemas
- What appears as just more columns in a table may be a separate entity in your ontology
- Thus, one database row becomes multiple related RDF resources
Sub-Row Entities (2)

- Some sub-row entities are logically part of the row entity
  - Form the IRI by appending additional key fields to the row entity’s IRI
- Other sub-row entities are logically independent of the row entity
  - Ask yourself, “If the columns containing the sub-row entity in two rows contain the same values, should the end result be two row entities related to one sub-row entity?”
  - If yes, then the sub-row entity is logically independent
  - In this case, form the IRI from only the columns containing the sub-row entity, as if they were in a separate table
Conclusion

➤ Hopefully, you now understand:
  ➤ What’s at stake when creating effective IRIs
  ➤ How to create IRIs for structured, non-RDF sources of record
  ➤ Special considerations of DoD, IC, and government contracting

➤ The next question: When you can’t avoid coreferences, what to do about them?
  ➤ That will have to wait for another time.