Types of Change for Authoritative Data

Reference we can use as a starting point: Change Management NOTES from LODLAM 2020

Overview

This document describes the types of changes that can occur in authoritative data. For each type, the following information will be explored:

- provide a basic description of the change
- include a discussion of the nuances and challenges for providers and consumers
- suggest potential approaches for representing the change
- give example data stream as it would appear in the change document

NOTE: At least initially, the example data will be shown in json, json-ld, or possibly some other format. Final recommendations for format will be in the Deliverable documents.

Types of Changes

New

Description:

This type provides information on a completely new entity.

Discussion:

If the data for the entity is included in the data stream, how would the edges of the graph be defined. For some authorities, the entity may be defined as just the data where the new entity URI is the subject (e.g. ontology is limited to skos). For others, data for an entity can includes data several layers away from the subject URI of the new entity (e.g. works described in BibFrame). Some may include blank nodes.

The wider the graph, the more chance there is that data expressed in the NEW data would overlap with existing data in the cache.

Option 2:

- Feasible, but should be optional because the graph for certain authorities may be computationally expensive
- Expense may be in terms of how many new documents will be included
- What is a full entity? How much of the graph needs to be included? What serialization should be included? What ontology should be included if the authority supports multiple ontologies?

How to determine relevancy of the new entity? For full cache, all new are relevant. If not, then the downstream consumer will need to decide which new entities are relevant. This can be supported by the notification process where an institution requests to be notified when the entity is added. Intermediate 3rd party provides a notification. Authorities may not have the band width to do this directly.

Service providers may find the change management documents described here as a way to provide these notifications.

Examples:

Approach:

Two possible approaches:

- OPTION 1: send change type (i.e. NEW) and URI of the new entity ; downstream consumers use the URI to fetch the new entity's data
- OPTION 2: send all data related to the entity along with the information in OPTION 1. There is a question about where the edges of the graph will be for the entity data.

Example Data Stream:

For Option 1:

```
{
  type: NEW,
  URI: https://uri.of.new.entity
}
```

For Option 2:

```
{
  type: NEW,
  URI: https://uri.of.new.entity
  entity: { full entity as json-ld }
}
```

Deleted

Description:

This type provides information on an entity that was completely. See also Deprecated.

Discussion:

For a deleted entity, the URI will no longer resolve.

Minimum: URI of the deleted thing.

If cache is a blob graph, then URI is sufficient.

Challenges:

- Are there other outside references to the triple? Can produce orphaned sub-chains. If access is always through the subject URI, there may be tolerance for kruft where the outside reference doesn't exist when you try to access it.
- How far out into the graph would you delete? - following blank nodes; following through other URIs.

Better not to delete. Better to deprecate (see below) and add see instead.

Approach:

Example Data Stream:

Deprecated

Description:

This type provides information on an entity that still exists in the authority, but is marked as deprecated meaning it should no longer be used. For deprecated entities, the URI will continue to resolve. See also, Split and Merge.

Discussion:

Deprecation typically happens when:

- an entity is no longer needed
- an entity is being replaced by another entity with its own URI
- an entity was split with each of the new entities having their own new URIs
- two or more entities were merged and the new entity with the merged data has its own new URI

In all cases, the entity remains in the authority allowing its URI to still resolve for preservation, backward compatibility, and to provide downstream consumers with time to update their references to the entity.

It is common practice for the deprecated entity to include information on what should be used instead.

Are there challenges when there are lots of deprecated entities?

- triples do grow as deprecations grow
- deprecations can be indexed separately or together, but this is driven by decisions made at the authority provider

Deprecations are fulfilling a need to...

- maintain URIs over time
- desire not to throw away any information
- gives history of the authority
- aids disambiguation

Are there times when a deprecated term should become deleted?

- would this be a choice of the consumer?

Implementation

- entity deleted, but URI is marked deprecated and remains in a registry of URIs

Can you deprecate a part of the graph for an entity?

- Example - a specific attribute: a label is no longer valid or an occupation is no longer valid
- Example - a link to another entity: a link to a role or agent

RDF is atemporal meaning that when you publish a graph, it exists and will always exist. A graph is true and is later succeeded by another graph. How to identify what is different? Can deprecation be used to identify what has changed? If you have two graphs, version 1 and version 2, the difference is the change. Challenge with knowing the edges of the graph are different in 1 vs. 2.

LOC views graph as corresponding to a MARC record? Each update of the system replaces the graph. This is possible because the graph is a blob for the subject URI.

Example: Series - name-title authority record vs. multiple - author name changed and in perhaps 10 records, the records are updated upstream

Complication comes in when starting with a single URI and connects to another URI that has its own definition outside the current URI's graph. If blank node, then this is not an issue.

Challenge with blobs is that if the blob references a URI that has a life outside the blob, then any data for the outside URI will be out of date within the blob. Potential solution is for the blob to only have URIs and pulls in other data for the outside URI live.

Similar to normal forms in database theory. Don't repeat data to avoid data redundancy.

Copy cataloging in BibFrame – work, instance, agent, item, etc. – Ideally, copy cataloging would be list of URIs. Significant practical difficulties.

Can we provide a sparql query that would indicate what needs to be updated in the graph?

Another option is to have delete, add, insert type diff commands that can be processed in order to make the graph changes.

Examples:

Approach:

Example Data Stream:

Canceled

Description:

MeSH uses the terminology that a heading is canceled. They generally include use_instead to identify the alternative. This may be the same as Deprecated.

Split

Description:

This type provides information on an entity that was split into two or more separate entities.

Discussion:

This commonly results in a new entities for each entity of the split. The original entity becomes deprecated or deleted. In some cases, the original entity for the split continues to exist with a different set of data.

This may be a sub-class of deleted or deprecated since the original entity is typically no longer valid under the original URI.

Examples:

Approach:

Example Data Stream:

Merge

Description:

This type provides information on two or more entities that were merged into a single entity.

Discussion:

This commonly results in a new entity with the data coming from each of the merged entities. The original entities become deprecated or deleted. In some cases, the merged entities are merged into one of the existing entities of the merge.

This may be a sub-class of deleted or deprecated since the original entities are typically no longer valid under the original URIs.

Examples:

- In MeSH, a common name is merged into the scientific name.
- Two works can merge into one.
- WikiData can get redundant entities for the same person that are then merged into one entity.

Approach:

Example Data Stream:

Changed

Description:

This type provides information on an existing entity with changed data.

Discussion:

Examples:

Approach:

Example Data Stream:

Label Change Only

Description:

This type provides information on an existing entity with changed label data.

Discussion:

This specifically meets the need of applications that cache labels. Question whether there should be caching of labels in downstream consumers? Several indicate that this is common practice.

Examples of use cases for caching labels in applications:

- Sinopia caches labels to avoid having to dereference URIs when viewing the data.
- Cache primary and variant labels when creating indices for searching.

Discussion on Primary Label:

- Primary label may not be a human understandable label. For example, ISNI number is primary label in the ISNI authority.
- Although it is common for primary label to be a single value, some authorities may allow multiple values for the primary label. For example, Wikidata has multiple primary labels for different languages and some may be missing for some languages.
- In some authorities, the primary label is critical and in others it may be optional or used for convenience.
- The predicate used to identify primary and variant labels can, and commonly do, differ between authorities.
- If DELETE_LABEL is supported, is there a concept of minimally viable data required for a valid entity and would this include a primary label? Thus, when the primary label is removed, does it require a replacement label? What happens for primary label if there is a delete but no add? Probably not a problem because primary label is likely to be minimal required data for a valid record by the authority, so if there is a change, it will always include both the DELETE and ADD.

Discussion on Variant Labels:

- It is common for there to be multiple variant labels. In some cases, variants may represent different languages.
- If DELETE_LABEL is supported, is it ok for a variant label to be deleted and not have a corresponding add? This seems ok.

Is there an ontology that describes types of changes? Activity streams accommodates this functionality.

- Is the process of defining this a process to create an ontology or similar to ontology creation?
- Can activity stream documents describe the types of change?
- Does this increase the level of agreement required?

Examples:

- LOC is looking at using a feed that provides information on authoritative labels. This is mostly used for name changes (e.g. person died and the death date is added to the label).

Challenges:

- The label in the graph may not be directly connected to the URI (e.g. <URI><PREDICATE><NEW_LABEL>). It may go through one or more other URIs or black nodes with a number of different predicates.
- What constitutes a label may vary between ontologies. Common label scenarios:
    - a single primary label
    - multiple primary labels
    - primary and variant labels
    - primary label is not human readable
    - no primary label
- Roles of consumers
    - Full cache processed fully by machine
    - Partial cache processed fully by machine or may need some human intervention
    - Human processing with a need to quickly see changes that are important from their perspective
- Ordering
    - Graphs are inherently not ordered; what implications are this?
    - Is there a way to express a diff with unordered data?
    - Desire is for an easy visual way to see, process, and focus in on a specific change
    - Perhaps use a notification where user subscribes to particular types of changes or certain data changes

    - There is a RDF graph source diff (https://www.w3.org/2001/sw/wiki/How_to_diff_RDF)
- Label changes in one language but not another

Approach:

Minimally need to include:

- URI
- NEW_LABEL - the new label to use
- PREDICATE (or some other identifier) - identifies which type of label is being replaced - perhaps should be LDPATH instead to address a label that is farther down the graph from the subjectURI.

NOTE: This can be represented as a triple. `<URI> <PREDICATE> "new label"@en`

To be able to replace a label, also need:

- OLD_LABEL - the value of the literal that is being replaced

NOTE: This would remove triple. `<URI> <PREDICATE> "old label"@en`

OPTION 1: Single type LABEL_CHANGE - all change information is in a single change entry

OPTION 2: Two change entries, one to DELETE_LABEL being replaced, followed by ADD_LABEL to add the new label. *Question: Will this provide an adequate indicator to downstream consumers allowing them to update cached values?*

Example Data Stream:

For Option 1:

```
{
  "type": "LABEL_CHANGE",
  "URI": "https://uri.of.changing.entity",
  "PREDICATE": "skos:prefLabel",
  "NEW_LABEL": "new value"@en,
  "OLD_LABEL": "old value"@en
}
```

```
{
  "type": "LABEL_CHANGE",
  "ADD": "https://uri.of.changing.entity",
  "PREDICATE": "skos:prefLabel",
  "NEW_LABEL": "new value"@en,
  "OLD_LABEL": "old value"@en
}
```

For Option 2:

```
{
  "type": "DELETE_LABEL",
  "URI": "https://uri.of.changing.entity",
  "PREDICATE": "skos:prefLabel",
  "LABEL": "old value"@en
}
```

```
{
  "type": "ADD_LABEL",
  "URI": "https://uri.of.changing.entity",
  "PREDICATE": "skos:prefLabel",
  "LABEL": "new value"@en,
}
```

Other Considerations and Questions

Are the following handled differently when managing change?

- High velocity changes vs. Low velocity changes
    - labels typically change less frequently
- High impact changes vs. Low impact changes
    - Importance of change

Notification when an initial search fails to match and later a match becomes available. Same for partial match.

- Null to actual value.
- How to distinguish between a missing subject vs. a typo?
- Notification of missed value to a specific user. This would likely be outside the change management stream.
- Notification of submitted entities that were accepted and those that were not accepted.
    - Ex. Cataloger adds record to OCLC, but it is not immediately available. Get notification when it is available or rejected.

Would the types of changes be different for different types of entities (e.g. MeSH Subjects vs. Names)?

- The goal is to have the types be flexible enough to use the same basic data for each type for all authorities and all types of entities.

Could information be expressed as a DIFF similar to how GIT does DIFFs?

- visual diffs are easy for humans to quickly process and focus in on the area of concern
- how would this work for computer processing?
- See more notes in General Discussion on Approaches on Existing Change Management Approaches