When androids dream of ... oh you know
RDF Management Process: Translate UI actions into RDF adds/deletions

Conforms to specific patterns

Search Index: Solr or ElasticSearch

Authorization/Permissions

GraphQl

Search Index API

Translation

Ingest/Update

Combine

RDF Data

Shapes

Conforms to specific patterns

RDF Service/SPOARL

Authorization/Permissions

Triple Pattern Fragments

Triplestore

Ontology

Data

Asset Store

Configuration: Mapping/Events and Triggers

Configuration: Results/Events and Triggers

Reasoning Process

RDF Service/SPOARL

UI: Display

UI: Also with editing

Authorization/Permissions

UI:

Display
Extract/Transform:
Map to common model based on shapes

Validate data based on shapes.
Deduplicate/disambiguate.

Generate serialization of RDF Data in requested format: N3, JSON-LD, quads, etc.

Additions that can be sent to VIVO using the ingest/update API
Persons, publications, grants, and related research information is harvested from multiple sources and made available as a change set with additions to be stored in the triple store. The ingest API (which accepts additions and rejections) populates the triple store with the provided data. Images can be added separately to the asset store using the image upload process/API.
The reasoning process can be triggered independently or in conjunction with an update. A potential new feature would be to enable configuration of the reasoned to specify through configuration what triples should be inferred and added to the triple store.

---

**Event Configuration:**
Trigger reasoning when requested.
Trigger reasoning when new triples are added.

**Reasoning Process**

**Added triple:**
<person> rdf:type vivo:Faculty.

**Reasoner Configuration:**
Create following inferences based on added triples.

**Add:**
Class hierarchy for type
<person> rdf:type foaf:Person
The indexing process, further described on the next slide, can be configured to run when specific events occur (such as update to a particular ‘shape’ or entity type or on ingest) and/or on request. Queries define how information will be extracted and added to the search index.

The “index” may actually be multiple indices, with one central general search index and multiple shape-based indices.

**Index update:**
Extract person, publication, grant relationships

**Granular/Shape update:**
Extract person and related info to update index

<person> authored <publication>.
<person> rdfs:label “Grapes of Wrath”.
<publication> output of <grant>.
<grant> funded by <organization>.

<table>
<thead>
<tr>
<th>Person:</th>
<th>Author</th>
<th>Publication</th>
</tr>
</thead>
<tbody>
<tr>
<td>Label</td>
<td>Grapes of Wrath</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Publication:</th>
<th>Label</th>
<th>Output Of</th>
</tr>
</thead>
<tbody>
<tr>
<td>Grapes of Wrath</td>
<td>Grant</td>
<td></td>
</tr>
</tbody>
</table>
**Event Configuration:**
- Trigger Indexing When Indexing Request Received.
- Trigger Indexing When New Triples Added using Ingest/Update API.

**Mapping Configuration (Index field => Query):**
- Entity Type => Select ?type WHERE {?individual ...}
- Person Position => Select ?label WHERE {?person...}
- Publication Author => Select ?person WHERE {?publication ...}
- Grant Investigator => Select ?person WHERE {?grant ...}

**Search Indexer:**
- Document Modifiers (i.e. mapping config)

**Search Engine Interface**
- Execute Queries
- Add structured document
- Populate index

**Implementation:**
- Solr Implementation
- Elastic Search Implementation

**Shapes:**
- Person Shape
- Publication Shape
- Grant Shape
Indexing decisions:
What portion of index is updated when?

*Shapes could inform indexing process.*

E.g. If code can determine a person’s info has been updated, that shape can be used to make the appropriate updates.
Externalization may involve channeling requests to triple store through an API layer e.g. RDF Service Interface or SPARQL API.

- Output graph: single or entire graph store
- Submit change set: + Additions - Retractions
- Create URI
- Execute SPARQL Query
- Compare graph with serialization