Putting Enterprise Ontologies Into Production

Michael Uschold

April 21, 2021

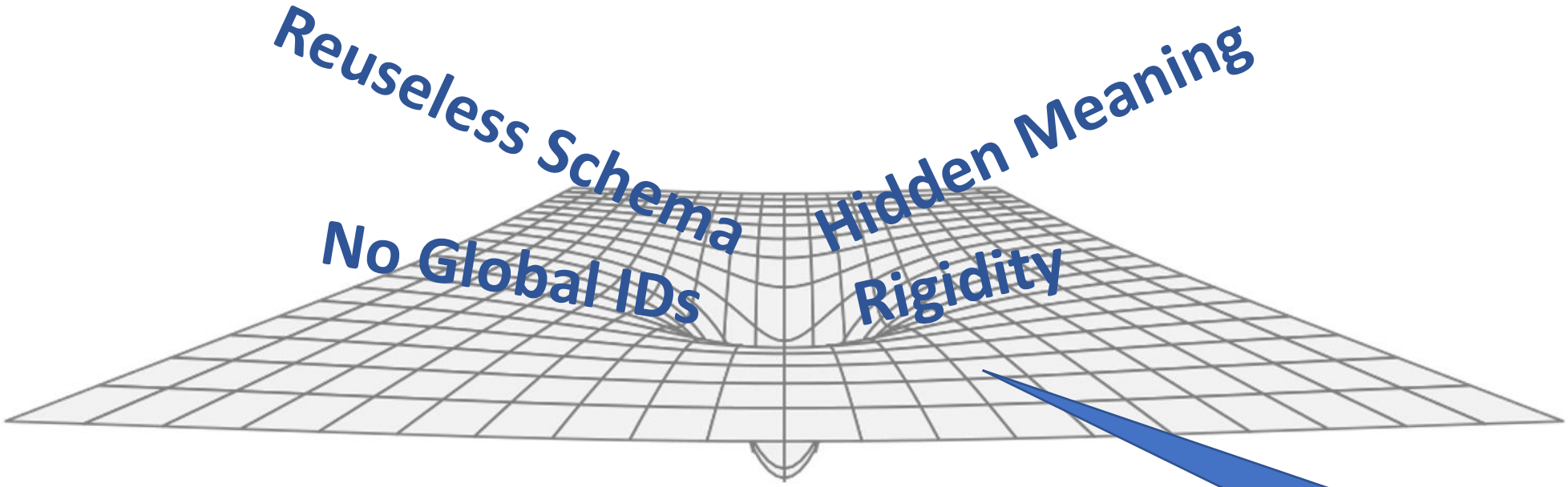
semantic arts

Outline

- Enterprise Ontology: the answer to Semantic Silos
- gist: a critical enabler
- Think Big - Start Small
- Things are evolving
  - Internal reuse
  - Using existing vocabularies and ontologies
  - Consistent patterns and style
  - Semantic hygiene: SHACL & SPARQL
  - Scripts and pipelines
  - Change management

SILOS: Root Cause Analysis

Drivers that exert a strong gravitational pull towards silos.



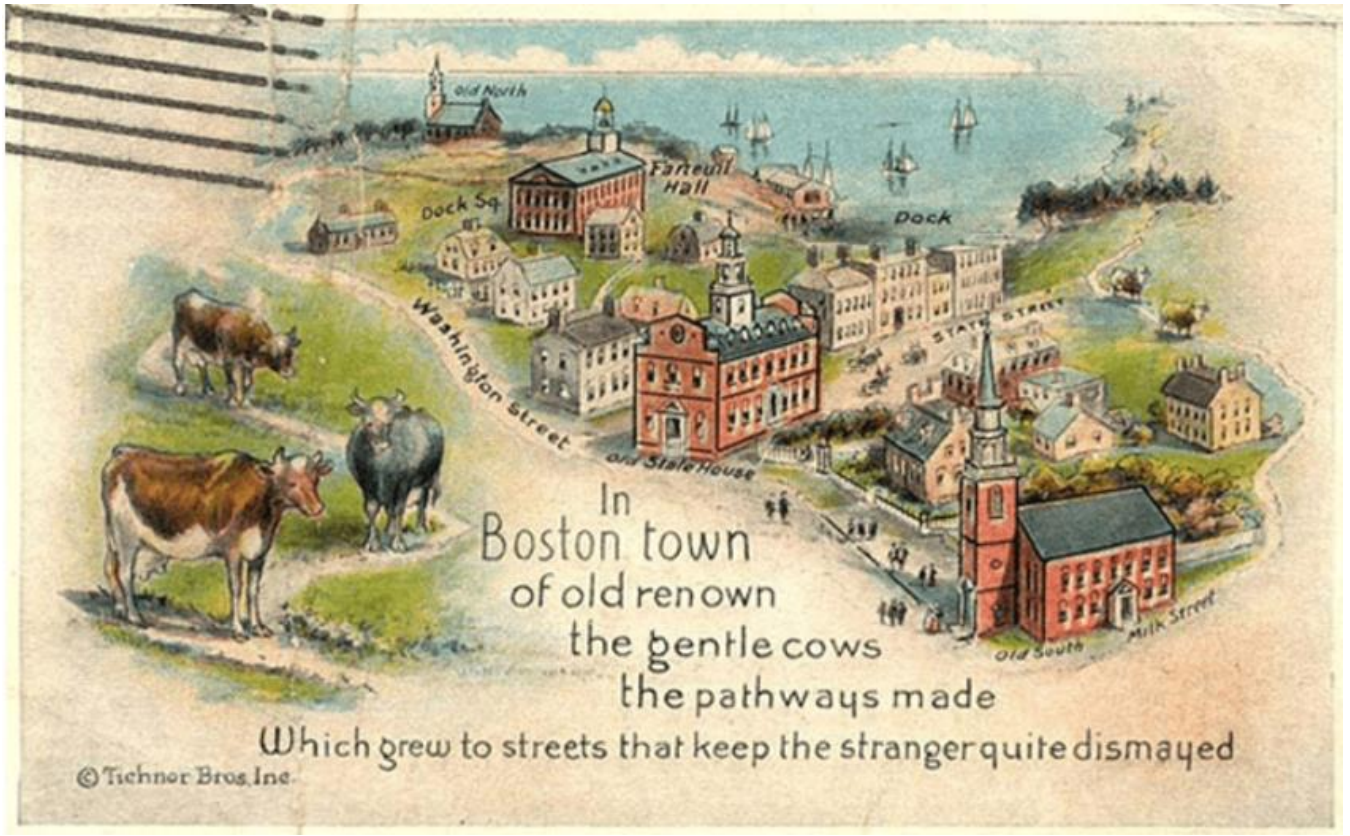


Much of this is rooted in traditional relational database technology.

semantic arts

Beware of Paving the Cow Paths

- Boston streets are paved cow paths.
- Expedient, but worse than purpose-built street plan.

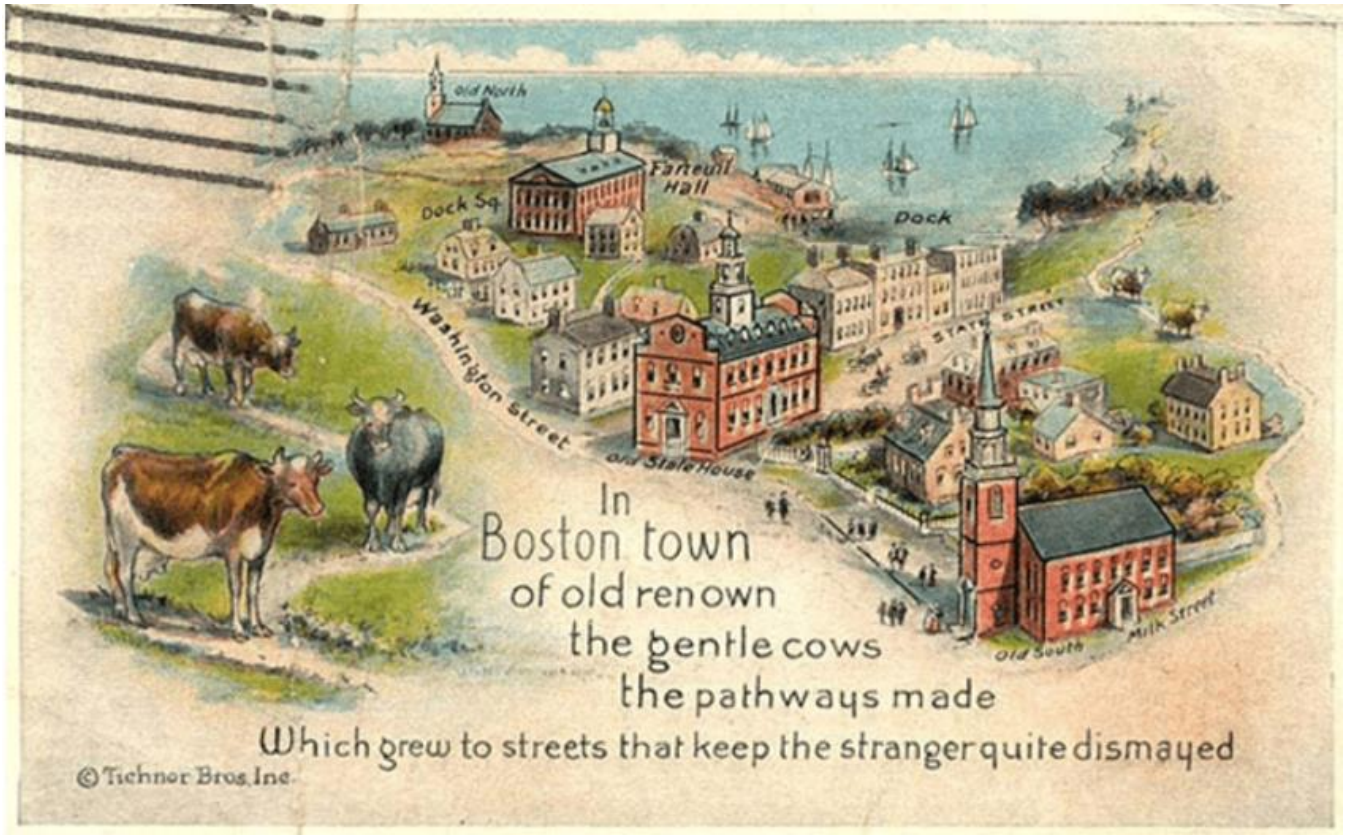


semantic arts

Beware of Paving the Cow Paths

- If you independently convert local groups of silos you get local benefits.

- BUT:
  You may still end up with silos unless you have enterprise-wide coordination.



semantic arts

Beware of Paving the Cow Paths

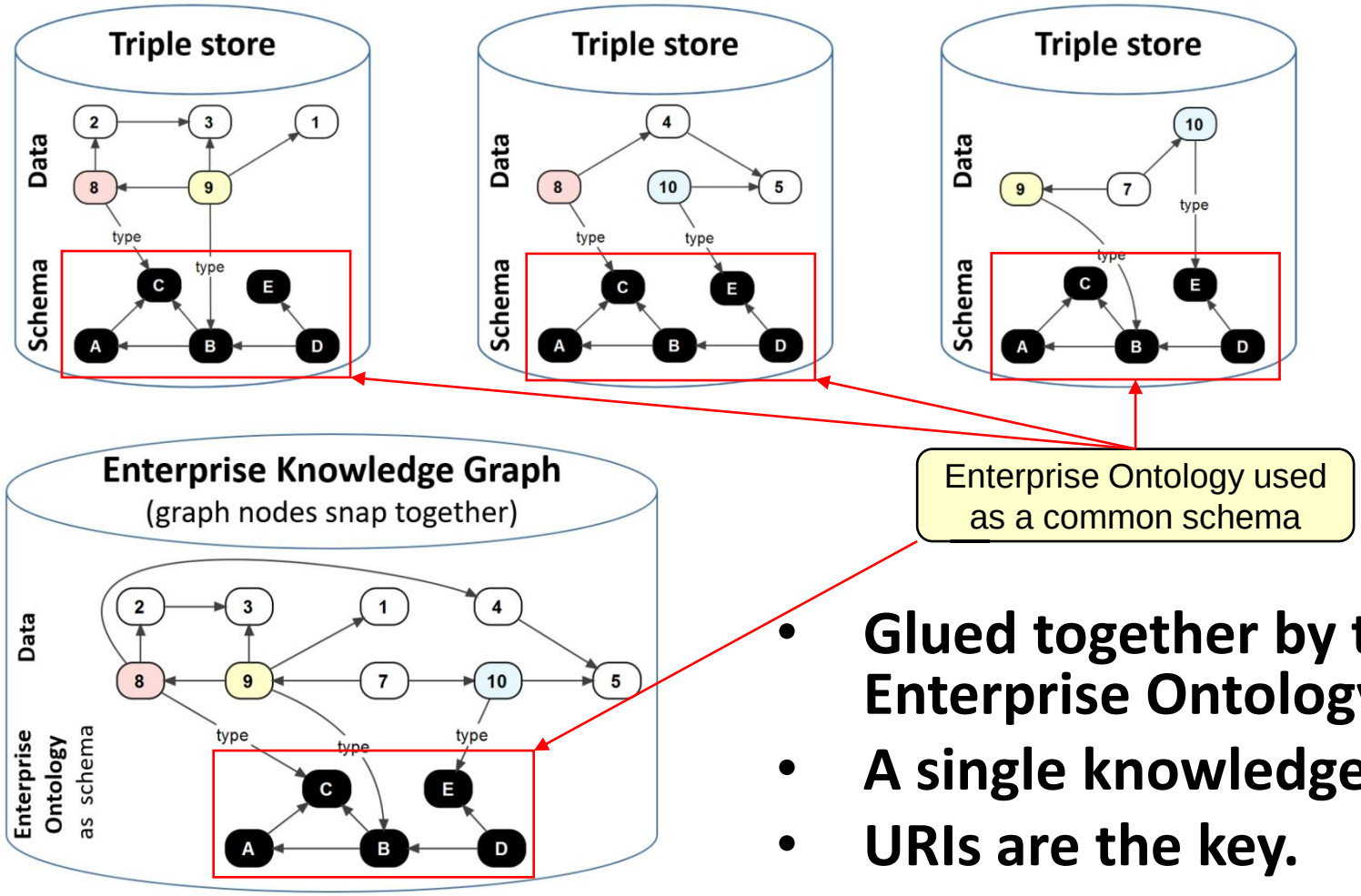
Layering semantic technology over silos is like paving over the cow paths.



semantic arts

Use Enterprise Ontology to Get Rid of the Silos!!!

- **These look like silos, but they are not. Why?**

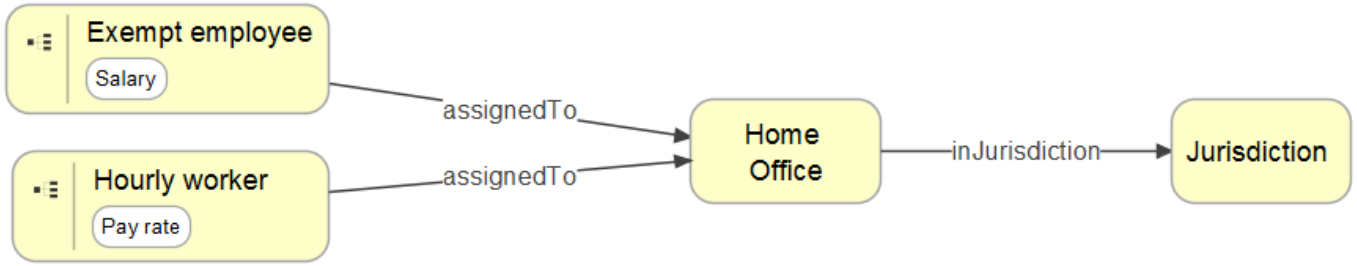


- **Glued together by the Enterprise Ontology**
- **A single knowledge graph**
- **URIs are the key.**

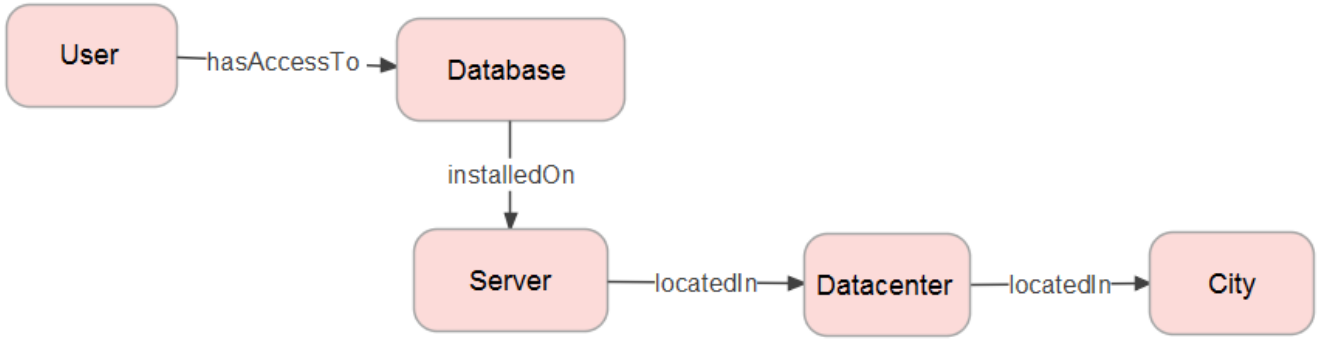
semantic arts

Modularity, Reuse & Federated Queries

- HR Department:



- IT Department:

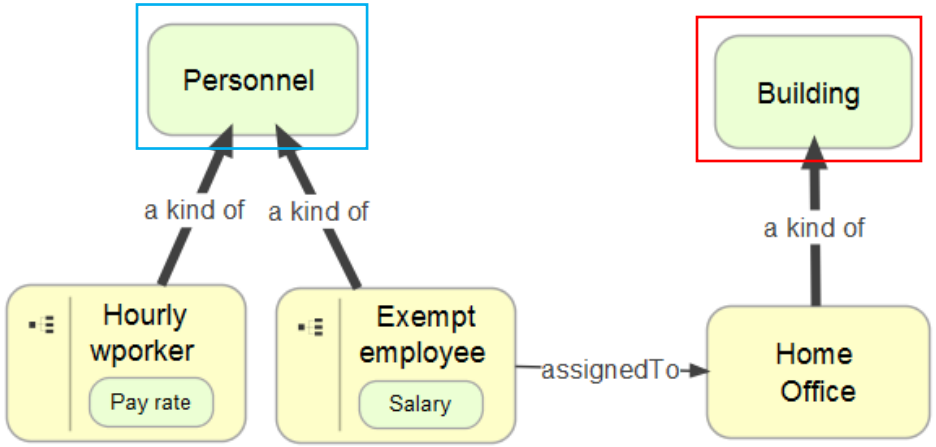


- How to do a federated query to identify all Personnel?

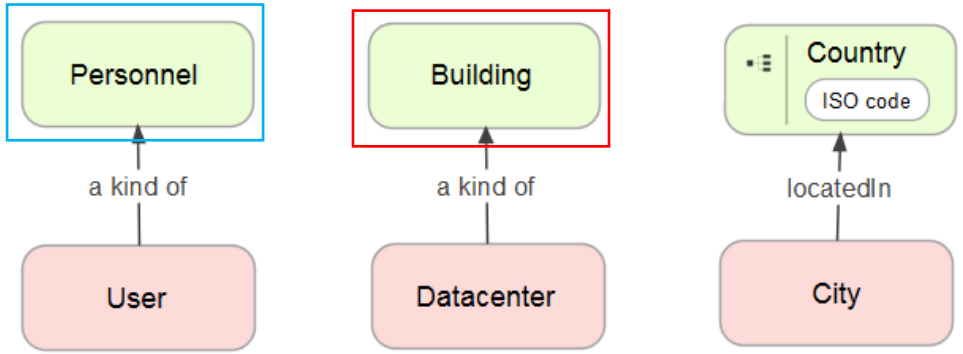
semantic arts

Example: connect to generic schema

- HR Department:



- IT Department:

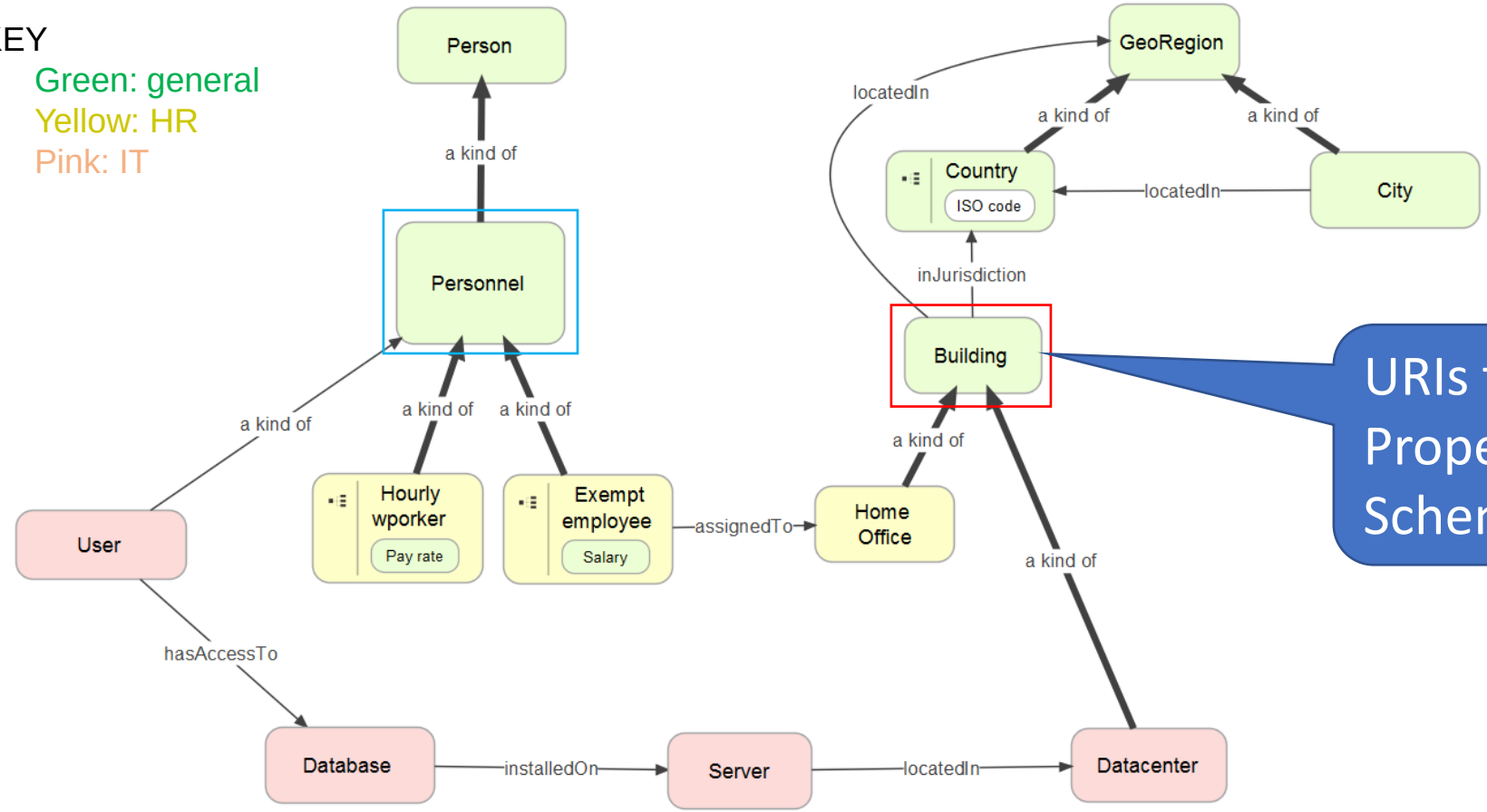


Identical concepts will collapse into one.
- Personnel
- Building

semantic arts

Example: as a single graph

KEY
- Green: general
- Yellow: HR
- Pink: IT



- Federated query to identify all Personnel becomes possible.
- Jurisdiction, City and Country are all GeoRegions

semantic arts

Don’t Let anyone tell you it is easy. Its not.

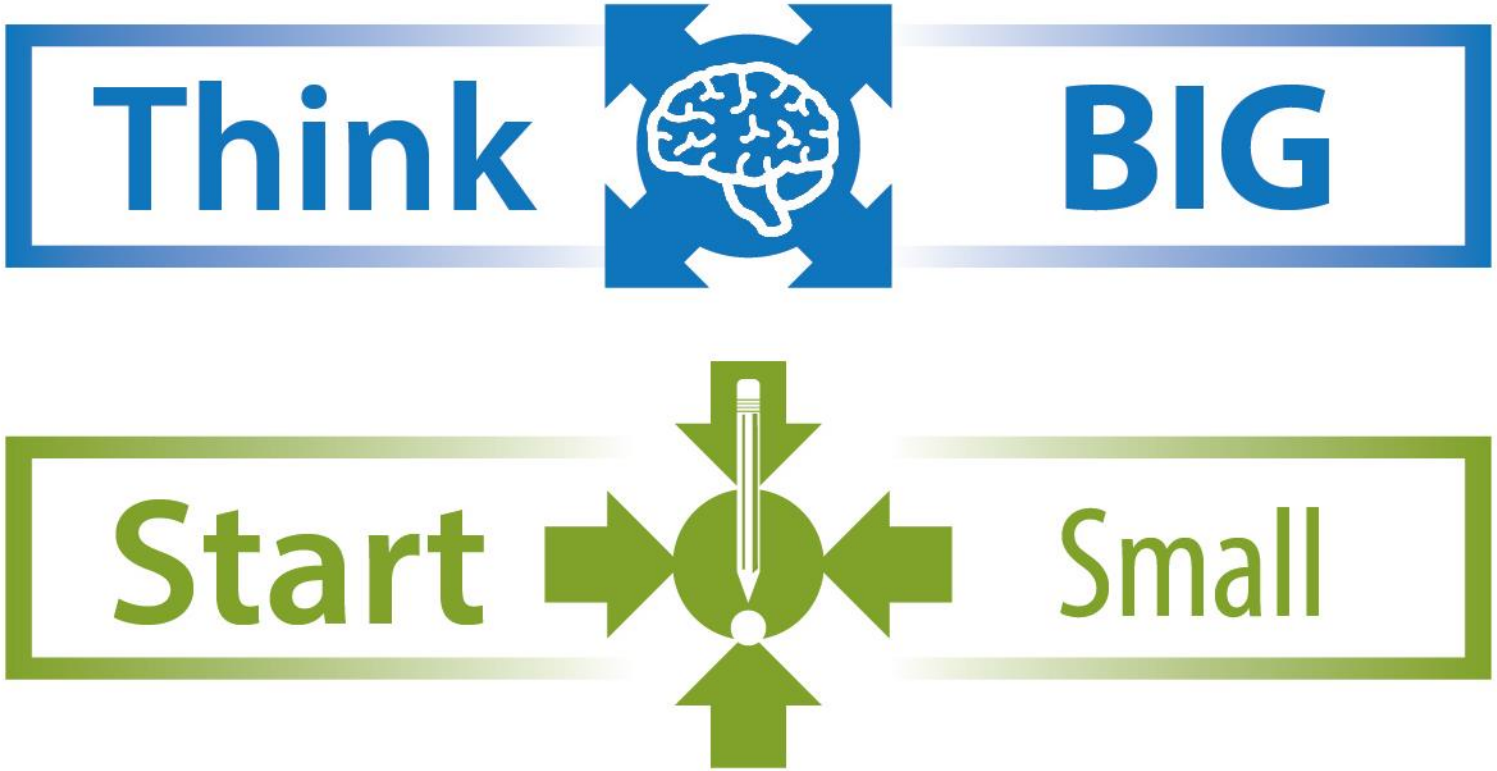
- It’s a lot of work
- Agreeing on terminology
- Agreeing on minting patterns for URIs
- Updating schema

semantic arts

gist: a critical enabler

- A upper enterprise ontology
- All of our projects use it
- Saves time
- Produces better ontologies
- Common usage across the enterprise avoids silos.
- Easier to share ontology work across projects and clients

Think Big - Start Small



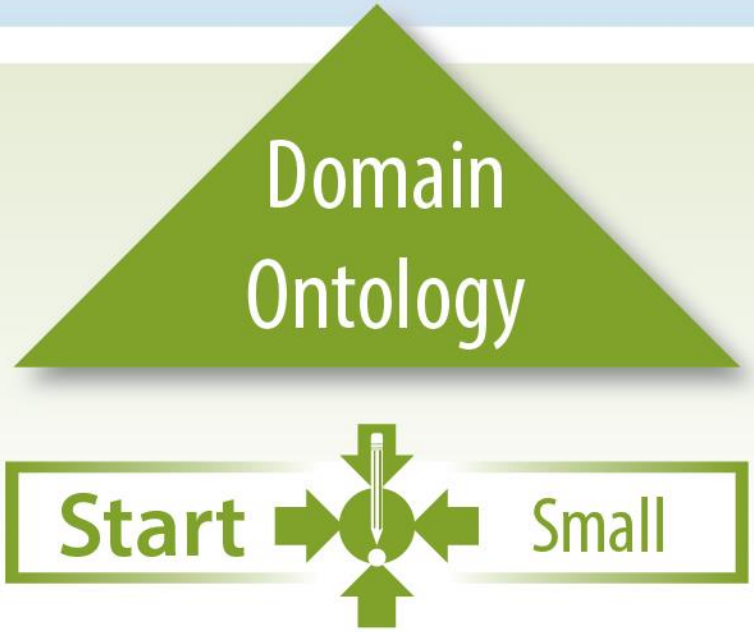
General Approach

A collaborative workshop-driven approach focused on delivering real business value



ENTERPRISE CORE ONTOLOGY

Thinking big, by itself, tends to lead to grandiose projects that often don't see the light of day



Starting small on the other hand, by itself, tends to create more stand-alone projects and further extends the silo landscape

We've found combining the two approaches and time boxing them leads to the best result and a way to create reusability

Key deliverables and objectives

A collaborative workshop-driven approach focused on delivering real business value



Objective: Cover the maximum number of future subdomains with the fewest concepts

Key Deliverable: The Enterprise Ontology



Objective: Demonstrate true business value by querying previously difficult-to-integrate data to capture new business insights

Key Deliverable: A populated triple store with “real” data from at least two existing sources

Drill down on some key interim deliverables

Define Key concepts and relationships. Draft and Evolve conceptual design



- **Flush out and review the Enterprise Ontology**
- **Articulate the Business Model**
- **”Parti” a high-level sketch or diagram of key components and their interrelationships**
- **Build an Inventory of concepts that define the business**

Drill down on some key interim deliverables

Build and populate Knowledge Graph. Continue to refine and validate design



- **Inventory Data Sources**
- **Formulate requirements to Frame Competency Questions**
- **Detail and build subdomain ontology**
- **Demonstrate Competency Question answers via SPARQL query**
- **Harvest and conform data (TARQL or equivalent)**
- **Form minting and triple patterns**

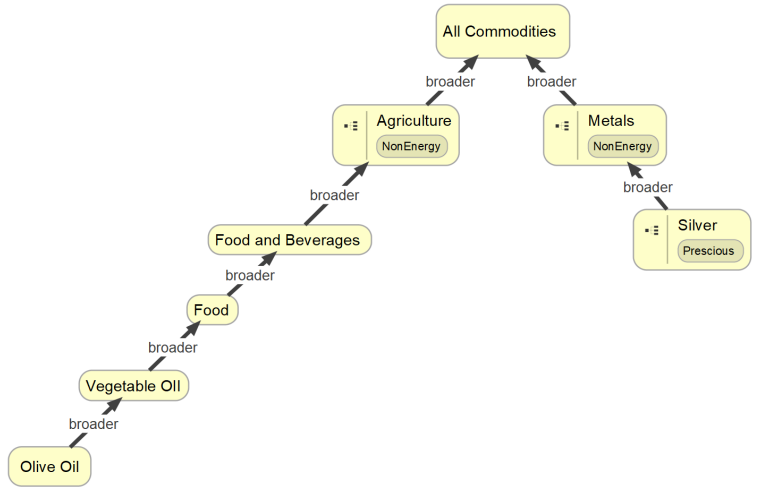
Internal Reuse

Lots of things are common across different companies. We borrow ideas mostly, sometimes, large chunks of OWL.

- Measures, Characteristics, Specifications and Observations
  - Formulas and recipes for industrial manufacturing of ‘stuff’
  - Electrical power
  - Price assessments
  - Medical observations
  - Economic indicators
  - Defining characteristics of an industry

Internal Reuse

- Hierarchies of categories and types
  - Products
  - Commodities



- Temporal relations: what was true when
  - Tenure of a CEO
  - Tenure of heading up a committee
  - Registered address of a company
  - Country evolution - when was the USSR a thing?

Moving towards Use of Existing Vocabularies & Ontologies

- Now using SKOS annotations.
- GeoNames
- Medical taxonomies

Sometimes we borrow ideas.

Sometimes we just link things up

Using existing vocabularies and ontologies

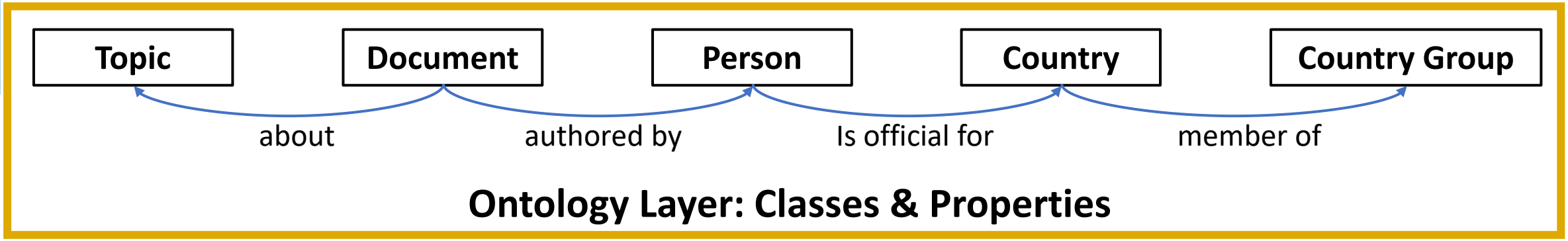
- These are the exceptions.
- Most reuse is reuse of ideas and patterns.
- We look for inspiration.
- Gotchas:
  - Transliterated from a relational schema
  - Over-constrained domain and range
  - Glorified taxonomies, no restrictions
  - Property proliferation – for domain/range variations
  - Class proliferation: 300,000 classes in some medical ontologies.
  - Better to have a hierarchy of category individuals.

Consistent Patterns & Style

We have a style guide for gist that we use for all of our ontologies.

- Always use EDM Council's rdf serializer for collaboration using git
- Conventions for using annotations: SKOS if possible.
- Conventions for naming classes, properties
  - Camel casing
  - Acronyms & Abbreviations: UsaCurrency not USACurrency or USA_Currency
- Separate layer for enumerated lists and taxonomies
- Separate namespace convention for each layer
- Minting IRIs for individuals – the idea of a 'midfix'

Enterprise Ontology Architecture



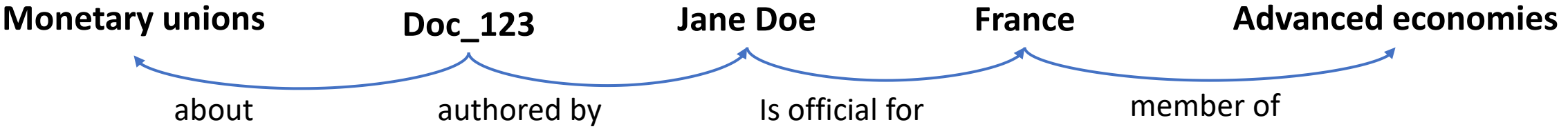
Econ & Financial
- Balance of payments
  - Adjusted process
- Economic integration
  - Monetary unions

Document types
- Agenda
- Briefing
- Claim

Geographic
- Eastern Europe
- Western Europe
  - France

Country Group
- Advanced Economies
- Fragile States

Taxonomy Layer: Lists and Hierarchies



Linked data Layer: relationships among individual people, countries, documents …

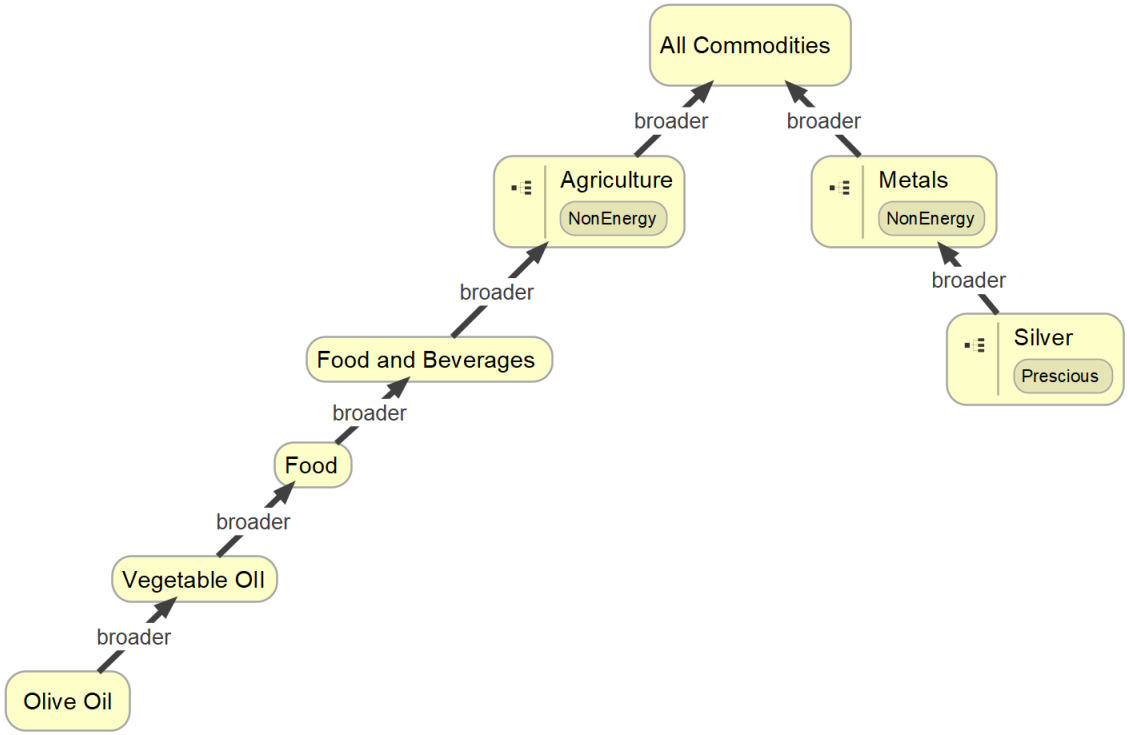
semantic arts

Example Data and Taxonomy

```
imfeod:_EconomicIndicator_PZPIOLOIL
    a imfeo:EconomicIndicator ;
    skos:prefLabel "Price of olive oil"^^xsd:string ;
    gist:categorizedBy
        imfeox:_CommodityType_OliveOil ,
        imfeox:_CtsSector_Prices ,
        imfeox:_CtsTopic_CommodityPrices
        ;
    gist:description "Prices, Commodity Prices, All
        Commodities, Non-energy, Agriculture, Food and
        Beverages, Food, Vegetable Oil, Olive Oil"^^xsd:string
         ;
    .
```

```
imfeox:_CommodityType_VegetableOil
    a imfeo:CommodityType ;
    skos:broader imfeox:_CommodityType_Food ;
    skos:prefLabel "Vegetable Oil"^^xsd:string .

imfeox:_CommodityType_OliveOil
    a imfeo:CommodityType ;
    skos:broader imfeox:_CommodityType_VegetableOil ;
    skos:prefLabel "Olive Oil"^^xsd:string .
```



Semantic Hygiene

Different things to check

- Which artifact is it relevant to: Ontology? Data?
- Are they domain dependent?
- Generic for any ontology – find likely mistakes
  - is there a named inverse for a symmetric property?
  - Is there a transitive property whose domain and range are different?
- Ensure matches style conventions (e.g. camelCasing)
- Ensure prefLabel is aligned with the IRI fragment for classes and properties.
- For datasets
  - Are there any untyped subjects or objects in any triples?

Semantic Hygiene

Endless things to check that are domain dependent.

We find a lot of mistakes that way, which helps clients clean their data.

Mostly use SHACL. Some SPARQL. Occasional code.

Libraries of queries called from scripts.

Change Management

What happens when a new version of an information artifact is released?

- What other artifacts could be affected?
- What changes need to be made?
- Who is responsible for making the changes?
- How are they notified that changes may be necessary?
- How to maintain version compatibility?
- How can we track dependencies?

- Ontologies
- Software
- Taxonomies
- Shapes

It's all about tracking dependencies.

Principles & Requirements

Basic dependency

- A particular version of an item depends on another item, whose version must be a particular value or in a range.

Generated dataset provenance

- using what versions of other items
- by what software application
- from what data source
- fresh vs. incremental
- when generated

• Ontologies • Software
• Taxonomies • Shapes

Principles & Requirements

- Datasets do not have versions, they used versioned artifacts.
- Automate as much as possible for quality and efficiency.
- A non-breaking change should require minimal effort
- Only things that are delivered independently should have their own version.
- Allow collections of items to be versioned together as one. (e.g. group on ontologies)

Update Process

Baseline: there is a mechanism for curators to learn about changes

1. Developers make change
2. Curators determine impact on artifacts under their governance.
3. Curators ensure required changes are made.
4. Curators ensure dependencies are updated.

Make Change & Determine Impact

Example: IRI of a concept in an ontology is renamed to something else.

- A compatibility-breaking change, requiring a major version bump.
- A new version number is given to the versioned ontology (or ontology collection).
- Identify all the versioned items that are dependent on the versioned item that changed.
- Curators determine the impact of any versioned items under their governance.

Currently in production, including automated validation checks

Scripts and Pipelines

A lot of things are automated.

- Hygiene queries
- Version validation
- ETL – csv to TARQL to triples.

- Here is something that was unexpected…

Speeding Up Data Refresh

- The client's relational data was very inter-connected.
- A very complex query with joins galore took 12 hrs to run to get a fresh .csv to convert to ontology-backed triples.
- We introduced a new step in the process.
- We transliterated pieces of the relational model into a pseudo-ontology.
- Used TARQL to create "flat" triples based on the pseudo-ontology.
- Used SPARQL CONSTRUCT to do all the work that the SQL query was doing.
- It was all automated in a pipeline
- The total refresh process went from 12+ hours to around 1.
- We may have another situation where this approach could help.

semantic arts

Summary

- Putting things into production is a whole new world, compared to PoCs.
- Name of the game is:
  - Reuse
  - Consistency
  - Checking
  - Automation