DATA MODEL REVERSE-ENGINEERED INTO ONTOLOGY

Abstract

This article describes a novel approach transforming Logical Data Models (LDM), rather than database schemas into ontologies. The uses cases are for Knowledge Graphs, Operational and Enterprise Ontologies, which benefit from using LDM logical names and subtypes rather than abbreviated database names and foreign keys. I revisit the isomorphism and bi-directional Metadata Sets, central to the Configurable Ontology to Data Model Transformation (CODT). Finally, for an optimal ontology, we must reverse engineer associative entities into object properties – not classes.

Background

Model transformations migrate a model into another type of model. Data Architects use the term forward-engineering for transformations of a higher-level logical model into a physical model and subsequently into a database schema. The term reverse-engineering refers to the automated conversion of a database schema into a physical model. Semantic Enterprise Information Architecture (SEIA) places the ontology at the apex with derived, forward-engineered models for data and object. Hence, I use the term reverse-engineered for data models transformed into ontologies. Note that "reverse" doesn't mean that a higher-level must be pre-existing. We reverse-engineer databases for which we have no physical model, and we can reverse-engineer data models without having a reference ontology.

While simplified ontology-to-data model mappings have been widely published, there was no tool to transform an ontology into a useful data model. Hence, 900 users downloaded the Open Source version of the Financial Industry Business Data Model (FIB-DM), derived with CODT from the industry-standard domain ontology.

Published reverse transformation research and tooling solely address databases – there is no tool or research to reverse-engineer an ontology from PowerDesigner, ERWin, or other data modeling tools.

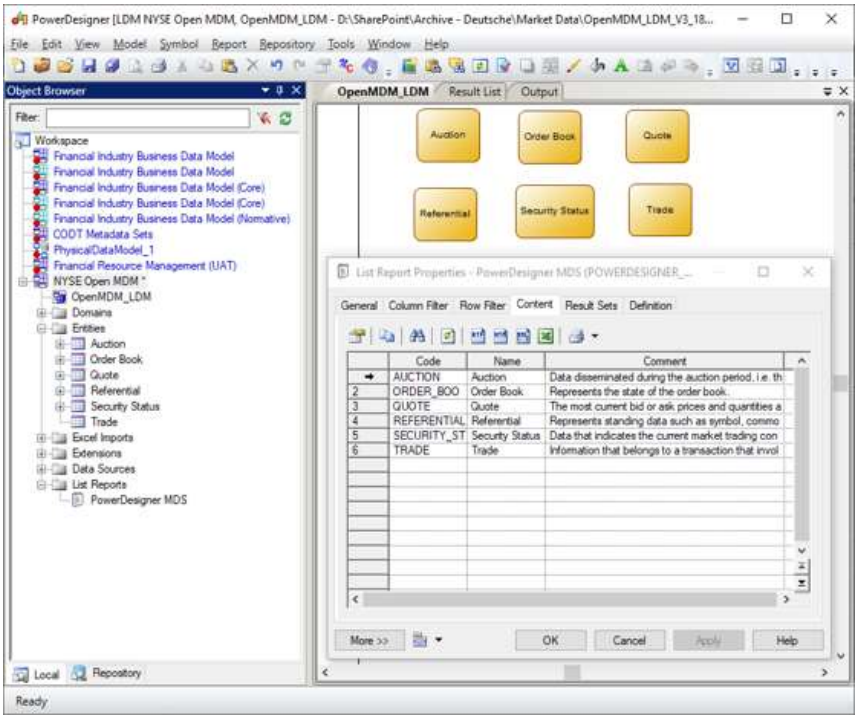
The first FIB-DM article, "Finance Ontology transformed into an Enterprise Data Model," and the CODT utility patent application state that the transformation technology is bi-directional, but they don't provide the rationale and detail. The second article showed that the "Ontology Class- and Data Model Entity-hierarchy" are the same, and the third states that "Object Properties are Associative Entities."

This final installment examines the reverse transformation.

Overview – the Trading Model example

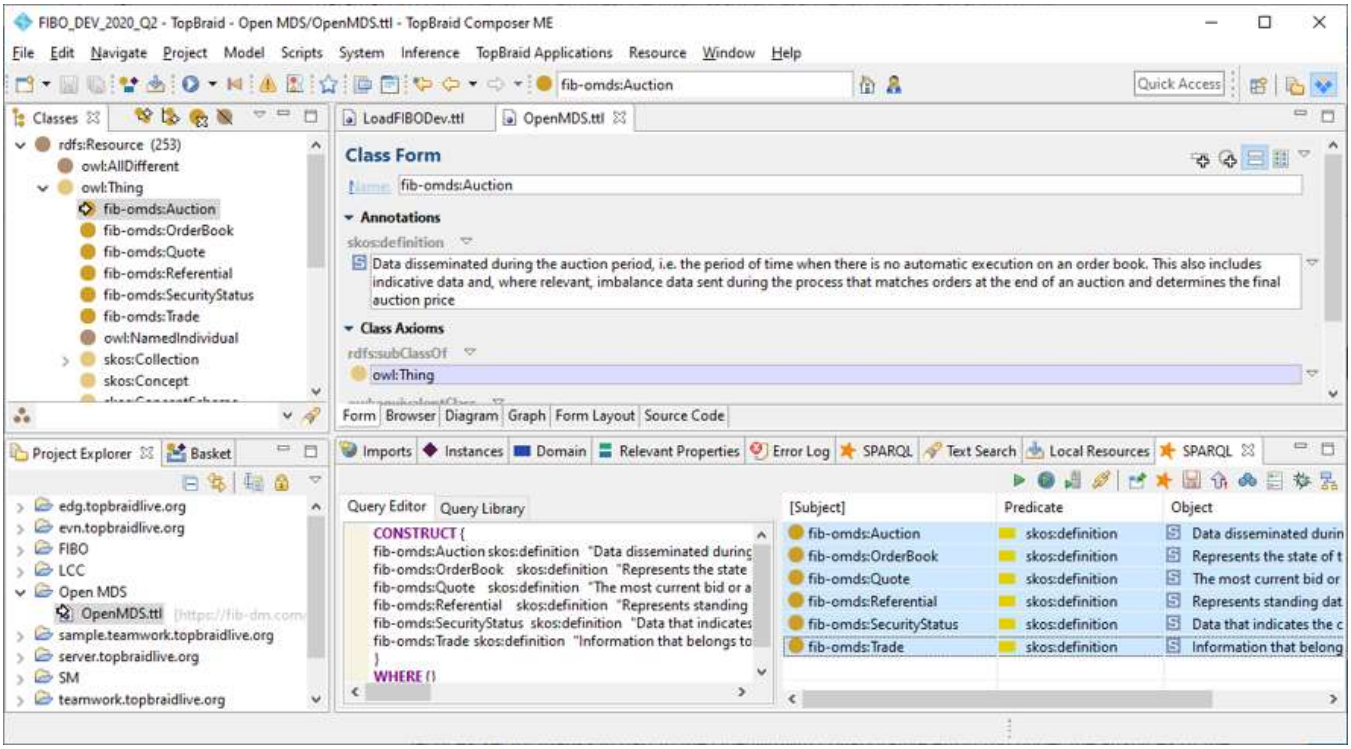
In CODT POC class, "Semantics for extra-large Banks," students transform first the FIBO and then their proprietary ontology extensions into data models. The course briefly covers the reverse transformation.

The input, our starting point, is a logical model in a data modeling tool. The screenshot shows a subset of the New York Stock Exchange's OpenMAMA (Middleware Agnostic Messaging API) with entities for Auction, Order Book, Quote, Security and Trade.



CODT Reverse example – Open MDM data model in PowerDesigner

Suppose we want to create an OpenMAMA ontology. Rather than typing in classes and properties, we transform the data model into RDF/OWL. The image below shows the classes for Auction, Order Book, Quote, Security, and Trade, reverse-engineered from data model entities, in an ontology editor, TopBrad Composer.



CODT Reverse-Engineered ontology in TopBraid Composer

Logical data model names convert into ontology Camel Case notation, “Order Book” becomes “fib-omds:OrderBook” with a configured ontology Prefix and Namespace. The transformation also harvests the Entity Comment as an ontology annotation property, the SKOS Definition.

The transformation process is a reversal of the ontology to data model transformation. The symbol left-hand side of the diagram below is the PowerDesigner icon.

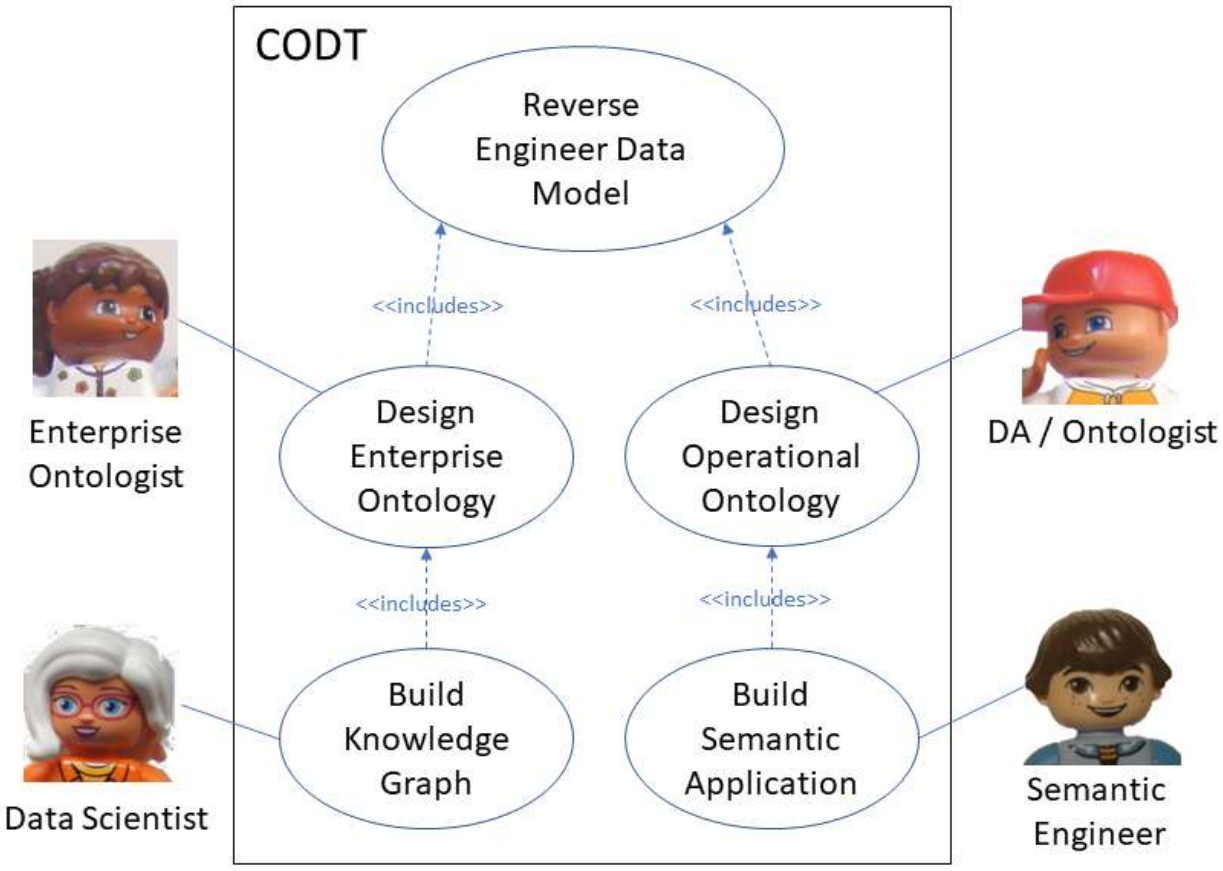
CODT Data Model to Ontology – Extract Transform Load

We extract data model metadata from the data modeling tool, transform Entity/Relationship metadata into ontology metadata, and load into the ontology editor or RDF database – **ETL**:

- Extract: The Data Architect generates List Reports matching the Data Modeling tool-specific MDS. Power Query populates the Metadata sets, performing basic data cleansing.
- Transform: The Entity-Relationship Metadata Sets populate from tool-specific metadata set.
- Load: The Ontology MDS populates from the Entity-Relationship MDS. Power Queries and formulas break the data set down into triples. We load in triples into the ontology platform, using SPARQL CONSTRUCT or bulk insert.

Use Cases

The UML diagram depicts the CODT system with the primary use case to reverse-engineer a data model into an ontology. Just like FIB-DM saves manual work of entering ontology-derived data model objects into the modeling tool, the reverse-engineered ontology provides a starting point of data model-derived classes and properties. The ontologist can focus on the value-added work of revising the design, adding Defined Classes, complex class restrictions, beyond the Entity/Relationship model, and hence beyond CODT reverse-engineering.



Ontology reverse-engineering from data models – Use Cases

The left-hand side shows institution-level applications of the use case. The Enterprise Ontologist reverse engineers data models to accelerate design the Enterprise Ontology and Data Scientists use the ontology to data model mapping building the

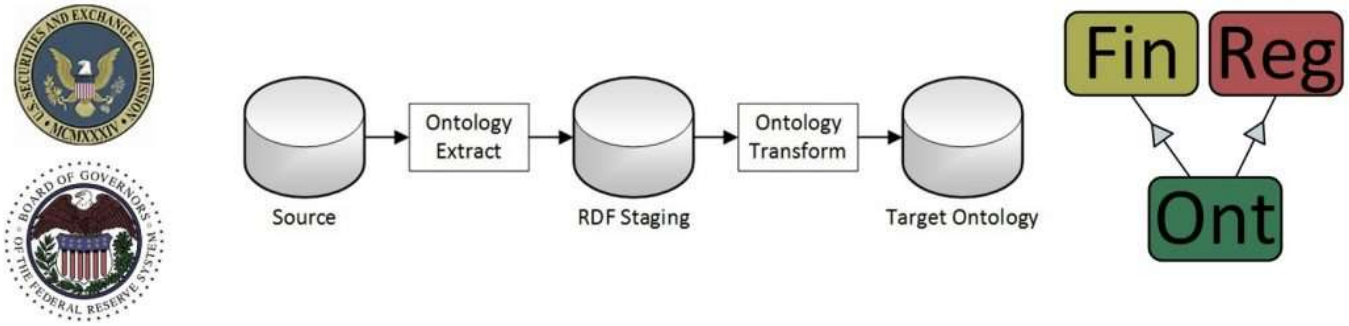
knowledge graph.

The right side use cases are project-level. Data Architects turned ontologists reverse-engineer an RDF/OWL view of departmental or application data models, and engineers use the ontology to build semantic or knowledge applications.

Reverse-engineering into Operational Ontologies

Semantic Applications infer knowledge and gain insight for operational purposes. My 2018 Enterprise Data World presentation, "The US Investment Adviser Act in FIBO," discussed the example of determining the Securities & Exchange Commission (SEC) filing requirements.

Ad-hoc use cases are Anti-Money-Laundering (AML) and Credit Applications. Typically Relational Database Systems (RDBM) handle data origination and changes, while an RDF-Store holds the same data in triples. The diagram shows a relational Source, extract into RDF-Staging already on the triple-store, and transformation and load into the Target Ontology.



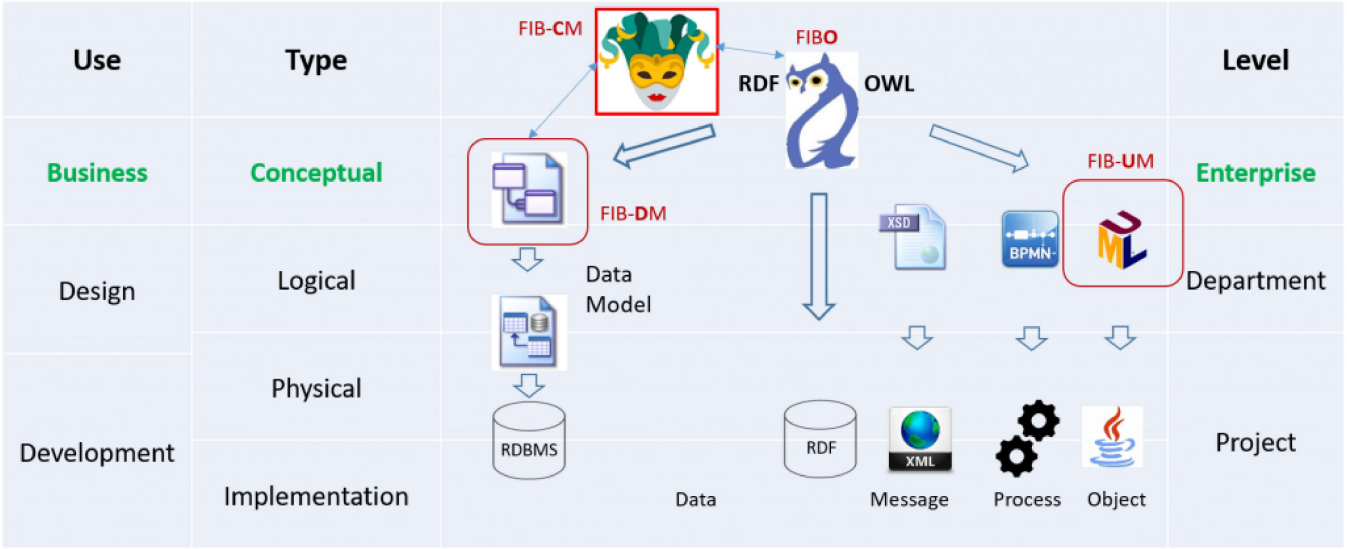
Semantic ETL

Ontology platforms and editors provide transformation and imports of relational data into an RDF-S representation. Standards like D2RQ and R2RML specify classes with instances for database tables and columns. However, RDF Staging is not an ontological design. An ontologist designs the Target Ontology.

We can accelerate the design of the Target Ontologyif we reverse-engineer the logical data model (LDM) of the RDBMS. The database table and column names are cryptic abbreviations, and the schema is likely denormalized. The LDM provides a normalized design with English names that translate to the ontology CamelCase naming convention.

Accelerating the Enterprise Ontology design

CODT's primary purpose is to transform ontologies into data models. Semantic Enterprise Information Architecture places the ontology at the apex with derived models for data, message, process, and objects.



Semantic Enterprise Information Architecture.

However, reverse-engineered data models accelerate the initial and ongoing design of the Enterprise Ontology. In the beginning, a Financial Institution adopts the Financial Industry Business Ontology, the FIBO as their reference ontology. Subsequently, Enterprise Ontologists customize and extend the design. Customization may comprise of a scope, a reduced version of the reference model. For example, a Depository Institution (FIBO for Retail Bank) may not need Derivatives and Market Data but extends the reference ontology with a module for Credit Cards. The bank reverse-engineers their Credit Card LDM to create classes and properties for the new sub-domain.

Even once the custom Enterprise Ontology is complete, the development lifecycle is similar to Enterprise Data Models (EDM):

1. Data Architects **scope** a subset of the enterprise model for a particular project.
2. Project Modelers **detail** the design adding entities, relationships, and attributes.
3. The Enterprise Architecture team **harvest**s the project model for content.useful to the whole organization.
   For an EDM the team merges in entities and other model objects.
   For an SEIA Enterprise Ontology, the team reverse-engineers the project data models and adds classes and other ontology objects.

Reverse-engineering for Knowledge Graphs

The Enterprise Data Management Council, EDMC, promotes the FIBO the ontology for an Open Knowledge Graph (OKG).

Data Architects can understand the KG is a better modern version of the Metadata Repository and Data Warehouse.

Like the metadata repository, the KG provides links and definitions to data sources. We can navigate by a business taxonomy and discover systems, database tables, and unstructured data sources. The KG is superior because its metamodel is RDF/OWL rather than a proprietary repository structure.

Like a data warehouse, the KG integrates data in conformed design. The data warehouse model corresponds to the knowledge graph ontology, and the data may be physical, moved vial ETL, or virtual with query transformation and confederated databases.

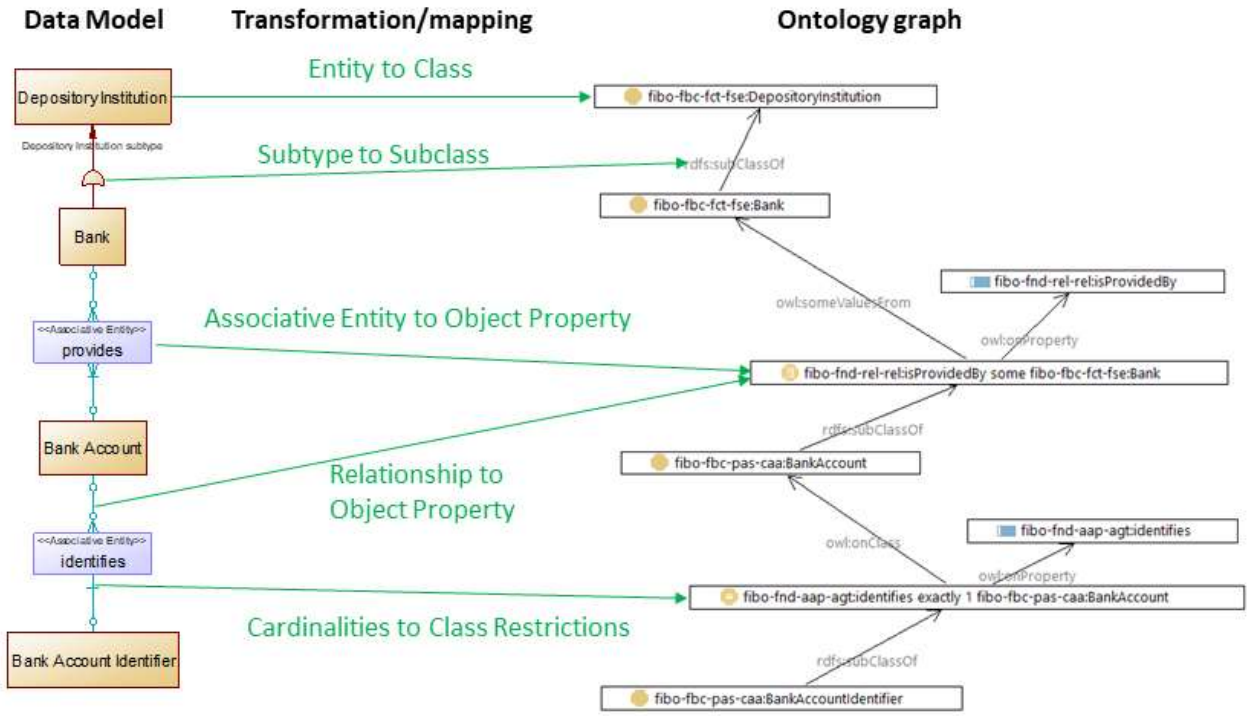
The FIBO is the industry-standard reference for both an Enterprise Ontology and a Knowledge Graph Ontology.

A difference between the two is that the Enterprise Ontology is the strategic design, whereas the KG ontology extends the design with representations of non-strategic, even legacy data sources. In other words, while we don't derive new models from non-strategic design, we still want to gain knowledge from their data.

Hence, the Enterprise Ontology use cases entirely apply to the Knowledge Graph. Besides, we may reverse-engineer non-strategic systems as we onboard their data.

Reverse mapping and transformation

The mapping diagram is a mirror image of the ontology to data model transformation. Now, we have the data model as a source with the ontology as the target.



Data Model to Ontology mapping

CODT in reverse mode transforms data model Entities into ontology classes. Data Model subtypes (PowerDesigner inheritances) generate subclass properties (rdfs:subClassOf). Depending on configuration parameters, both Associative Entities and Relationships transform into Object Properties. Cardinalities determine ontology class restrictions.

Isomorphism revisited

"An isomorphism is a structure-preserving mapping between two structures of the same type that can be preserved by an inverse mapping.https://en.wikipedia.org/wiki/Isomorphism"

The second FIB-DM article showed that ontology and data model hierarchies are the same, stating:

"There is an isomorphism between a subset Ontology Web Language (OWL) and the Entity-Relationship Model (ERM). OWL has a higher expressivity and semantic richness than the ERM. In other words, we cannot express complex axioms in a data model. However, we can preserve the structure of classes, data properties, and object properties as entities, attributes, and associations in a Conceptual Data Model (CDM). The morphism is bidirectional. That means we can preserve the structure of a CDM through ontology classes and properties. https://fib-dm.com/ontology-class-and-data-model-entity-hierarchy/"

Data Modelers are familiar with a similar isomorphic subset between the Logical and Physical Data Model (PDM). We can forward engineer an LDM into a PDM, and reverse-engineer a PDM or database schema into a logical model. Some logical model properties, like the subtype symbol, have no representation in the PDM, and of course, most table- and index properties do no reverse-engineer into properties of the entity. The test case is that a change in the source model must reflect in the target model. E.g., adding an entity to the LDM generates a new table in the PDM.

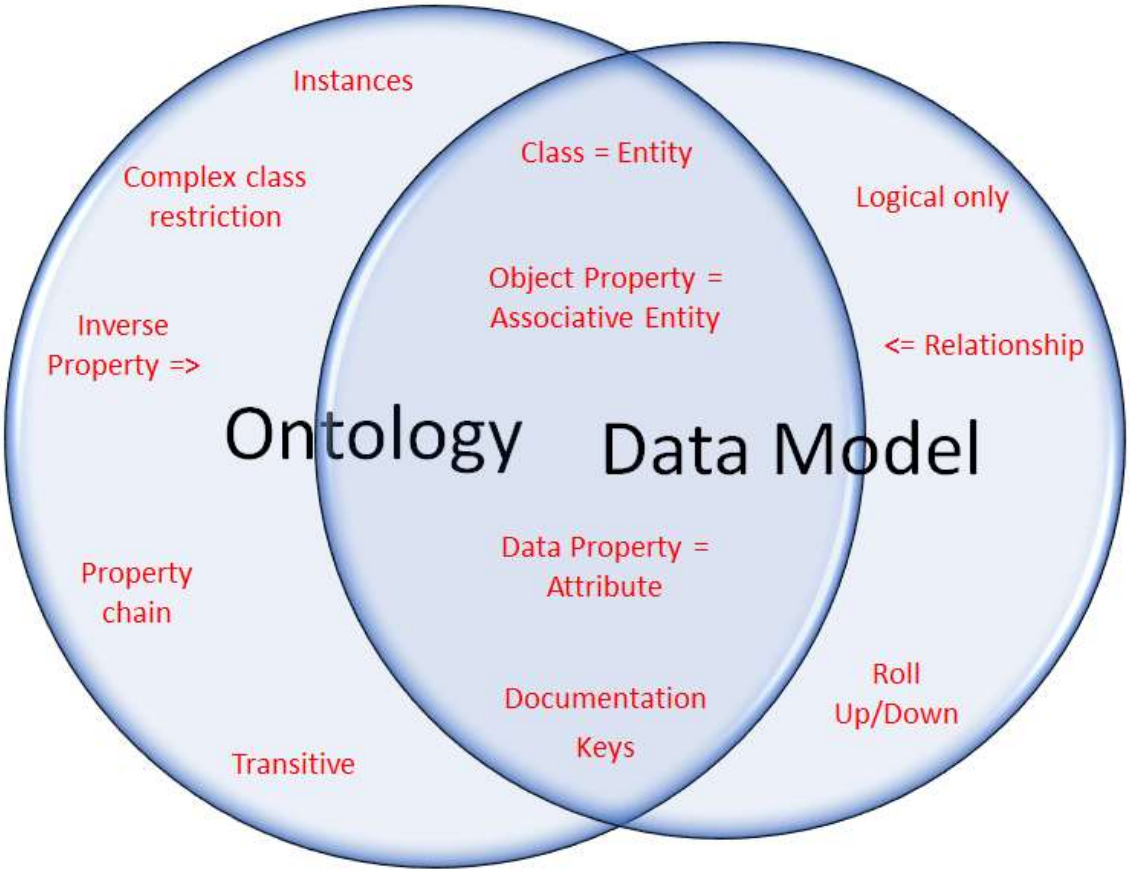
Data Modelers use the tools Compare/Merge functionality to validate change and to update the target model.

Below, for example, is an excerpt from the compare report for FIB-DM Q2 release.

```
[#] Financial Industry Business Data Model (Normative) An OWL-to-CDM transformation of the FIBO 2020/Q1
Production release.. : An OWL-to-CDM transformation of the FIBO 2020/Q2 Production release.. 2.0 (FIBO
2020/Q1) https://fib-dm.com : 2 (FIBO 2020/Q2) https://fib-dm.com Entities:
[+-] Account Number
[+-] Accounting Transaction Event
[-+] Actor
[-+] Affiliation
...
```

The listed model changes must accurately reflect the FIBO release notes.

The Venn-diagram depicts the Ontology Metamodel, the smaller Logical Data Model metamodel, and the intersection of isomorphic model elements.



The isomorphic intersection between Data and Ontology metamodel

Looking at the Data Model circle, we observe the subset, the intersection with the ontology metamodel, comprising of base and associative entities, attributes, documentation, and keys that have an inverse mapping to their ontology counterparts. Some data model properties like “Logical Only” and “Rollup/Rolldown” configure PDM generation – they are not relevant for the ontology. Looking at the ontology metamodel circle, we see that our reverse-engineered ontology is very simple – classes, object-, data properties, documentation (annotation properties), and keys. It is a substantial starting point for the ontologist, who may add semantics beyond the entity-relationship model, like instances, complex class restrictions, inverse/transitive properties, and property chains.

Before we move on to metadata-sets, it is important to note: The isomorphism itself is transitive.

To revisit the data model case: If a Conceptual Data Model (CDM) is isomorphic toward the LDM, and the LDM is isomorphic towards a PDM, then there is an isomorphism from Conceptual to Physical Model.

Bi-directional Metadata Sets

The CODT Patent drawing FIG 23 provides a more detailed view of the transformation method, Extract, Transform, and Load, in Business Process Modeling Notation. It shows the Metadata Sets for Data Model (PowerDesigner), generic Entity/Relationship, and Ontology as input and output object for the transformation steps.

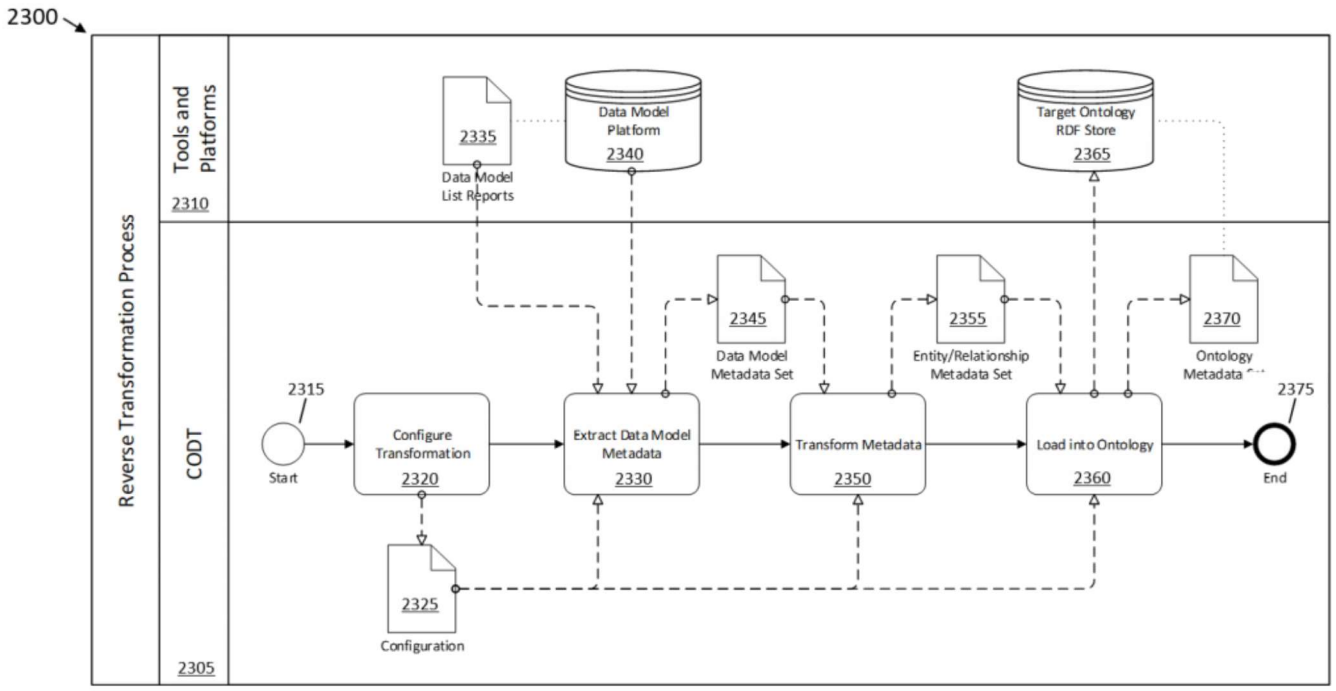


FIG. 23

CODT Patent drawing FIG 23 – Data Model to Ontology Method

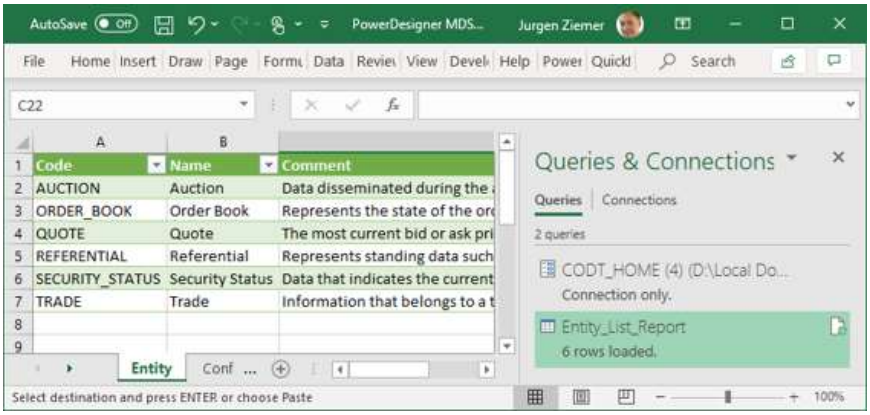
"As defined herein, a metadata set is a data set for metadata. The CODT metadata sets are coupled with computer instructions that cause the population of the metadata sets, with the non-transitory storage medium storing ontology metadata sets, entity-relationship, tool-specific metadata sets, and the code to populate them.Patent Specification"

Physical Data Modelers are familiar with metadata sets – the database system tables. Both are structures (models) of a database schema, and the modeler forward and revere-engineers between the two.

Tool-specific Metadata Sets

The data modeling tool-specific metadata sets are structural equivalent to the model. The CODT POC class showed how to import the metadata set into PowerDesigner.

Every data modeling tool can generate list reports in MS-Excel or CSV format. We can define a list report to reproduce the PowerDesigner metadata set.



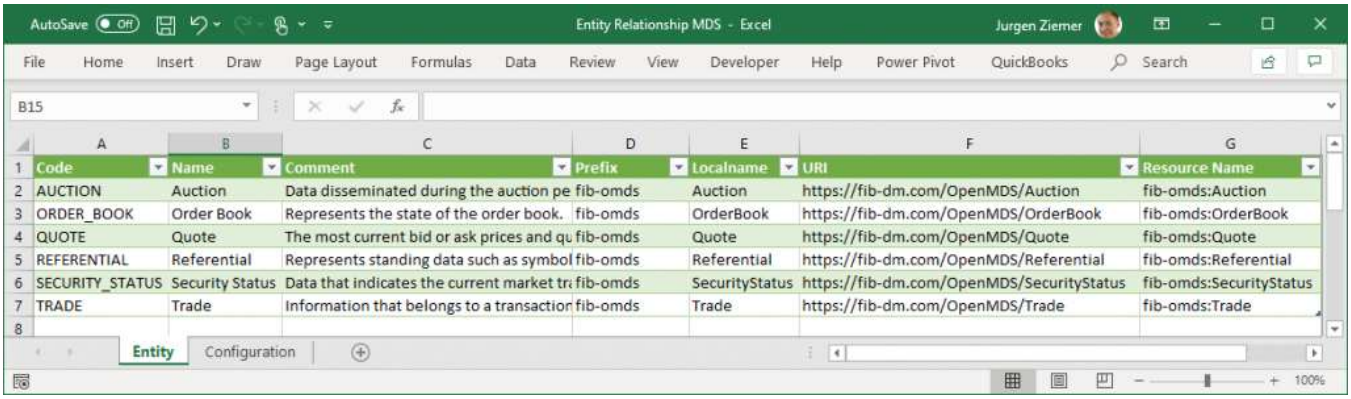
CODT Reverse-Engineered PowerDesigner Metadata Set

Alternatively, we can extract the same metadata from a model repository.

The metadata sets and the data model are isomorphic – e.g., an entity added in PowerDesigner results in an additional record in the Excel sheet, and vice versa.

Entity-Relationship Metadata Sets

The generic Entity-Relationship Metadata Set is a bridge from modeling tool-specific to ontology metadata set. First, it replaces tool-specific metamodel names with common names. For example, PowerDesigner has Inheritances, Sparx EA has Generalizations, the E/R metadata set uses the term Subtype.



	Code	Name	Comment	Prefix	Localname	URI	Resource Name
2	AUCTION	Auction	Data disseminated during the auction pe	fib-omds	Auction	https://fib-dm.com/OpenMDS/Auction	fib-omds:Auction
3	ORDER_BOOK	Order Book	Represents the state of the order book.	fib-omds	OrderBook	https://fib-dm.com/OpenMDS/OrderBook	fib-omds:OrderBook
4	QUOTE	Quote	The most current bid or ask prices and qu	fib-omds	Quote	https://fib-dm.com/OpenMDS/Quote	fib-omds:Quote
5	REFERENTIAL	Referential	Represents standing data such as symbol	fib-omds	Referential	https://fib-dm.com/OpenMDS/Referential	fib-omds:Referential
6	SECURITY_STATUS	Security Status	Data that indicates the current market tr	fib-omds	SecurityStatus	https://fib-dm.com/OpenMDS/SecurityStatus	fib-omds:SecurityStatus
7	TRADE	Trade	Information that belongs to a transaction	fib-omds	Trade	https://fib-dm.com/OpenMDS/Trade	fib-omds:Trade

CODT Reverse-Engineered Entity Relationship Metadata Set

The E/R Entity Metadata Set add columns for Prefix, Localname, URI, and Resource Name. We must configure the base URI for our target ontology, here “https://fib-dm.com/OpenMDS/”, and its abbreviation, the Prefix, “fib-omds”. The Localname is a simple “UnCamel” string function of the logical data model entity name, and the Resource Name concatenates Prefix, a colon, and the Localname.

Ontology Metadata Sets

The Ontology Metadata Sets depicted in the LDM diagram (patent drawing 14) are an isomorphic representation of an RDF/OWL metamodel, but for reverse-engineering we only populate the common subset. For example, the LDM does not have a concept of Inverse Object Properties and Equivalent Classes.

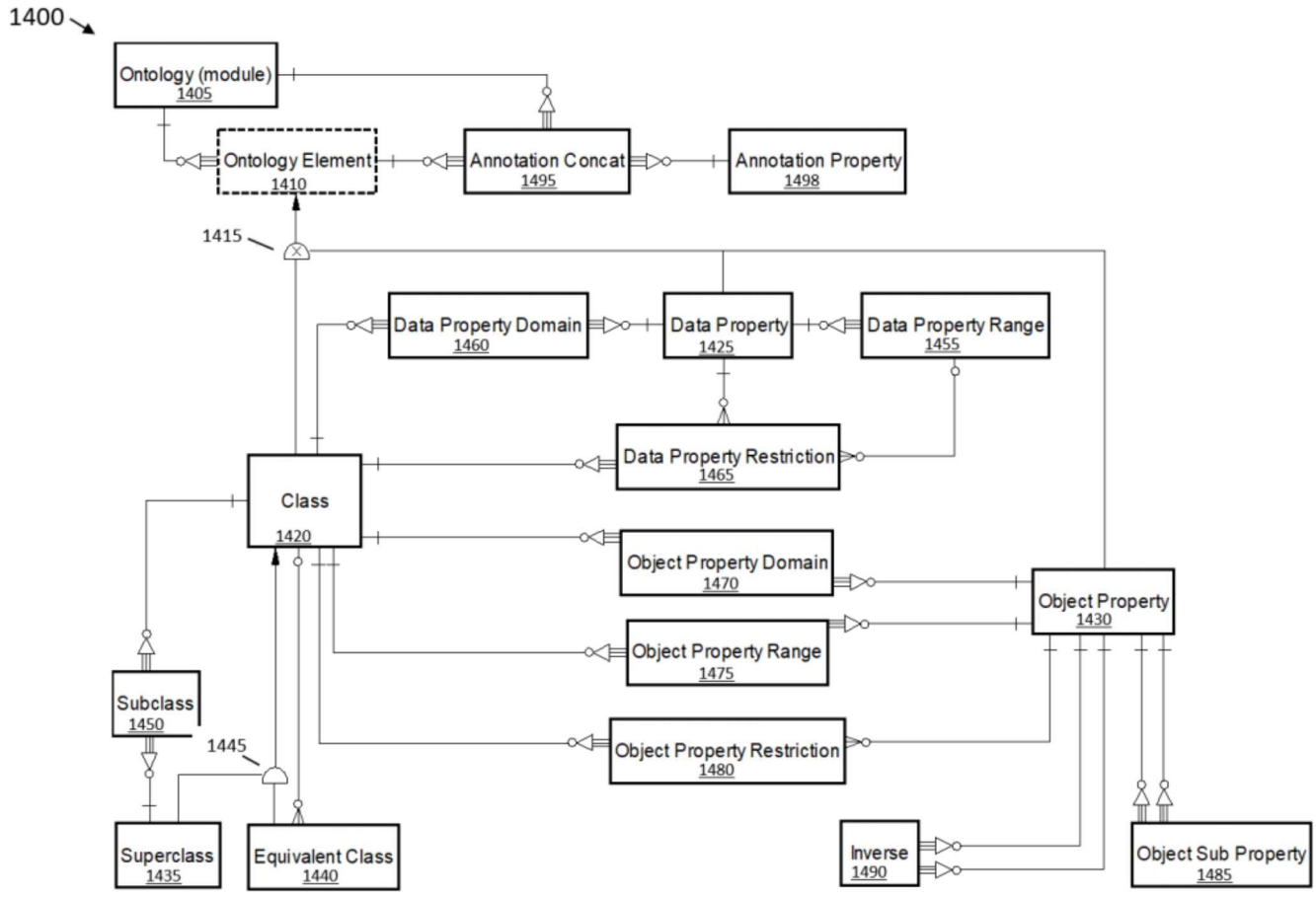
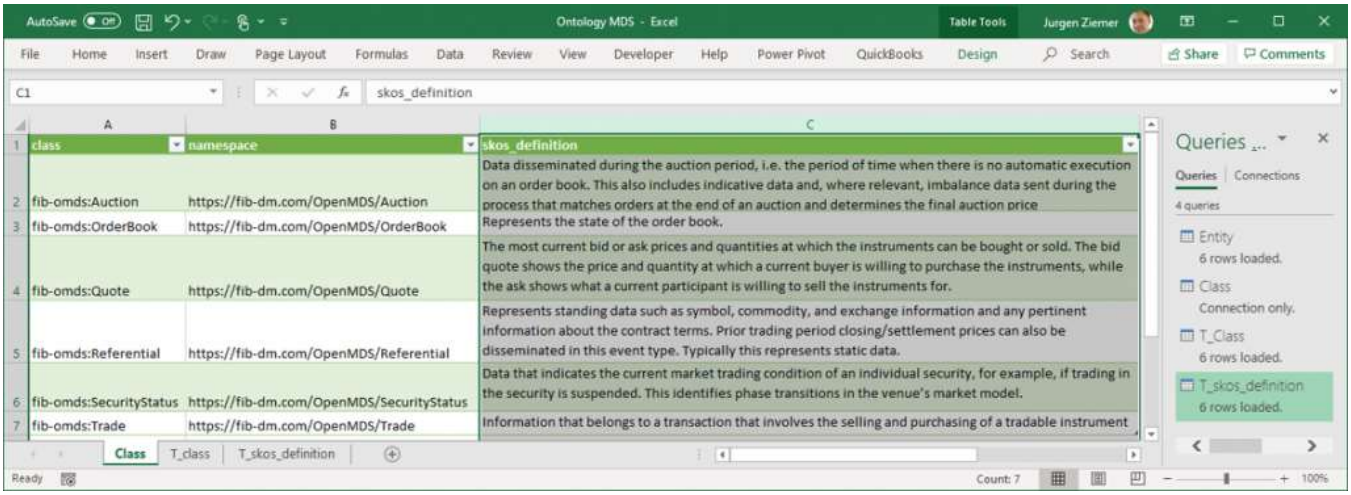


FIG. 14

CODT Patent drawing FIG 14 – Ontology Logical Data Model

The Ontologist may add these higher semantics after importing the base ontology.

The Class Metadata Set below populates from the E/R Entity Metadata Set. Again, reverse-engineered Metadata Sets may have fewer columns than those extracted from the ontology. For example, the average data model may only have a single documentation item, per default transformed into SKOS Definition.



CODT Reverse-Engineered Ontology Metadata Set

In order to Load the ontology metadata set, we must break it down into triple of subject, predicate, and object.

The T-tabs above are Power Queries sourcing the Ontology Metadata Set. The T_Class data set below is the raw data to CONSTRUCT or bulk-load triples for classes.

```
CONSTRUCT {
fib-omds:Auction rdf:type owl:Class .
}
WHERE {}
```

Or we can bulk-insert a CSV file below.

subject	predicate	object
fib-omds:Auction	rdf:type	owl:Class
fib-omds:OrderBook	rdf:type	owl:Class
fib-omds:Quote	rdf:type	owl:Class
fib-omds:SecurityStatus	rdf:type	owl:Class
fib-omds:Trade	rdf:type	owl:Class

CONSTRUCT triples for Classes

Note that the CONSTRUCT triples match the joins in the SPARQL SELECT queries, extracting metadata from the ontology.

```
# Owl Classes.rq
SELECT ?class ?qname ?namespace ?skos_definition
WHERE {
  ?class a owl:Class .
# ...
OPTIONAL {
  ?class skos:definition ?skos_definition}
}
```

Likewise, we break down triples for the Class Definition annotation property.

subject	predicate	object
fib-omds:Auction	skos:definition	Data disseminated during the auction period, i.e. the period of time when there is no automatic execution on an order book. This also includes indicative data and, where relevant, imbalance data sent during the process that matches orders at the end of an auction and determines the final auction price
fib-omds:OrderBook	skos:definition	Represents the state of the order book.
fib-omds:Quote	skos:definition	The most current bid or ask prices and quantities at which the instruments can be bought or sold. The bid quote shows the price and quantity at which a current buyer is willing to purchase the instruments, while the ask shows what a current participant is willing to sell the instruments for.
fib-omds:SecurityStatus	skos:definition	Data that indicates the current market trading condition of an individual security, for example, if trading in the security is suspended. This identifies phase transitions in the venue's market model.

subject	predicate	object
fib-omds:Trade	skos:definition	Information that belongs to a transaction that involves the selling and purchasing of a tradable instrument

CONSTRUCT triples for Class Definitions

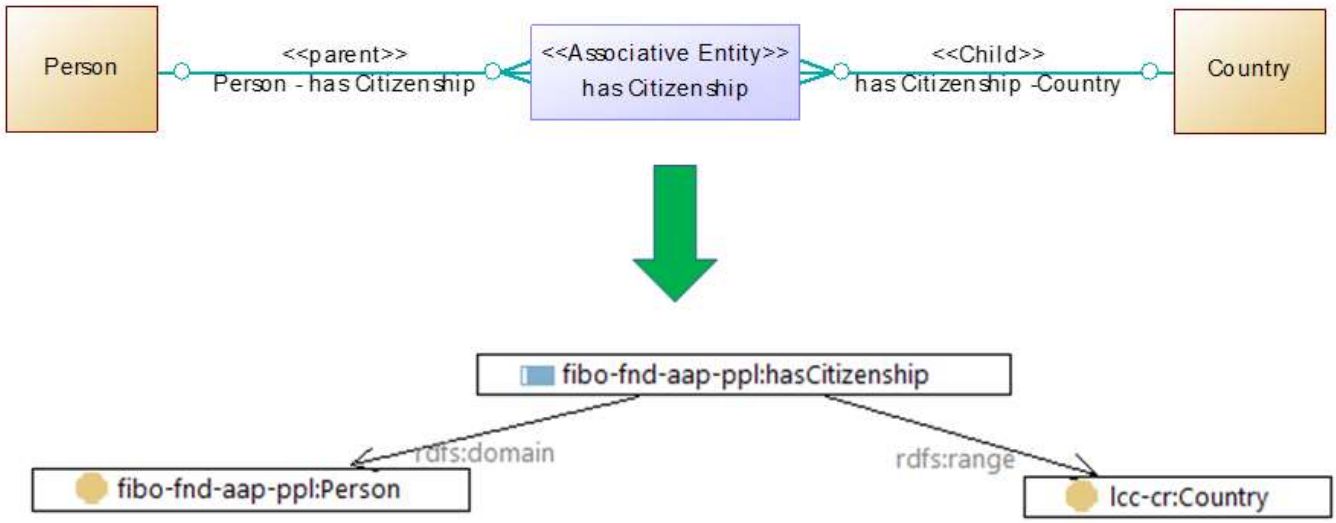
Just like PowerDesigner import and list report proved the existence isomorphic transformation, so does SPARQL: If we extract metadata with SELECT data, we can also load the same metadata using CONSTRUCT.
Unfortunately, ontology editors lack the powerful comparison report functionality of the data modeling tools.

Associative Entities are Object Properties

My third article made a detailed case that ontology Object Properties are Associative Entities – not Relationships. How about the reverse transformation?

The conventional wisdom derives object properties from data model relationships or database foreign keys, and **all** Entities/Tables transform into ontology classes. Many data models don't have named relationships, and modeling tool generated names, like "Relationship_1," don't provide meaningful Object Property Localnames.

Per default configuration, CODT transforms object properties into associative entities. For the isomorphism to hold, CODT must reverse-engineer these associative entities back into object properties.



CODT Associative Entity to Object Property Transformation

The easiest configuration examines the entity stereotype. CODT generated entities have a stereotype <<Associative Entity>>, and for those, we populate the Object Property metadata set and derive class restrictions from the relationships of the Associative Entity. Data Modelers extending a CODT derived data model, like FIB-DM, should set the stereotype for new Associative Entities.

Modelers should configure the Stereotype on Associative Entities for Data Models; they want to onboard onto the Knowledge Graph or Enterprise Ontology.

Can CODT detect Associative Entities? The standard Associative Entity resolves a many-to-many data model relationship. In this case, the associative entity participates as a child entity in exactly two identifying relationships.

However, some Associative Entities may have more than two related base entities, for example, the classic Project, Resource, Role pattern.

Criteria may stipulate that Associative Entities don't have attributes. If they do, then we need a class to hold them as data properties.

Every modeling standard and every data model is different. Data Architects and Ontologist should analyze the source model to determine the CODT optimal configuration. It fine to experiment and test different ways to reverse engineer the data model.

Conclusion

Logical and Conceptual Data Models are a better source to reverse-engineer ontologies than database schemas. We examined the use cases for operational-, enterprise-ontologies, and knowledge graphs. The isomorphism between the common metamodel subset of ontology and data model extends to their metadata set representations. Metadata Sets are bi-directional, and hence CODT can forward and reverse engineer. Associative Entities should transform into object properties. However, it remains a challenge to identify them.

The CODT transformation is the best way to transform data models into ontologies.

References

The canonical version of this article: https://fib-dm.com/data-model-reverse-engineered-into-ontology/

Join the discussion on LinkedIn: https://www.linkedin.com/pulse/data-model-transformed-ontology-jurgen-ziemer/

FIB-DM articles

1. Finance Domain ontology transformed into an Enterprise Data Model
2. Ontology Class- and Data Model Entity-hierarchy, are they the same?
3. Ontology Object Properties are Data Model Associative Entities – not Relationships

PowerPoints and videos

Semantics for extra-large banks, the CODT POC class/tutorial

Bank Call Report in FIBO, 2017 Enterprise Data World Conference