



Configurable Ontology to Data model Transformation

[Home](#) [Technology](#) ∨ [Patent](#) ∨ [About](#)

Data model reverse-engineered into ontology

Abstract


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

Background

Model transformations migrate a model into another type of model. Data Architects use the term forward-engineering to transform a higher-level logical model into a physical model and subsequently into a database schema. 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 objects. Hence, I use the term “reverse-engineered” to describe data models transformed into ontologies. Note that “reverse” doesn’t mean 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, no tool has been developed to transform an ontology into a functional data model. Hence, 3,000 users downloaded the Open-Source version of the Financial Industry Business Data Model (FIB-DM), which was derived using CODT ([US patent #12038939](#)) from the industry-standard domain ontology.

Reverse transformation research and tooling solely address databases—no tool or research exists to reverse-engineer an ontology from PowerDesigner, ERWin, or other data modeling tools.

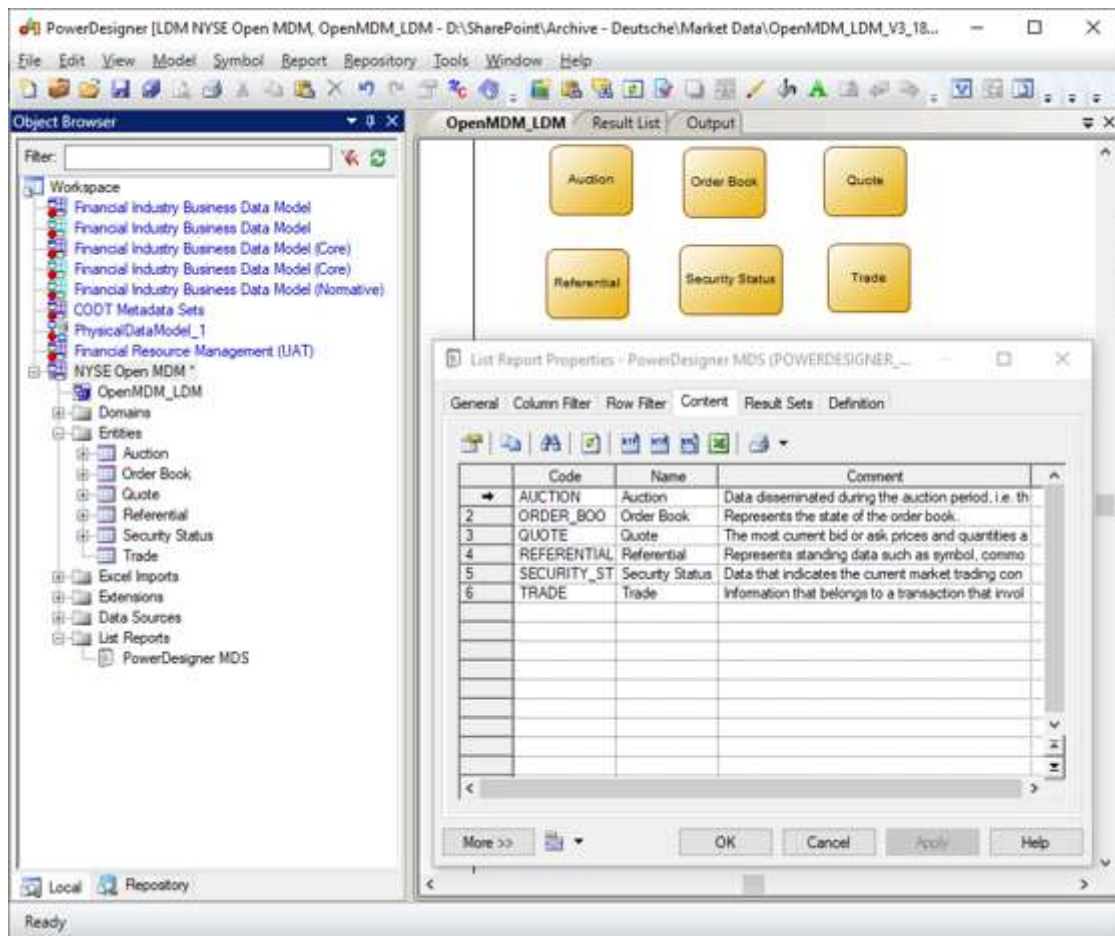
The first FIB-DM article, "[Finance Ontology Transformed into an Enterprise Data Model](#) , and the CODT Patent state that the transformation technology is bidirectional; however, they do not provide the rationale and details. The second article demonstrated that the "Ontology Class—and [Data Model Entity-hierarchy](#)" is equivalent, and the third states that "Object Properties are Associative Entities."

This final installment examines the reverse transformation.

Overview – the Trading Model example

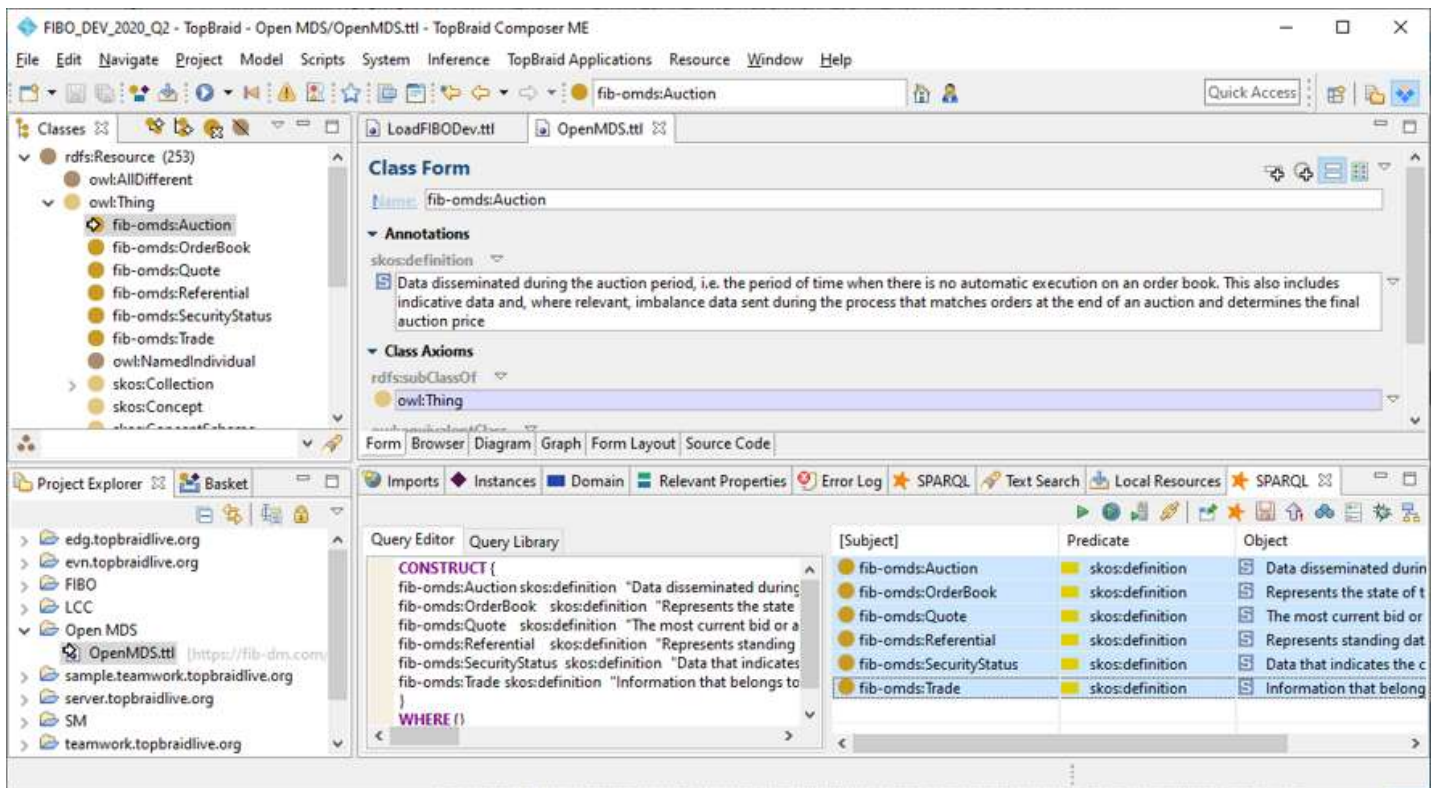
In the CODT POC class, "[Semantics for extra-large Banks](#)," participants transform first the FIBO and then their proprietary ontology extensions into data models. The course briefly covers the reverse transformation.

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



CODT Reverse example – Open MDM data model

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, TopBraid Composer.



CODT Reverse-Engineered ontology (TopBraid Composer)

Logical data model names are converted into ontology CamelCase notation, and “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 reverses the ontology-to-data model transformation. The symbol on the 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 it into the ontology editor or RDF database – **ETL**:

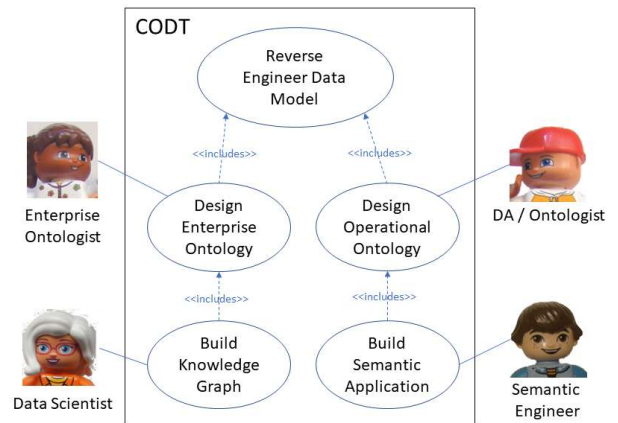
- Extract: The Data Architect generates list reports that match the Data Modeling tool-specific MDS. Power Query populates the Metadata sets, performing basic data

cleansing.

- Transform: The Entity-Relationship Metadata Sets are populated from tool-specific metadata sets.
- Load: The Ontology MDS populates from the Entity-Relationship MDS. Power Queries and formulas break the dataset down into triples. We load triples into the ontology platform using SPARQL CONSTRUCT or bulk insert.

Use Cases

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



Ontology reverse-engineering Use Case

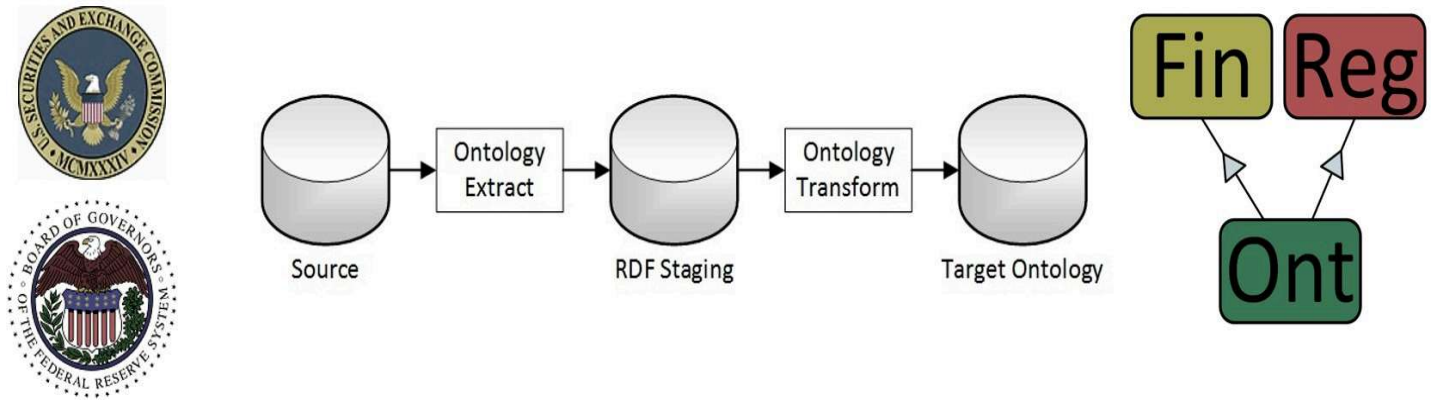
The left-hand side shows institution-level applications of the use case. The Enterprise Ontologist reverse engineers data models to accelerate the design of the Enterprise Ontology, and Data Scientists use the ontology-to-data model mapping to build the knowledge graph.

The right-side use cases are project-level. Data architects, who have become ontologists, reverse-engineer an RDF/OWL view of departmental or application data models, and engineers use the ontology to build semantic or knowledge-based 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. Relational Database Management Systems (RDBMS) typically handle data origination and changes, while RDF stores share the same data in triples. The diagram shows a relational Source, extracting it into RDF staging already on the triple store, transforming it, and loading it into the Target Ontology.



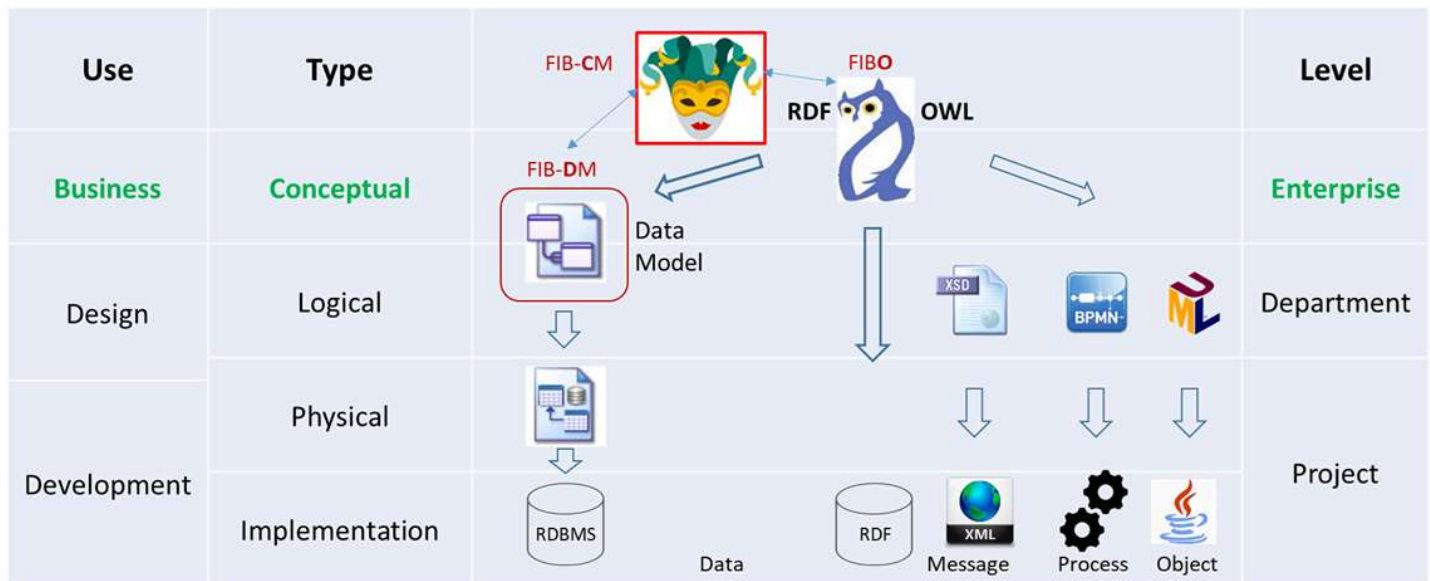
Semantic ETL

Ontology platforms and editors transform and import 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 Ontology by reverse-engineering 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, messages, processes, and objects.



Semantic Enterprise Information Architecture (SEIA)

However, reverse-engineered data models accelerate the initial and ongoing design of the Enterprise Ontology. Initially, a financial institution adopts the Financial Industry Business Ontology (FIBO) as its reference ontology for the financial industry. Subsequently, Enterprise Ontologists customize and extend the design. Customization may comprise 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 extend the reference ontology with a module for Credit Cards. The bank reverse-engineers its 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 **harvests** the project model for valuable content to the organization.

For an EDM, the team merges entities and other model objects.

The team reverse-engineers the project data models for a SEIA Enterprise Ontology 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 that the OKG is a better modern version of the Metadata Repository and Data Warehouse. Like the

metadata repository, the OKG 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 a conformed design. The data warehouse model corresponds to the knowledge graph ontology, and the data may be physical, moved via ETL, or virtual with query transformation and confederated databases.

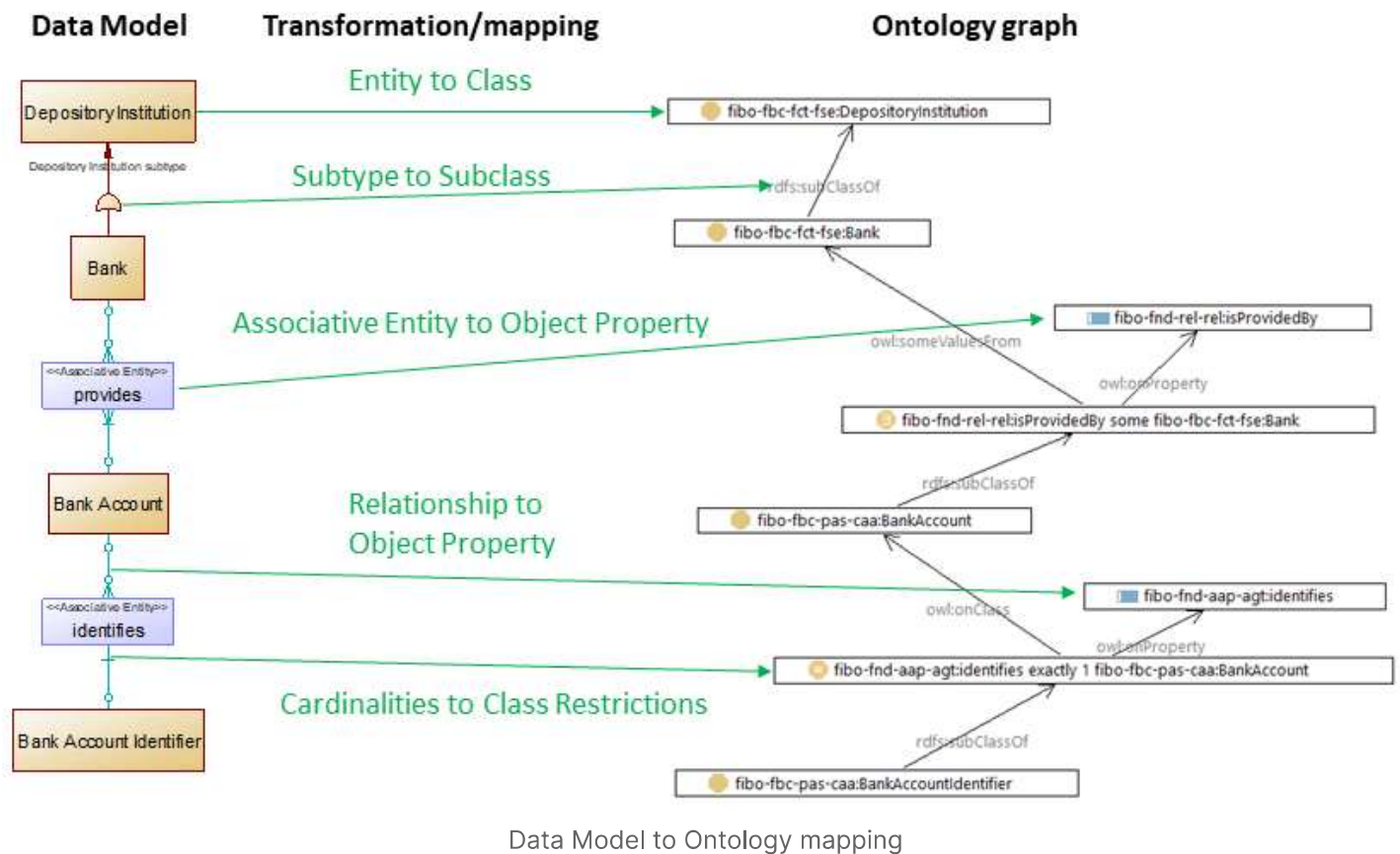
The FIBO is the industry-standard reference for Enterprise and Knowledge Graph Ontology.

A key difference between the two is that the Enterprise Ontology serves as 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 and the ontology as the target.



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 Models (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. Of course, most table and index properties do not reverse-engineer into properties of the entity. The test case is that a change in the source model is reflected 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 changes and update the target model.

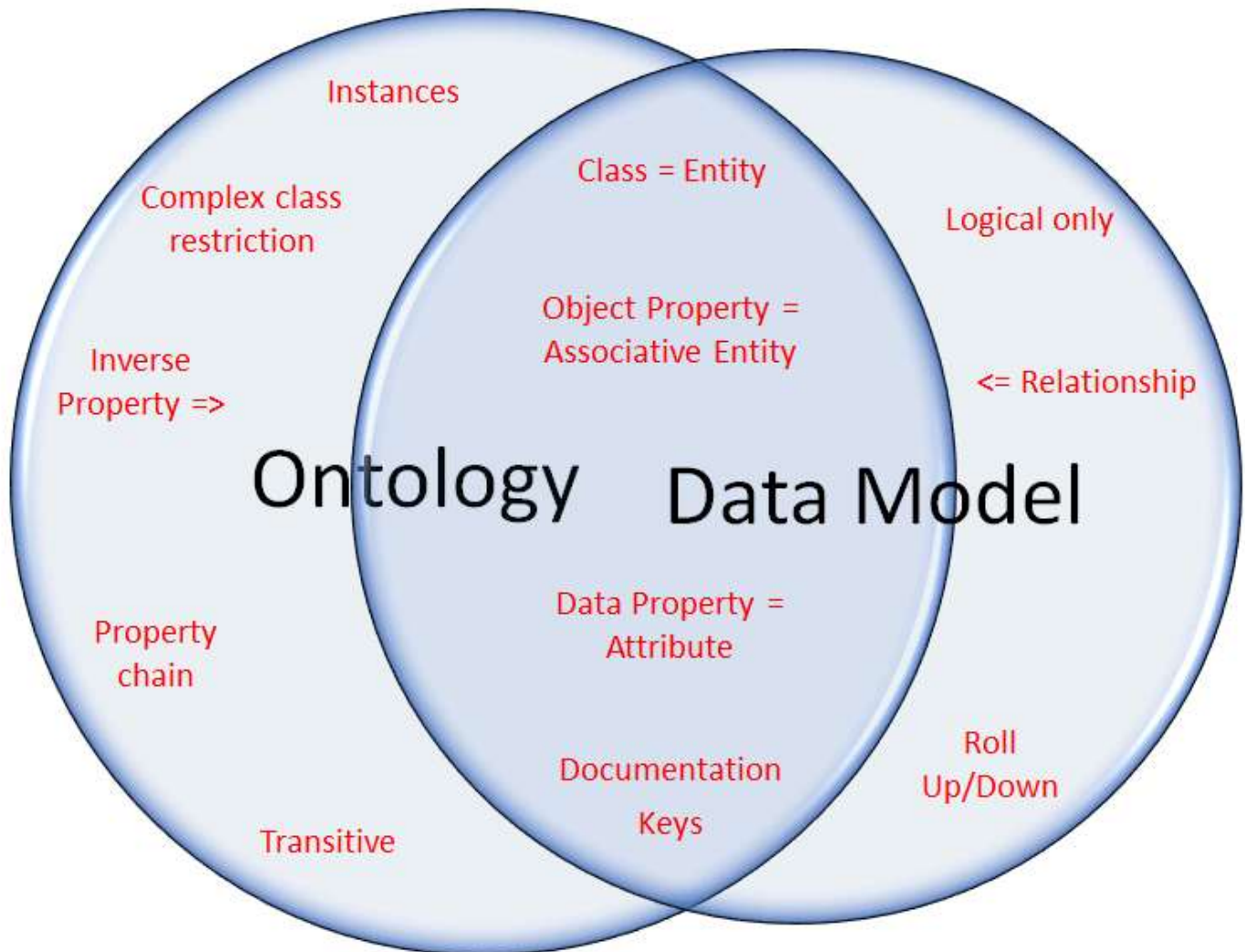
Below, for example, is an excerpt from the comparison report for the FIB-DM Q2 release.

```
[#] 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 Ontology and Data Model metamodel

Examining the Data Model circle, we observe a subset that intersects with the ontology metamodel, comprising base and associative entities, attributes, documentation, and keys with inverse mappings to their ontology counterparts. Some data model properties like “Logical Only” and “Rollup/Rolldown” configure PDM generation, irrelevant to the ontology. Examining the ontology metamodel circle, we observe that our reverse-

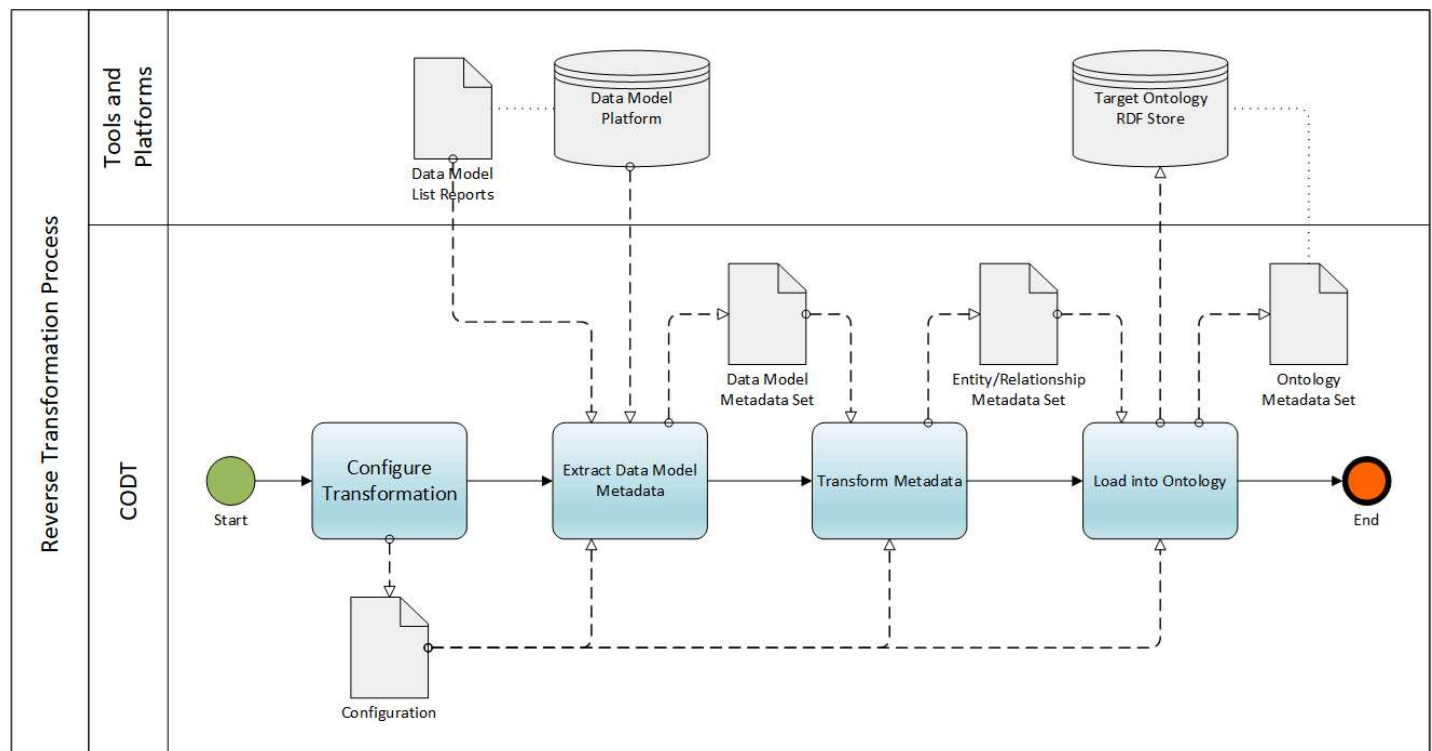
engineered ontology is relatively simple, comprising classes, object properties, data properties, documentation (annotation properties), and keys. It serves as a substantial starting point for the ontologist, who can then add semantics beyond the entity-relationship model, such as instances, complex class restrictions, inverse and transitive properties, and property chains.

Before we move on to metadata-sets, it is essential to note that the isomorphism itself is transitive.

To revisit the data model case: If a Conceptual Data Model (CDM) is isomorphic to the LDM, and the LDM is isomorphic to a PDM, then there is an isomorphism from the Conceptual to the 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 the Data Model (PowerDesigner), generic Entity/Relationship, and Ontology as input and output objects for the transformation steps.



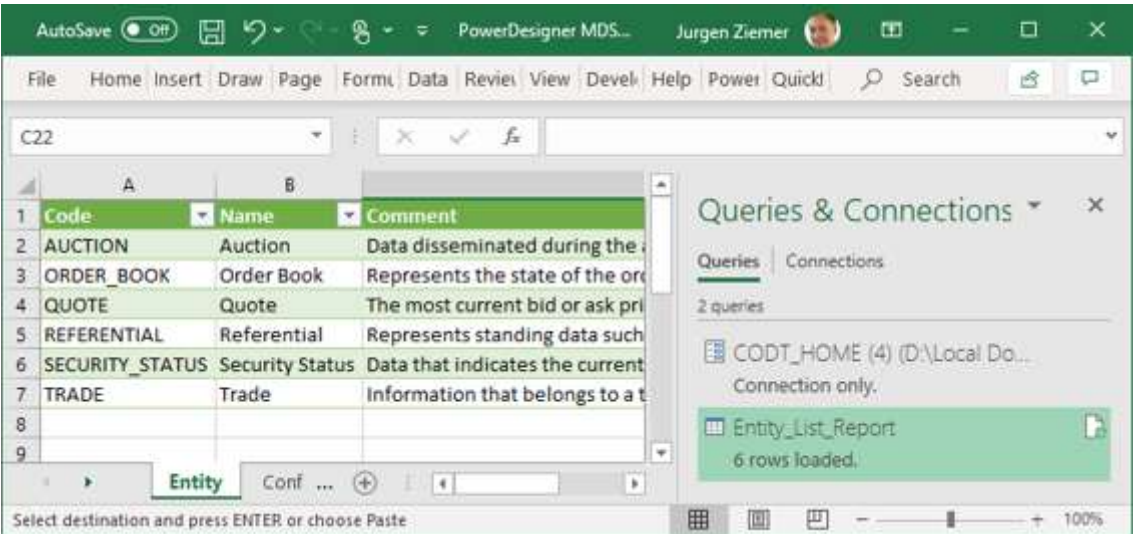
CODT Patent FIG 23 Reverse Transformation BPMN diagram (color).

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, which include the database system tables. Both are structures (models) of a database schema, and the modeler forwards and reverse-engineers are between the two.

Tool-specific



Code	Name	Comment
AUCTION	Auction	Data disseminated during the
ORDER_BOOK	Order Book	Represents the state of the or
QUOTE	Quote	The most current bid or ask pri
REFERENTIAL	Referential	Represents standing data such
SECURITY_STATUS	Security Status	Data that indicates the current
TRADE	Trade	Information that belongs to a t

CODT Reverse-Engineered PowerDesigner Metadata Set

Metadata Sets

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

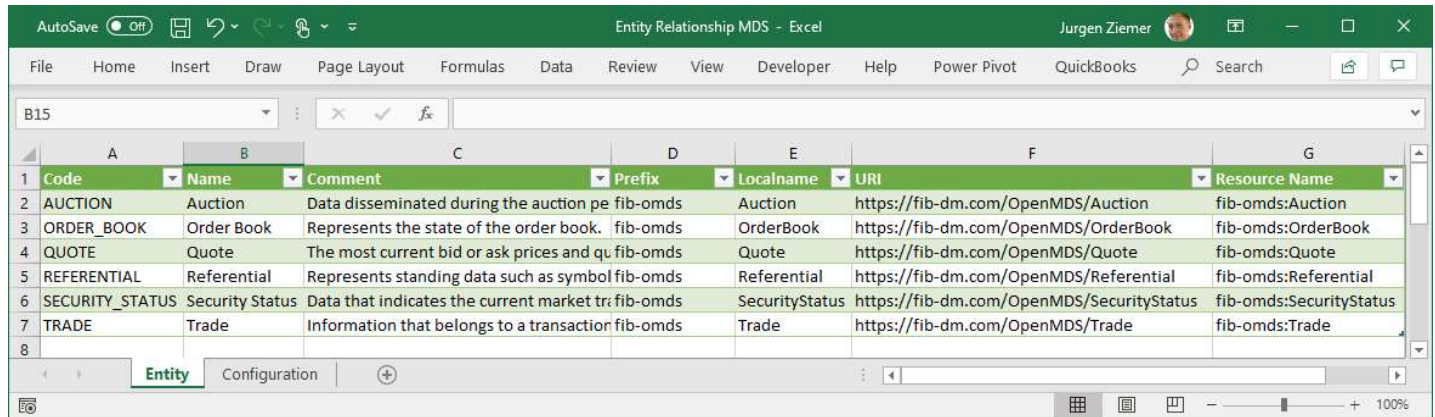
Every data modeling tool can generate list reports in MS Excel or CSV format. We can define a list report to reproduce the PowerDesigner metadata set.

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 serves as a bridge between modeling-tool-specific metadata and ontology metadata sets. First, it replaces tool-specific metamodel names with common names. For example, PowerDesigner has Inheritance, Sparx EA has Generalization, and the E/R metadata set uses Subtype.



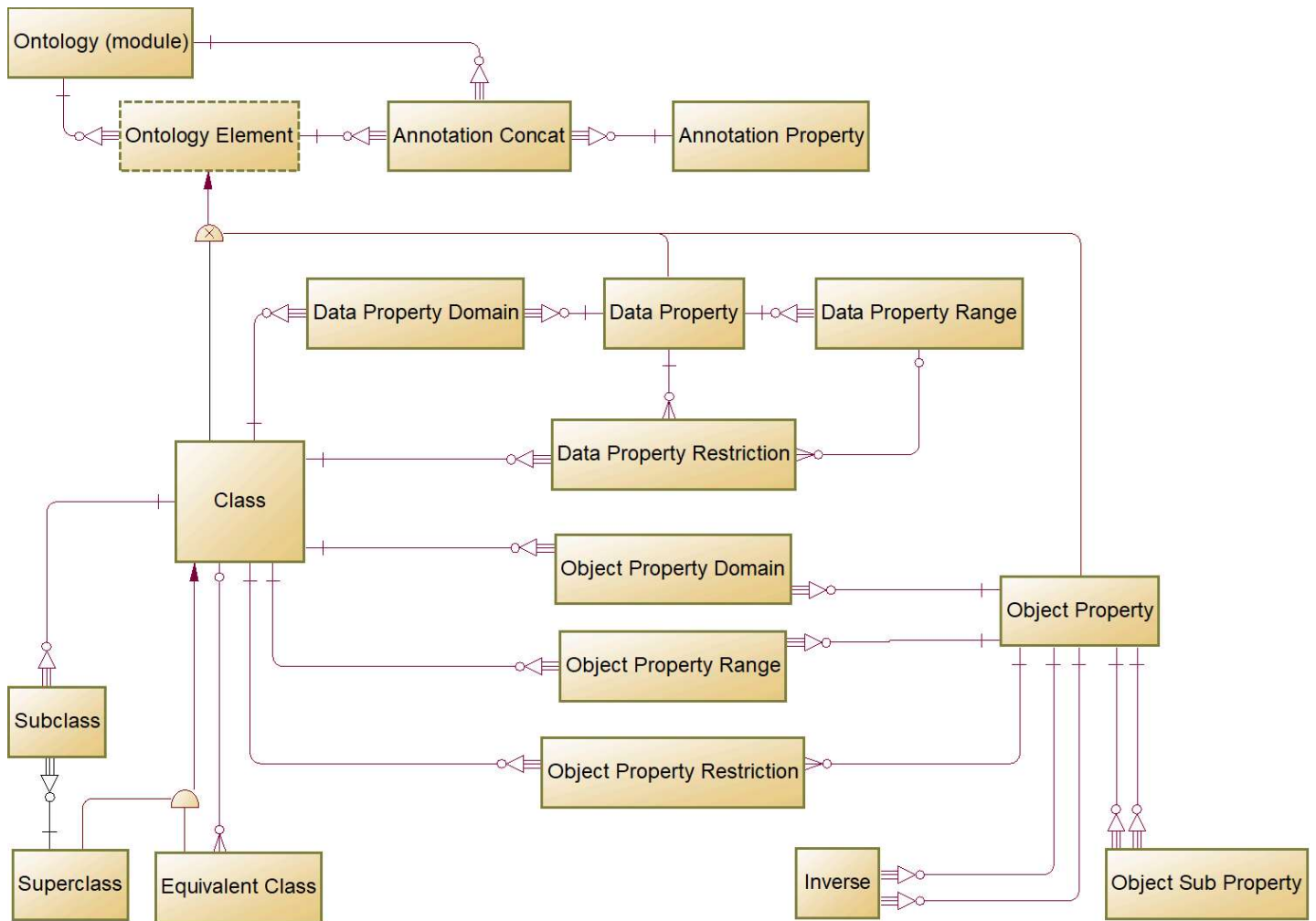
Code	Name	Comment	Prefix	Localname	URI	Resource Name
AUCTION	Auction	Data disseminated during the auction pe	fib-omds	Auction	https://fib-dm.com/OpenMDS/Auction	fib-omds:Auction
ORDER_BOOK	Order Book	Represents the state of the order book.	fib-omds	OrderBook	https://fib-dm.com/OpenMDS/OrderBook	fib-omds:OrderBook
QUOTE	Quote	The most current bid or ask prices and qu	fib-omds	Quote	https://fib-dm.com/OpenMDS/Quote	fib-omds:Quote
REFERENTIAL	Referential	Represents standing data such as symbol	fib-omds	Referential	https://fib-dm.com/OpenMDS/Referential	fib-omds:Referential
SECURITY_STATUS	Security Status	Data that indicates the current market tr	fib-omds	SecurityStatus	https://fib-dm.com/OpenMDS/SecurityStatus	fib-omds:SecurityStatus
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 adds Prefix, Localname, URI, and Resource Name columns. We must configure the base URI for our target ontology, which is “https://fib-dm.com/OpenMDS/” and its abbreviation, the Prefix “fib-mods.” The Localname is a simple “UnCamel” string function applied to the logical data model entity name, and the Resource Name is concatenated from the 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; however, we only populate the common subset for reverse engineering purposes. For example, the LDM does not have a concept of Inverse Object Properties and Equivalent Classes.



CODT Patent FIG 14 Ontology Metadata Sets Logical Data Model diagram (color)

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 a SKOS Definition.

class	namespace	skos_definition
fib-omds:Auction	https://fib-dm.com/OpenMDS/Auction	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	https://fib-dm.com/OpenMDS/OrderBook	Represents the state of the order book.
fib-omds:Quote	https://fib-dm.com/OpenMDS/Quote	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:Referential	https://fib-dm.com/OpenMDS/Referential	Represents standing data such as symbol, commodity, and exchange information and any pertinent information about the contract terms. Prior trading period closing/settlement prices can also be disseminated in this event type. Typically this represents static data.
fib-omds:SecurityStatus	https://fib-dm.com/OpenMDS/SecurityStatus	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.
fib-omds:Trade	https://fib-dm.com/OpenMDS/Trade	Information that belongs to a transaction that involves the selling and purchasing of a tradable instrument.

CODT Reverse-Engineered Ontology Metadata Set

To load the ontology metadata set, we must break it down into triples: 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.

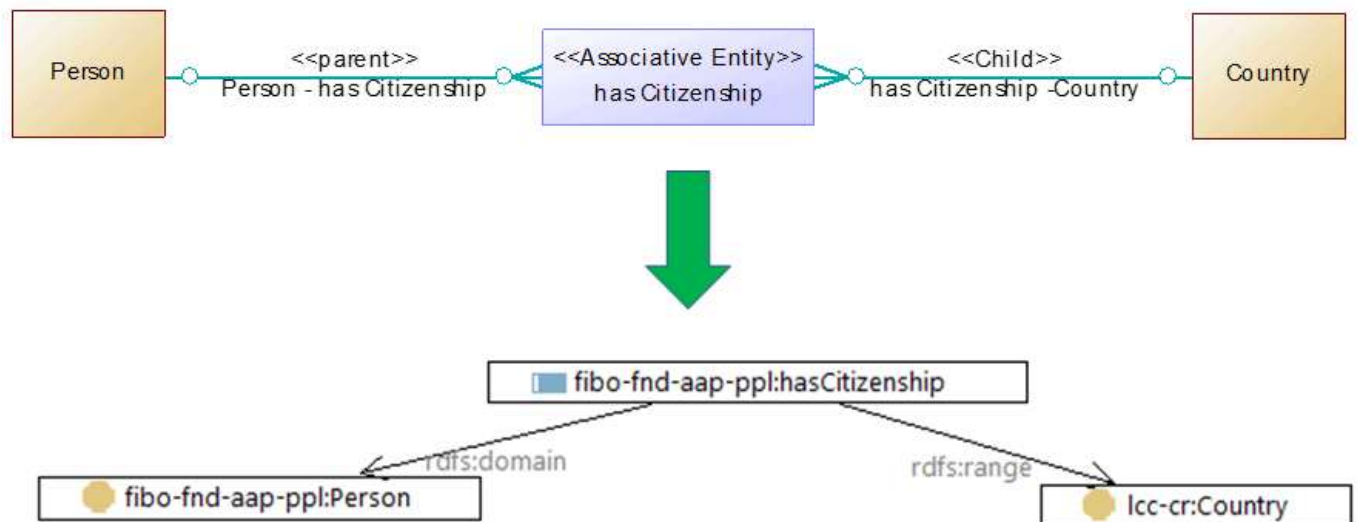
subject	predicate	object
fib-omds:Auction	skos:definition	Data disseminated during the auction period, i.e. the period of time when there is no automatic execution
		on an order book. This also includes indicative data and, where relevant, imbalance data sent during the process that matches orders at the end of an auction and determines the final auction price
fib-omds:OrderBook	skos:definition	Represents the state of the order book.
		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
fib-omds:Quote	skos:definition	The question shows what a current participant is willing to sell the instruments for.

subject	predicate	object
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.
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

Like PowerDesigner import and list report proved the existence of 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.



CODT Associative Entity to Object Property Transformation

Associative Entities are Object Properties

My third article detailed 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. However,

many data models lack named relationships, and modeling tool-generated names, such as "Relationship_1," do not provide meaningful Object Property Localnames.

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

The most straightforward configuration examines the entity stereotype. CODT-generated entities have a stereotype <<Associative Entity>>; 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, as they aim to onboard to 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 is 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, and Role pattern.

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

Every modeling standard and every data model is different. Data Architects and Ontologists should analyze the source model to determine the CODT optimal configuration. You should experiment and test various methods to reverse-engineer the data model.

Conclusion

Logical and Conceptual Data Models are better sources for reverse-engineering ontologies than database schemas. We examined the use cases for operational, enterprise, and knowledge graphs. The isomorphism between the common metamodel subset of the ontology and the data model extends to their metadata set representations. Metadata Sets are bidirectional; 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 PDF version of this article.

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.](#)
4. [CODT Patent, introduction, specification, claims, and questions](#)

PowerPoints and videos

[Semantics for extra-large banks](#), the CODT POC class/tutorial

© 1999-2025 Jayzed Data Models Inc. [Terms of Use](#)