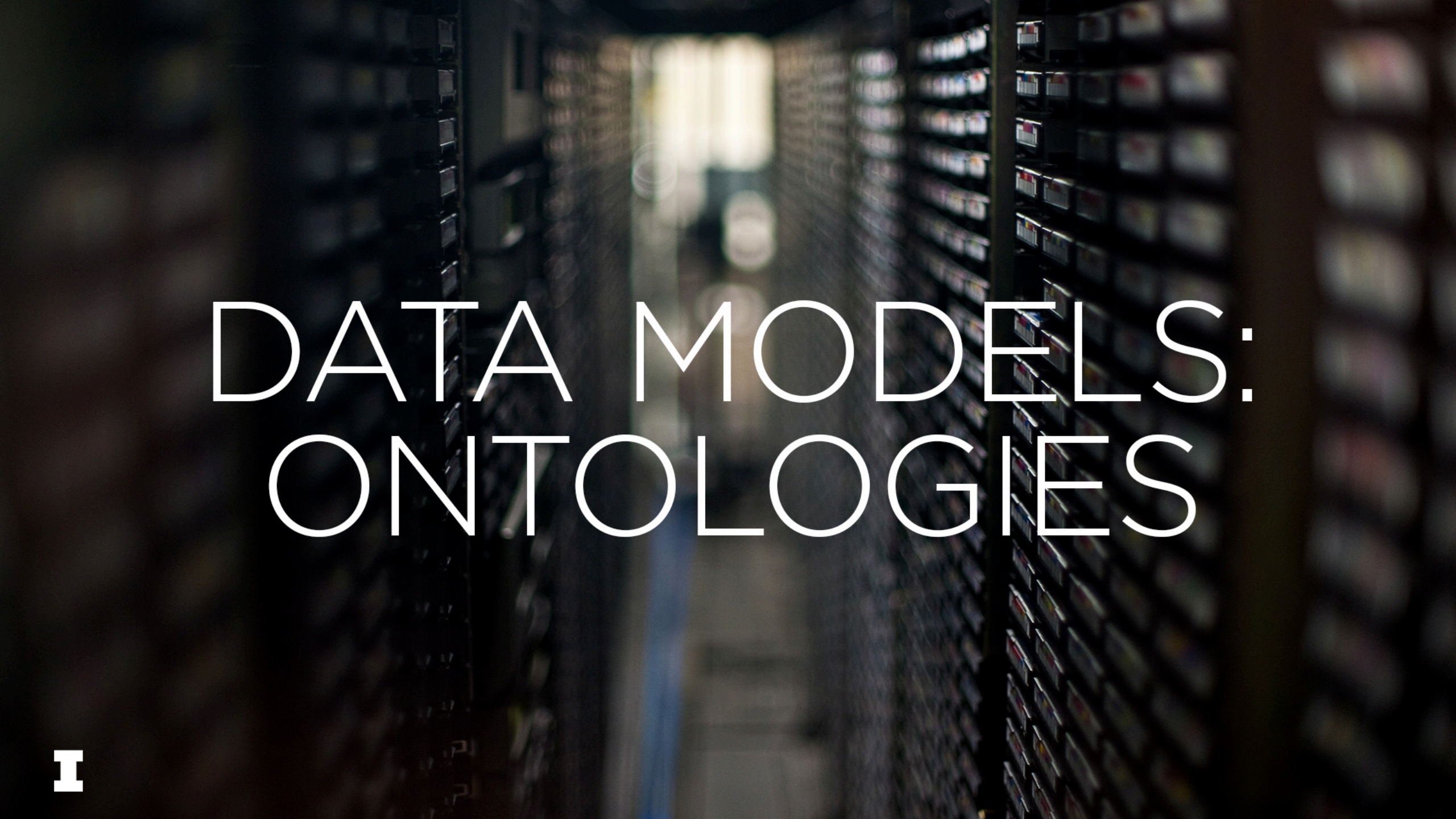

DATA MODELS:
ONTOLOGIES

The Problem: connecting data to *information*

Tables and trees abstract away from storage and processing
and let us focus on the data itself

But just as there are many ways to store the same data,
there are also many ways to use tables and trees to *express the same information*

This results problems similar but different than we saw with tables and trees

Setting up the problem

Imagine that you you need to manage some data.

You might use XML trees, or you might use relations, or you might use RDF triples.

Or you might start with XML trees, then in a few years use relations, and then later use RDF triples.

But all through these changes your domain of interest remains pretty much the same:
the same kinds of things, the same kinds of properties, the same kinds of relationships.

Moreover schemas for a domain are frequently revised even within a single type of logical data model

But again, even as schema details and structures are updated, the domain remains the same

So: different data model types and different schemas within a data model type

But through all this just one domain of kinds of objects, relationships, properties

The Problem

So, different data models types and different schemas within a data model

But just one domain of kinds of objects, relationships, properties

So obviously our understanding of that domain is of course extremely important.

In a single domain that understanding:

Guides initial schema development

Is constant across schema variations and revisions (within a model type)

Is constant across schemas from different model types

But where is this understanding of the domain articulated? And how is it connected to these logical schemas?

Too often only in the memory of programmers and other staff.

Even when prose documentation of the domain features exists that documentation rarely provides the formal precision and completeness needed to be useful; nor the mathematical constructs needed for computer processing.

How to solve this problem

To address this problem

we need an independent neutral way to describe the domain of interest.

This description cannot be tied to any particular logical model
Or to implementation variations of different specific schemas within a logical model.

It must in fact abstract away from logical models and schema variations
(just as those logical models themselves abstracted away from storage methods).

This description should be similarly precise, complete, and mathematical.

And it must be possible to map these descriptions to schemas at the logical level.

You look skeptical

It is natural to feel that this is going too far

After all, when we look at a schema, we feel that in fact it is describing the domain

But this is because of all the information we bring to bear in interpreting that schema

And since we do this so naturally and routinely we tend to forget just how technical and thin the formal theory of the relation model is

Let’s look at the issue from a different perspective:

What we think we see . . .
vs what is really there

Take a look at this relation; what do you see? What is it saying to you?

WorkID	Author	Title	FirstPublished
W58425	M42425	Moby Dick	1851
W85246	H24246	The Scarlet Letter	1850
W55427	H24246	Fanshawe	1828

What we intend to do here is record here, and what you see, *information*, propositions we believe to be true.

e.g., (1) *W58425, a work, is titled "Moby Dick" and was published in 1851.*

But given the formalities of the relational model the relation's explicit semantics produces, at best, this proposition:

(2) *There is a tuple where the primary key **WorkID** has the value'W5825' and the **title** attribute has as its value the string "Moby Dick."*

The Missing Link: *Information*

WorkID	Author	Title	First Published
W58425	M42425	Moby Dick	1851
W85246	H24246	The Scarlet Letter	1850
W55427	H24246	Fanshawe	1828

Yes, *you* can indeed infer from that relation that

(1) *W58425, a work, is titled "Moby Dick" and was published in 1851.*

But that is an *inference* humans make.

We can make such inferences from the fact that in a particular tuple certain attributes have certain values.

But the relational model does not support this inference.

And we make the inference using background information: English meanings of column headings and values*, knowledge of the domain, common sense, conversations with the DBA, bits of ancient documentation, etc.

This makes the *meaning*, the *semantics* of the relation unreliable and inaccessible to computer processing.

*Note that in the formal relational model the column headings are not attributes in the ordinary sense (properties), they are domain names, strings mapped to sets of allowed values. And there is no sense, in pure relational theory, in which the value of an attribute/value pair is *asserted of*, *said about*, or *describes* the thing identified by the corresponding primary key. That is all added interpretation, even when intended by the database designer. And in many cases that particular interpretation would be false or senseless (e.g. tables for M:M relationships).

The heart of the problem

We lack a shared framework
that could explicitly and formally map the relevant features in the domain of interest
to the relation or tree schemas for data about those things

Such a framework could be used to guide relation and tree schema development and revision
and to identify the common domain features reflected in different relation and tree schemas.

The framework would also provide relation and tree schemas with a *semantics*
and as a consequence their instances, relations and trees, would have meaning and assert propositions

Without that we really don't know, formally, what a relation (or an XML document) is telling us

Yes, it's in our head, but that not good enough

We need yet *another level of abstraction*

The Solution: Ontologies

The new problem (review)

The solution (historically):

Entity Relationship Modeling.

What it is

Why it works

Ontologies generally

What they are

How they look today (RDFS/OWL style)

Ontologies provide a way to abstract away from data structures such as tables and trees and focus on *information*: facts about things and relationships in the domain of interest.

And many data curation advantages ensure

The heart of the matter

We lack a shared framework
that could explicitly and formally map the relevant features in the domain of interest
to the relation or tree schemas holding data about those things

Such a framework could be used to guide relation and tree schema development and revision
and to identify the common domain features reflected in different relation and tree schemas.

The framework would also provide relation and tree schemas with a *semantics*
and as a consequence their instances, relations and trees, would have meaning and assert propositions

Without that we really don't know, formally, what a relation (or an XML document) is telling us

Yes, it's in our head, but that not good enough

We need yet *another level of abstraction*

The Solution

The old problem / solution:

There are many different ways to use *storage methods* to store *data*,

so, we need a single *data abstraction* that allows us to work with data regardless of what storage methods are used.

The solution: the relational model or the tree model

The new problem:

There are many different ways use *data abstractions* to record *information*

So we need a single *information abstraction* that allows us to work with information regardless of how the data expressing that information is stored.

The solution: . . . ?

The Solution

In 1976 Peter Chen proposed a simple solution.

Here's my interpretation:

Conceptualize your domain of interest
in terms of its things, relationships, etc.,
and then map that conceptualization
to whatever logical model schemas are being used

This (also) changed the world.

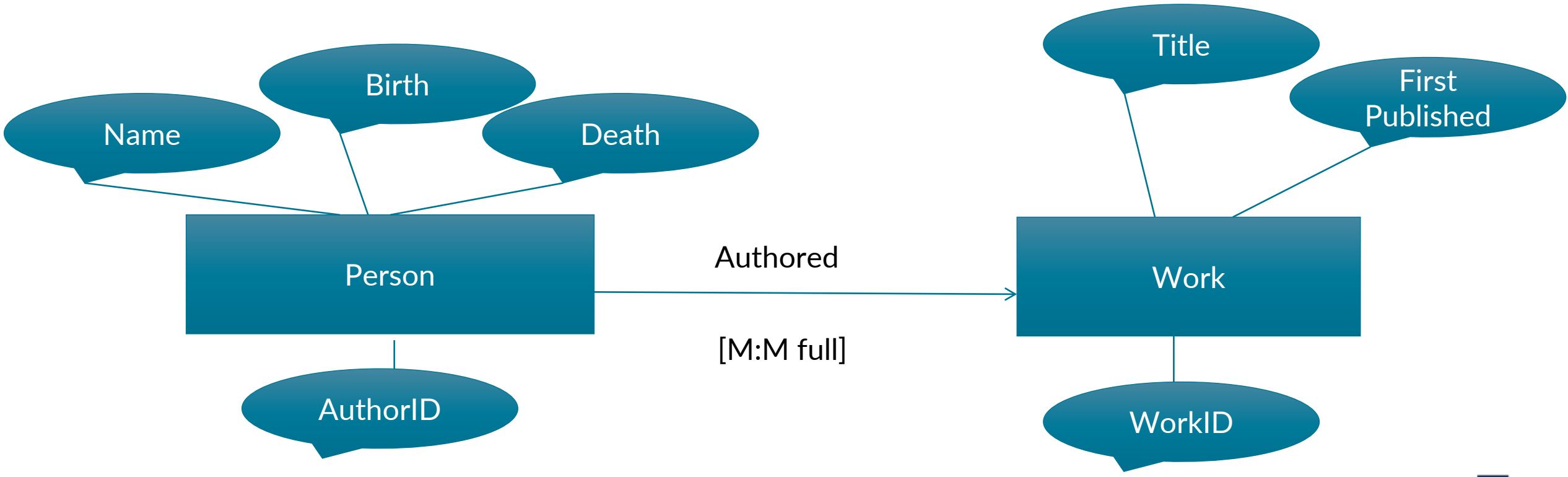
Peter Chen

Peter Pin-Shan Chen, "The Entity-Relationship Model-Toward a Unified View of Data" *ACM Transactions on Database Systems* (1970). *Also one of the most influential papers in computer science.*

Traditional ER Diagrams

An Entity Relationship diagram

Compare also UML class diagrams, etc.



The ER model has A foundation in first order logic (FOL)

Intensionally, *rectangles* indicate a kind (type) of entity.
Extensionally, they are a set of entities of a type.

In FOL: $\{ x: Fx \}$

Intensionally *attributes* are dyadic predicates.
Extensionally they are a function mapping entities to values in a domain

In FOL: $\{ \langle x,y \rangle : Rxy \ \& \ (\forall z)(Rxy \supset z=y) \}$

Intensionally *relationships* are (also) dyadic predications, but subject to a range possible cardinality and participation constraints.

Why is that logical basis important

We now have a description of the structure of the domain, a specification of the real world things, attributes, and relationships, in that domain.

Such as description can been called a *conceptual schema*.

A conceptual schema really is about the world:

its predicates correspond to real world properties (they are not simply names of domains); its variables range over real world individual things, explicitly, not by informal interpretation.

Connecting a conceptual schema to a logical level schema can give the instances of that logical schema *meaning*, formalizing how they express propositions, how they express *information about the world*.

Conceptual schemas solve the problem described earlier: they can be used to guide the development of logical schemas, or characterize the common of two different schemas that partially or totally equivalent.

And finally, perhaps most importantly, conceptual schemas provide critical *documentation* of the dataset.

Where does the mapping occur?

The most common use of conceptual models is to support the creation of logical schemas

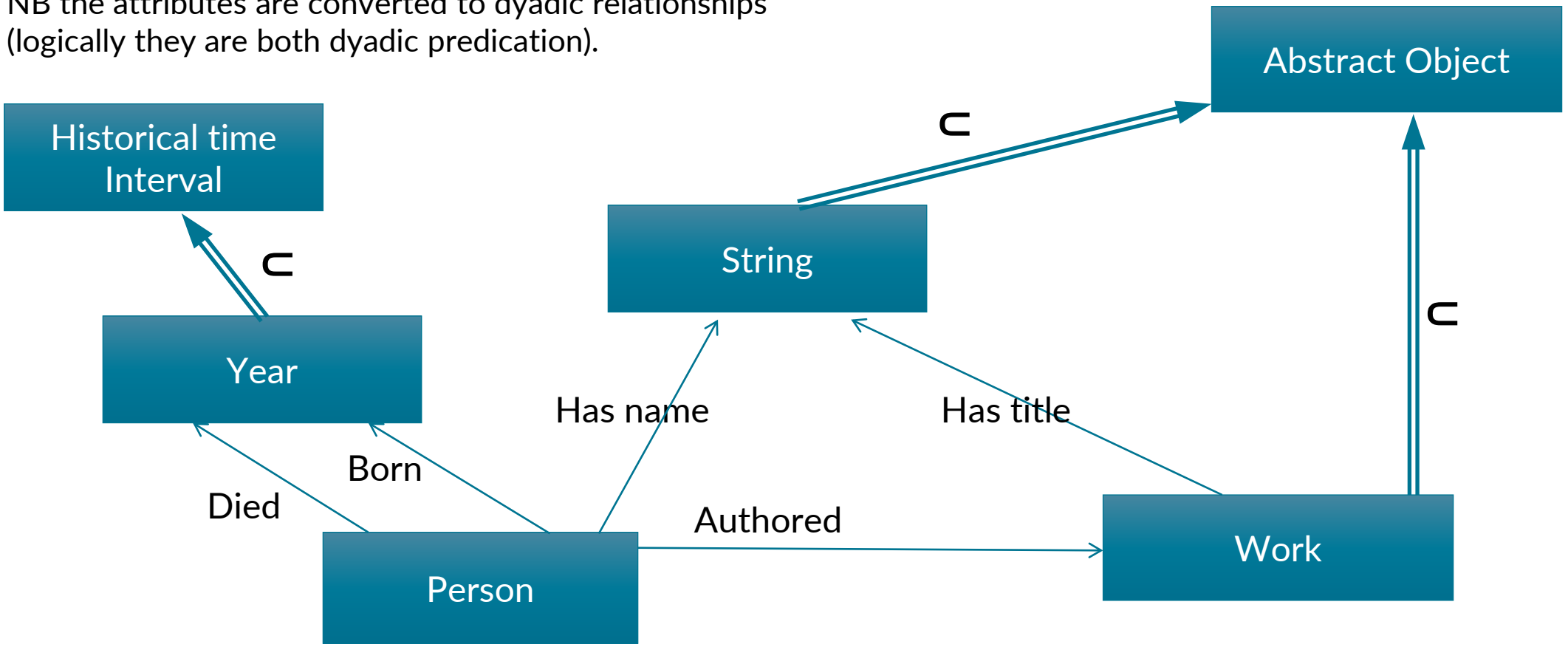
This typically involves the step by step generation of relational schemas from an ER schema

Most of you are familiar with the algorithm for this (there are many examples available on the web).

A data curation point: if your relational schema was generated mechanically from an ER schema then that ER schema reliably documents, at least partially, the relationship between your view of the domain and your relational schema. This is, obviously, very valuable.

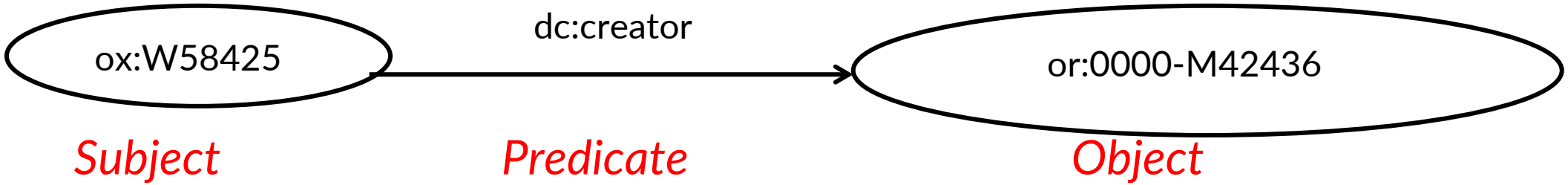
Generic Conceptual Modeling Today

NB the attributes are converted to dyadic relationships (logically they are both dyadic predication).

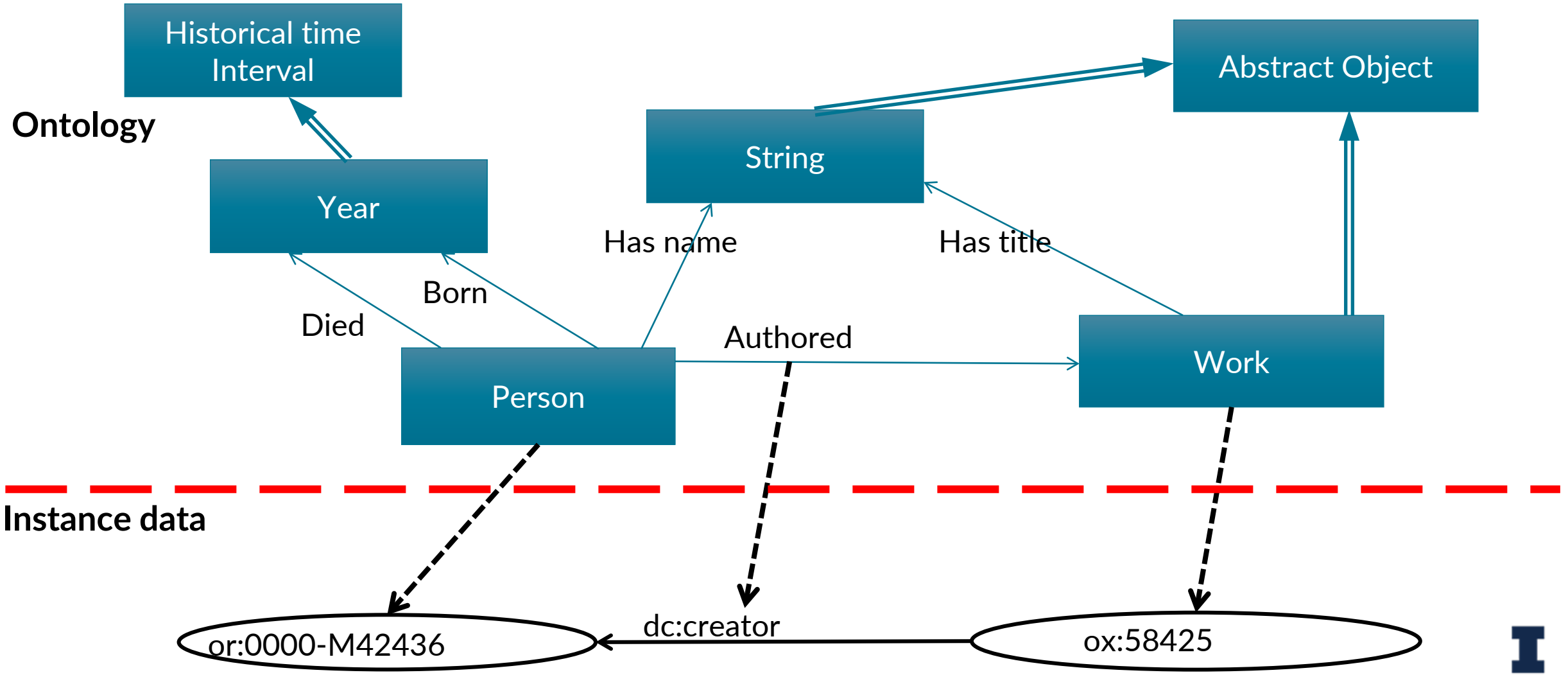


Now we sneak in A third logical level model

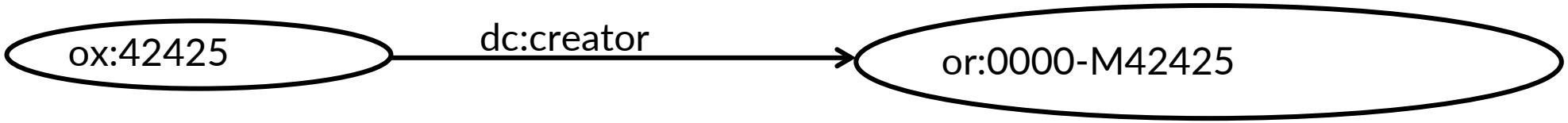
- The emergence of contemporary ontologies sparked interest in a logical level data model that had not been particularly important in the late 20th century: simple dyadic predication
- It makes its appearance as the “RDF triple”, typically represented with the generic graph model



Ontology + Instance Data



RDF Statements in N3 Triples



ox:W42425	dc:creator	or:0000-M42436
ox:W24246	dc:creator	or:0000-H24246
ox:W42427	dc:creator	or:0000-H24236

Compare:

WorkID	dc:creator	Title	First Published
W58425	M42425	Moby Dick	1851
W85246	H24236	The Scarlet Letter	1850
W55427	H24236	Fanshawe	1828

Hmm . . .

What is an Ontology?

“An ontology is an explicit specification of a conceptualization of a domain.”

- A body of formally represented knowledge is based on a *conceptualization*:
  - The objects, concepts, and other entities that are assumed to exist in some area of interest and the relationships that hold among them.
- A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.

  [Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly.]

Tom Gruber (Stanford), "What is an Ontology?”

Conceptual Models vs. Ontologies

I don’t see any useful distinction between conceptual models and ontologies and so will be using “ontologies” to include models like ER.

Nevertheless there are slight differences of emphasis:

- Ontologies almost always have class relationships (and perhaps an extensive class hierarchy)
- Ontologies will often include very abstract concepts at the levels of that hierarchy (*physical object*, *set*, *event*, *time interval*, *property*, etc.)
- Ontologies are usually intended to be relative stable and multipurpose.
- Ontologies are sometimes accompanied by logical axioms that support inferencing.
- Sometimes a specific formal technique is used for developing ontologies
- Often in ontology development minimizing the number of general kinds of entities is a concern.

Conceptual Models for XML Trees?

You will have noticed that most of this discussion is about conceptual models, or ontologies, that can be mapped to relational schemas.

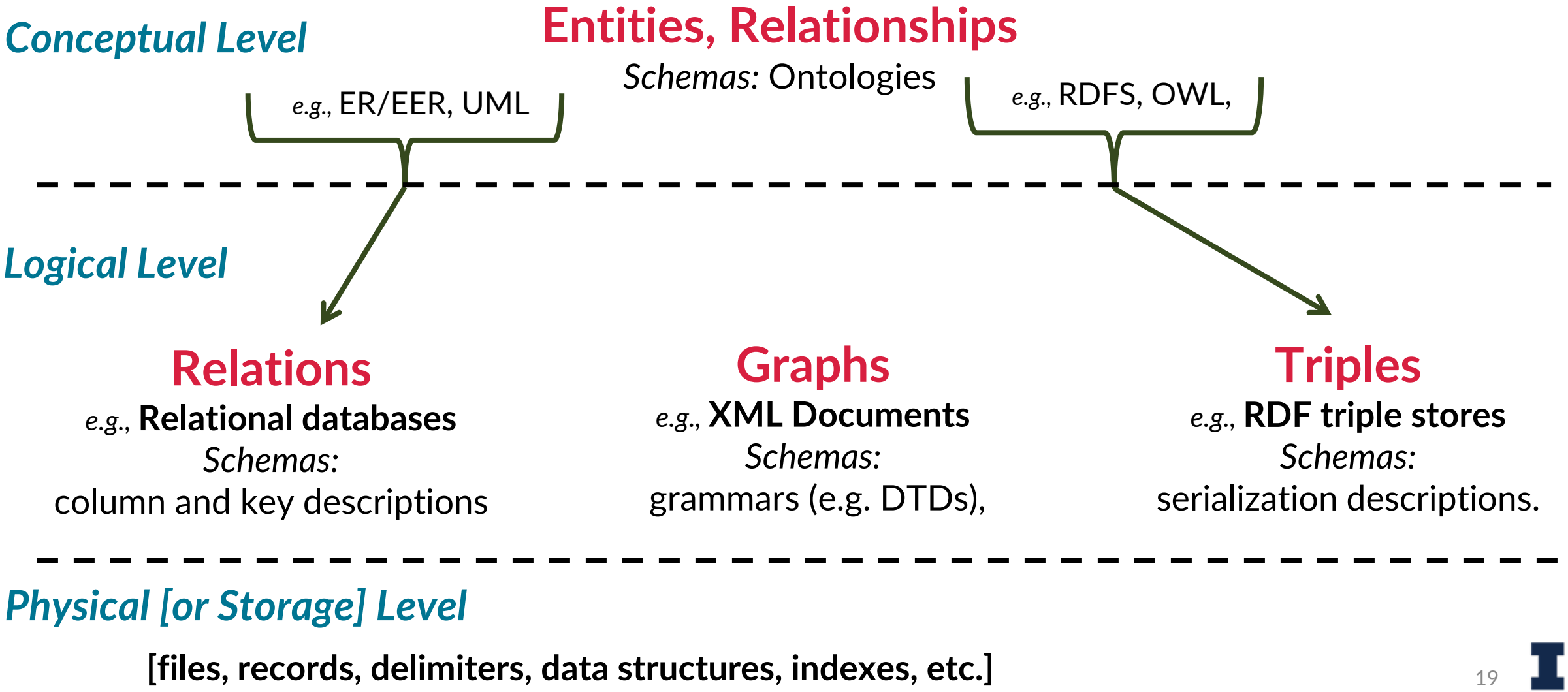
But what about the tree model? What about XML documents with descriptive markup?

Turns out that the semi-structured nature of XML documents makes it very challenging to develop a conceptual model that will play the same semantic role as ER diagrams do for the relational model.

For more on that see:

> “Towards a Semantics for XML Markup.”
> *Proceedings of the 2002 ACM Symposium on Document Engineering*,
> New York: Association for Computing Machinery. 2002.
> Allen H. Renear, David Dubin, C. M. Sperberg-McQueen, and Claus Huitfeldt.

Data Model Relationships



An ER/Ontology Example: FRBR

- We will show how a fragment of the FRBR conceptual model was developed in order to support the complexities of library cataloguing.
- Although the example is from library cataloguing the solution is directly related to problems in data curation and digital asset management, as we will see.

What is a *book*?

Did you read the same book I did?

This can mean different things:

the same novel I read?	Yes, but the French translation (text)
the same text I read?	Yes, but but the large type edition
the same edition I read?	Yes, but the copy in the summer house
the same copy I read?	The one with the mustard stain

FRBR: Functional Requirements for Bibliographic Records

The ordinary “book” can be simultaneously

about Manet,
in French,
typeset in neo-Bauhaus,
mustard-stained

Which is, strictly speaking, impossible.

so FRBR replaces the book with four kinds of entities:

the work	is about Manet,
the expression	is in French,
the manifestation	is typeset in neo-Bauhaus,
the item	is mustard-stained

The FRBR Group 1 Entity Types

There are four Group 1 entity types

Work: “a distinct intellectual or artistic creation”

Expression: “the intellectual or artistic realization of a work in the form of alphanumeric, musical, or choreographic notation, sound, image … etc.”

Manifestation: “the physical embodiment of an expression of a work”.

Item: “a single exemplar of a manifestation”

Or, colloquially (for book-like objects): work, text, edition, copy

Illustrative attribute assignments

Each entity type is assigned a *distinctive* set of attributes…

- works have attributes such as *subject* and *genre*
- expressions have attributes such as *language*
- manifestations have attributes such as *typeface*
- items have attributes such as *condition* and *location*.

NB: these attribute assignments are *disjoint*

A work may have a *subject*.
It does not have a *language*, *typeface*, or *condition*.

An expression may have a *language*;
It does not have a *subject*.

A manifestation may have a *typeface*.
It does not have a *subject* or a *language*

An item may have a *condition*
It does not have a *subject*, *language*, or *typeface*.

Foundations

FRBR…

•generalizes current best practice in cataloguing and bibliographic control

•reflects an emerging theoretical consensus

and longstanding bibliographic thinking, e.g.,

Seymour Lubetzky: work vs book

Eva Verona: literary unit vs bibliographic unit

See also Patrick Wilson, Elaine Svenonius, Akos Domanovszky, Martha Yee, et al.

•refines and extends current notions,
proposing new concepts based on new analysis
and anticipating new formats and new technologies

FRBR: Adoption and Influence

Don’t be misled by the example of a traditional paper book, FRBR is not only a major event in cataloguing and technology in libraries

But it is influential in developing ontologies for digital asset management and data curation:

cf. “When a scientist asks ‘what data do you have?’ …”

—Joe Hourcle (NASA/SDAC), “FRBR in a scientific context” 2007

As we will see

The method (simplified), in two steps.

The first step in the entity analysis technique is to:

1) isolate the kinds (types) objects (entities) that are of interest.

   Don't focus on data but on the things the data describe.

   Each entity type becomes a focal point for a cluster of data.

Once this high-level structure for the model has been charted

2) identify the important attributes of each entity.
   - identify relationships between entities
   - identify attributes and values

.

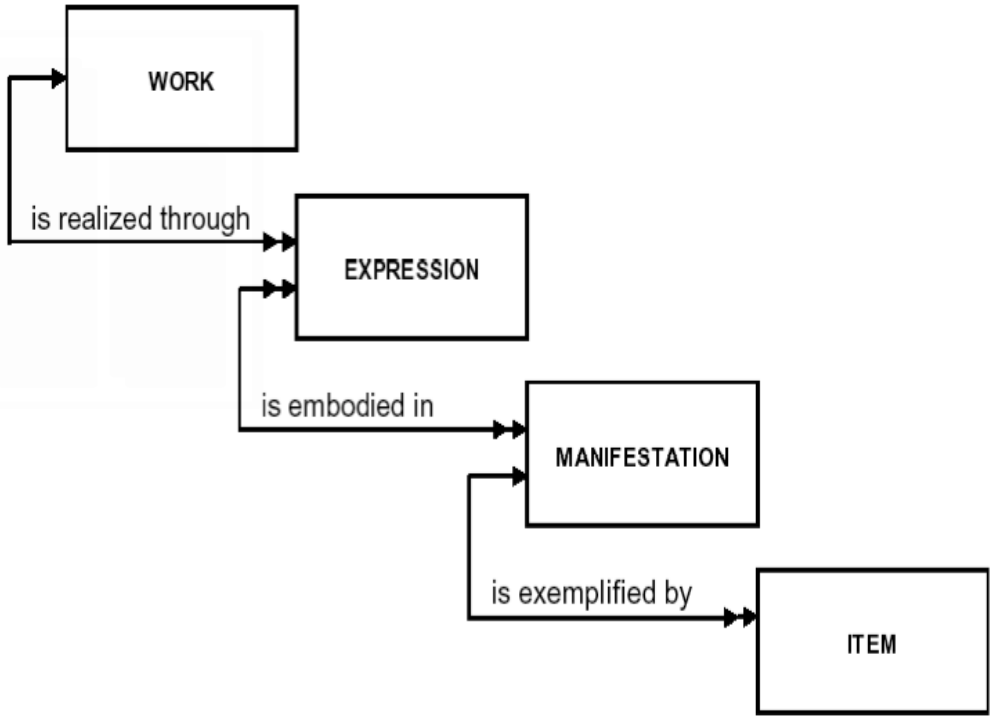
Relationships between the entities types

The novel Moby Dick (a work)

-- is realized through many different expressions (the particular texts of Moby Dick)
(the text of the 1859 edition, the corrected edition, the Murray French translation…)

-- each of which maybe embodied in many various manifestations
(the 1859 edition, a microform of that edition, an ebook with the 1859 text…)

-- each of which may exemplified by many different items
(my copy, the Houghton copy, the Beinecke copy, the Ransom copy)



Overview

Entities

- Group 1: Work, Expression, Manifestation, Item
- Group 2: Person, Family, Corporate Body
- Group 3: Concept, Object, Event, Place

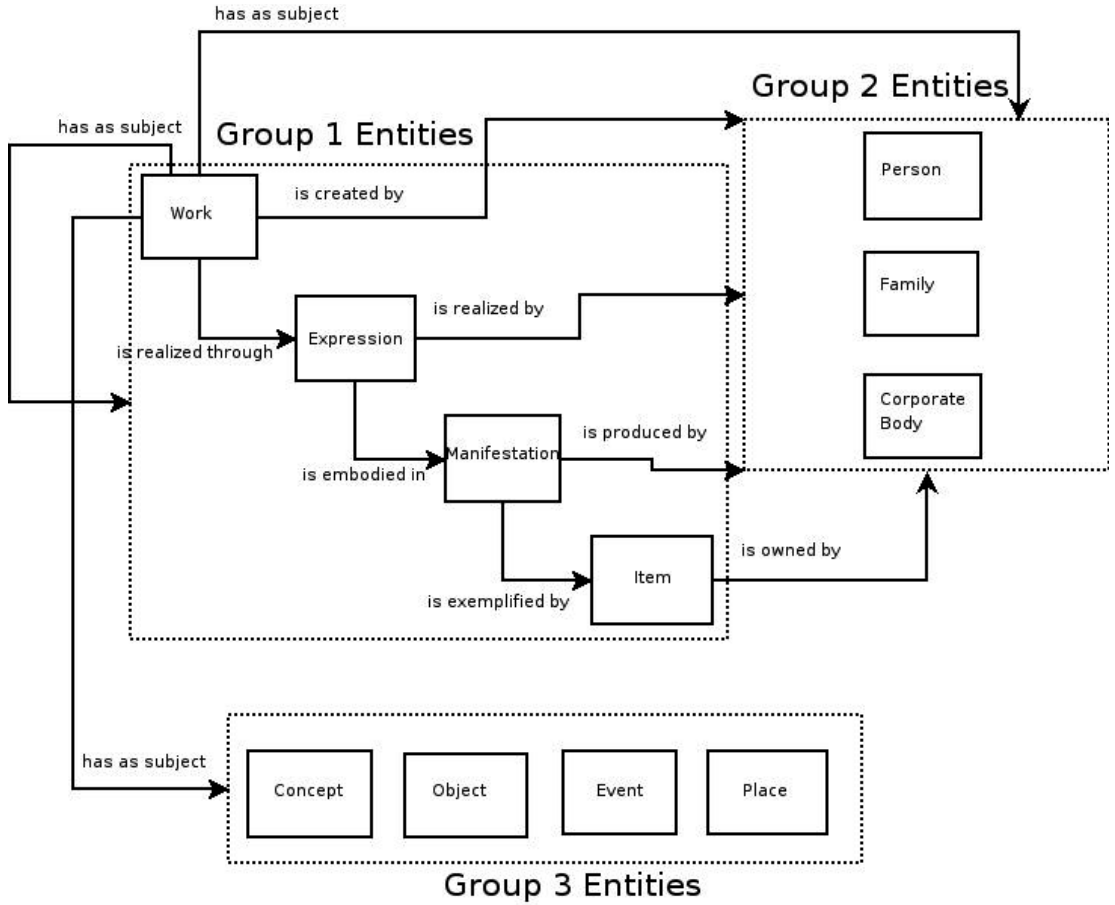
Attributes

- Clusters of appropriate attributes for each entity.
- Some “inherent”, other “externally imputed”

Relationships

- Some “integral to the definition of the entities”: e.g. exemplifies, embodies, realizes. Others are not: adaption, translation, abridgement, part of.
- Some exists between entities of the same type, others between entities of different types.

The FRBR Entity Groups



Not shown:
-- attributes
-- lower-level relationships

At http://www.frbr.org/files/entity-relationships.png; unattributed, pls send email to renear@uiuc.edu if you know the attribution.

FRBR as an ontology (using FOL) (work with Karen Wickett)

Definitions

work(x)		… x is an artistic or intellectual creation
expression(x)	=df	(∃y)[realizes(x,y)]
manifestation(x)	=df	(∃y)[embodies(x,y)]
item(x)	=df	(∃y)[exemplifies(x,y)]

Primitives:

work(x), realizes(x,y), embodies(x,y), exemplifies(x,y)

Cardinality Axioms

(realizes(y,x))	→	[(∀z)(realizes(y,z) → (z=x)]
(exemplifies(y,x))	→	[(∀z)(exemplifies(y,z) → (z=x)]

Domain/Range Axioms

realized(x,y)	→	work(y)
embodies(x,y)	→	expression(y)
exemplifies(x,y)	→	manifestation(y)

Disjointness Axioms

expression(x)	→	~ [work(x) v manifestation(x) v item(x)]
manifestation(x)	→	~ [work(x) v expression(x) v item(x)]
item(x)	→	~ [work(x) v expression(x) v manifestation(x)]

Theorems

realizes(x,y)	→	~(∃z)[embodies(x,z) v exemplifies(x,z)]
embodies(x,y)	→	~(∃z)[realizes(x,z) v exemplifies(x,z)]
exemplifies(x,y)	→	~(∃z)[realizes(x,z) v embodies(x,z)]

all relationships are now irreflexive and asymmetric; and trivially, transitive.

Implementing Ontologies In RDF/RDFS

- The RDF graph model for predication
- RDF Schema: for defining a simple ontology
- An example
- Our data model diagram is now completed!

RDF, RDFS, OWL

Today ontologies are often expressed in RDFS or OWL, theses are powerful logic-based languages designed to be read a processed by software, including inferencing systems.

RDF: Resource Description Framework

A simple model for predication; can be serialized in different languages

RDFS: RDF Schema

Basic schema concepts that can be used to define schemas for RDF instances

OWL: Web Ontology Language

General schema concepts for more advanced schemas

Resource Description Framework

- RDF gives a general model for describing things - by saying that some *thing* has a particular *property* with a particular *value*.

- These descriptions are intended to be processed by software and shared across applications and computing environments without loss of meaning.

- RDF can be expressed as a graph or serialized in various ways, including XML.

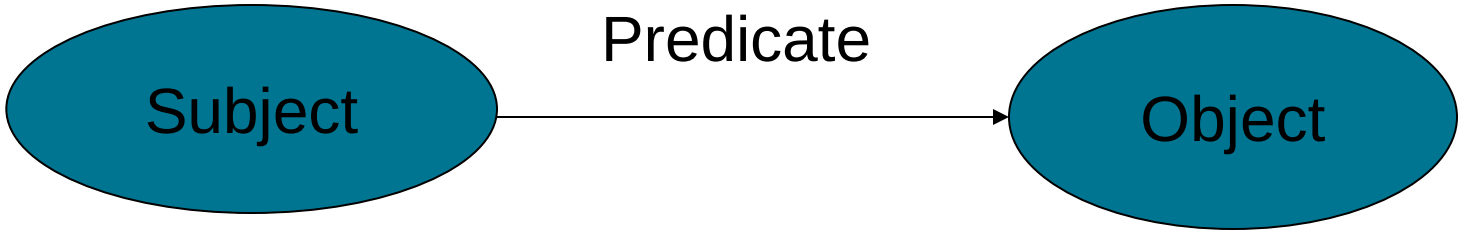
RDF Data Model

The basic abstract structure of a single descriptive assertion in RDF is a triple, an ordered set of these three elements:

- Subject (thing)
- Predicate (property)
- Object (value)

RDF Graph Languague

The representation of a triple in the RDF graph language:



An RDF graph is a collection of such triples

NB: Where ER has distinct constructs for both attributes and relationships, RDF uses a single construct, the predicate/property, for both. After all, under the hood they are both just two-place FOL predications R(x,y),as we have seen.

Example - To Express in RDF

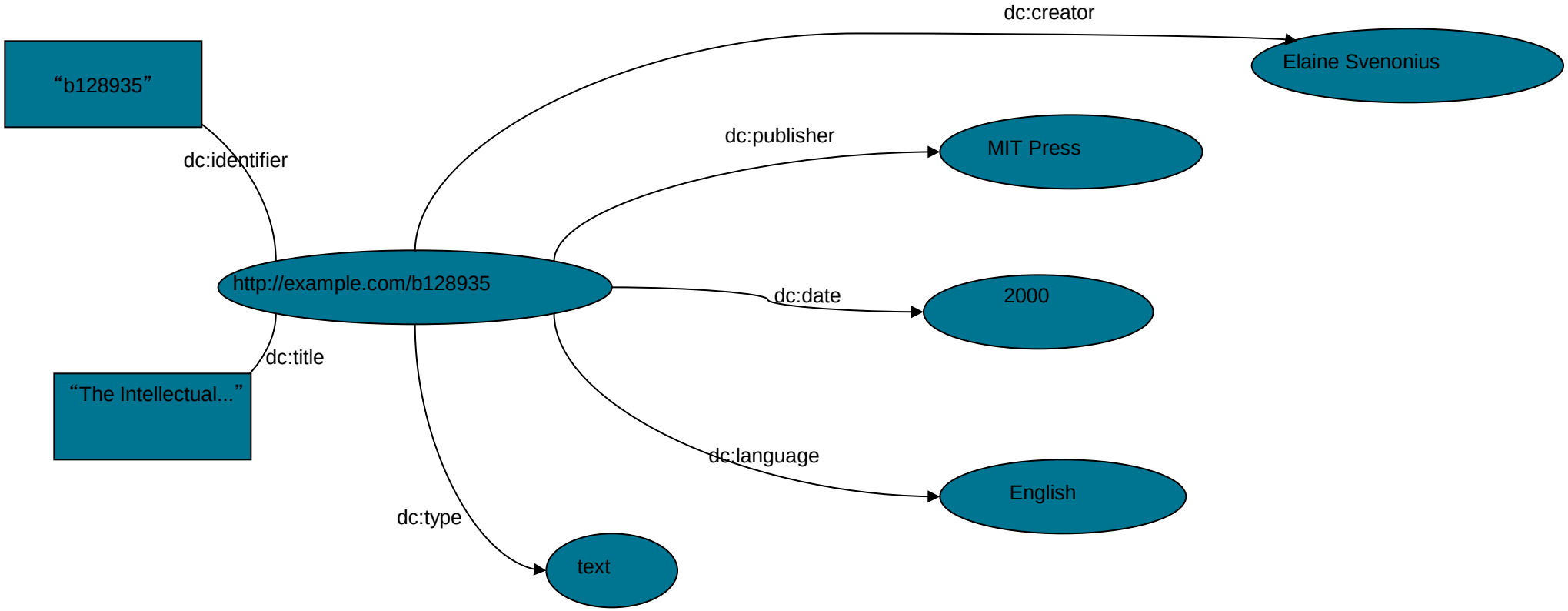
A simple bibliographic description using Dublin Core*:

- `identifier = b128935`
- `title = The Intellectual Foundation of Information Organization`
- `creator = Elaine Svenonius`
- `date = 2000`
- `publisher = MIT Press`
- `language = english`
- `type = text`

- *http://dublincore.org/documents/dces/

Expressing DC in RDF

The description as an RDF graph



NB: This graph represents a set of seven triples.

RDF Description as Triples

<b128935,dc:identifier,”b128935”>
<b128935,dc:title,”The Intellectual...”>
<b128935,dc:creator,Elaine Svenonius>
<b128935,dc:date,2000>
<b128935,dc:publisher,MIT Press>
<b128935,dc:language,english>
<b128935,dc:type,text>

But that picture is pre-FRBR

The RDF graph on the previous slide is, of course, not consistent with FRBR.

To adapt that description to the FRBR conceptual model, we need a system of related classes and properties.

That is, we need a schema, the sort of thing an ER diagram is:

- A schema language that interoperates with RDF
- That would be: RDF Schema

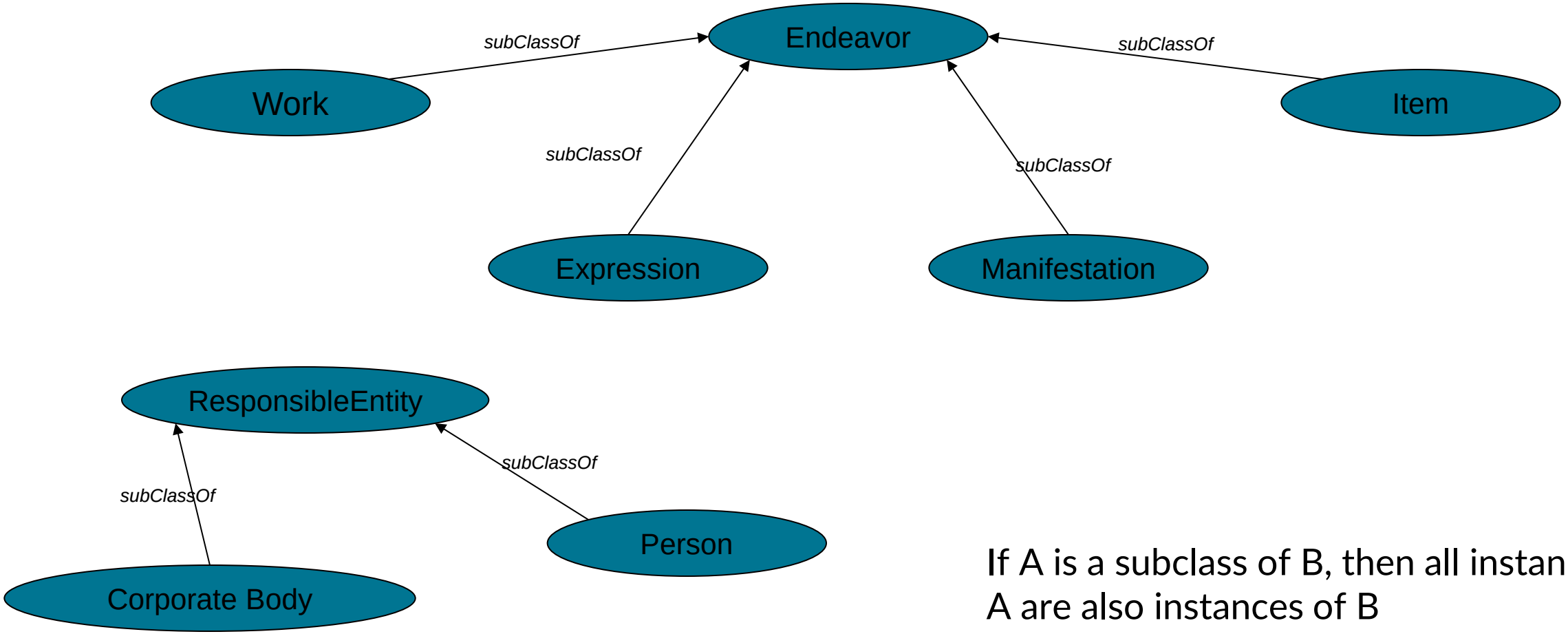
RDF Schema

RDF Schema is a schema language for defining RDF vocabularies:

- The RDF Schema language itself uses RDF triples as its syntax (!)
- It needs some vocabulary for its own. That is, it needs some fundamental properties and classes that it can use to define RDF vocabularies for specific domains. See the next slide.

What follows is largely based on the RDF vocabulary for FRBR published here: http://vocab.org/frbr/core

FRBR in RDFS: Classes

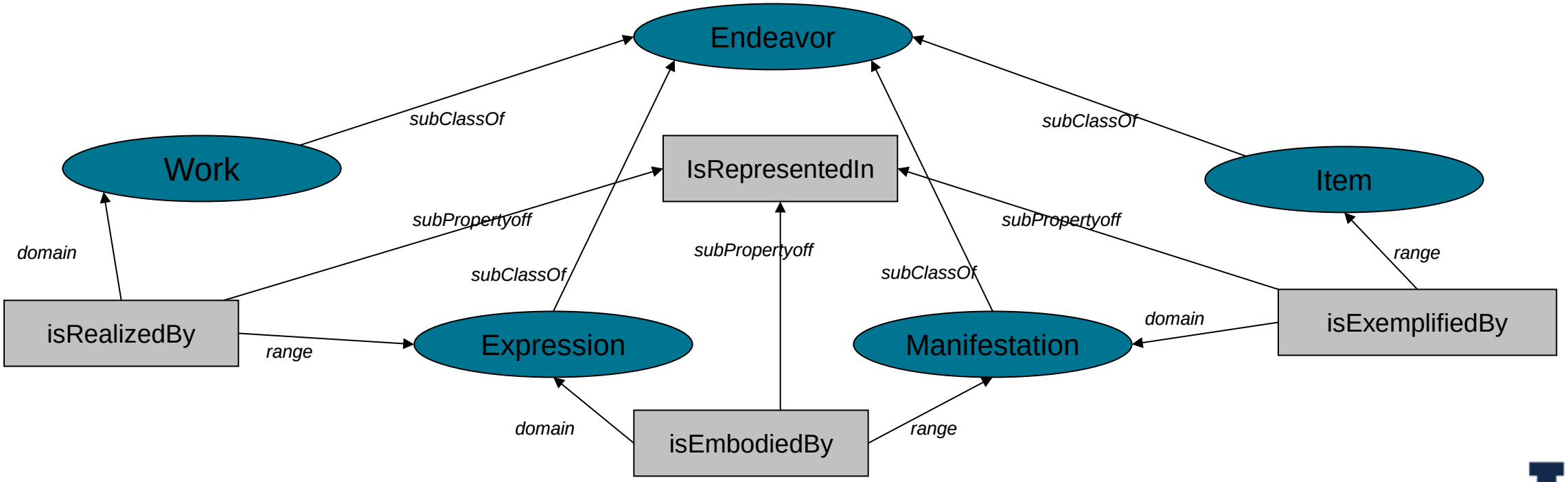


If A is a subclass of B, then all instances of A are also instances of B

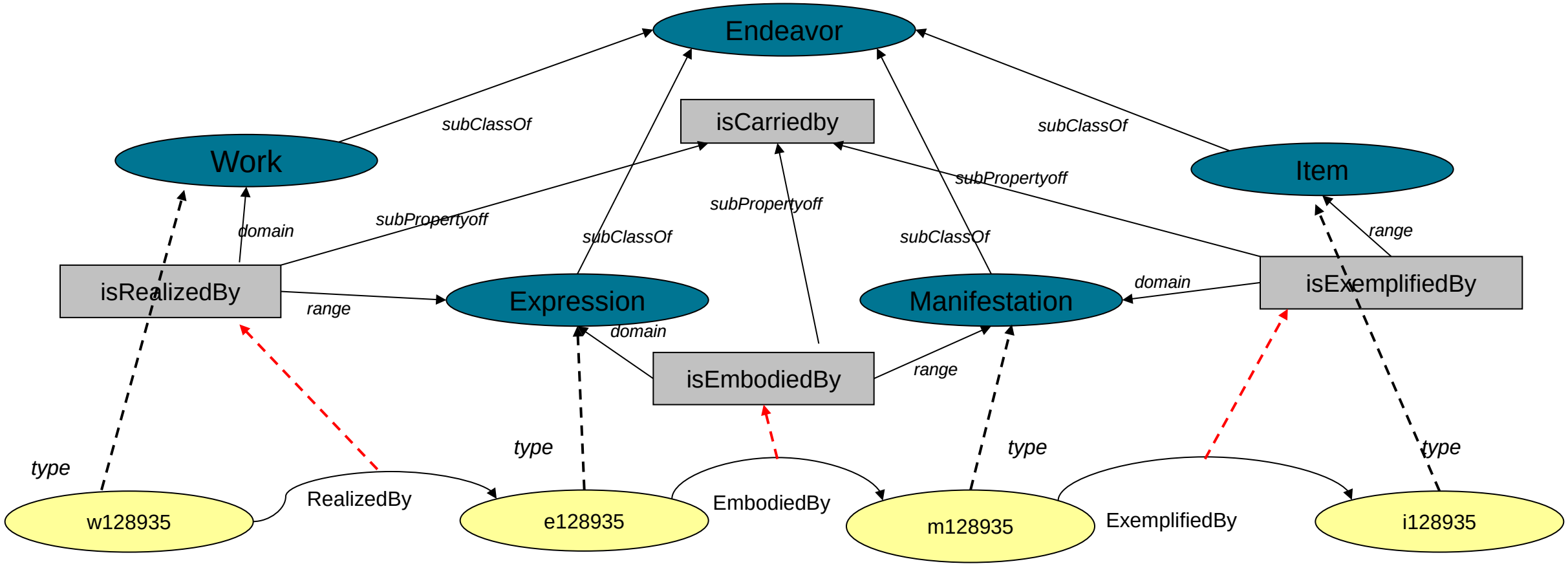
FRBR in RDFS: Properties

Properties have domains and ranges indicated by RDFS core properties domain and range.

Properties can also be arranged in a hierarchy just as classes can, indicated by the RDFS core property subproperty.



Connecting an Instance to the Schema: FRBR Entities

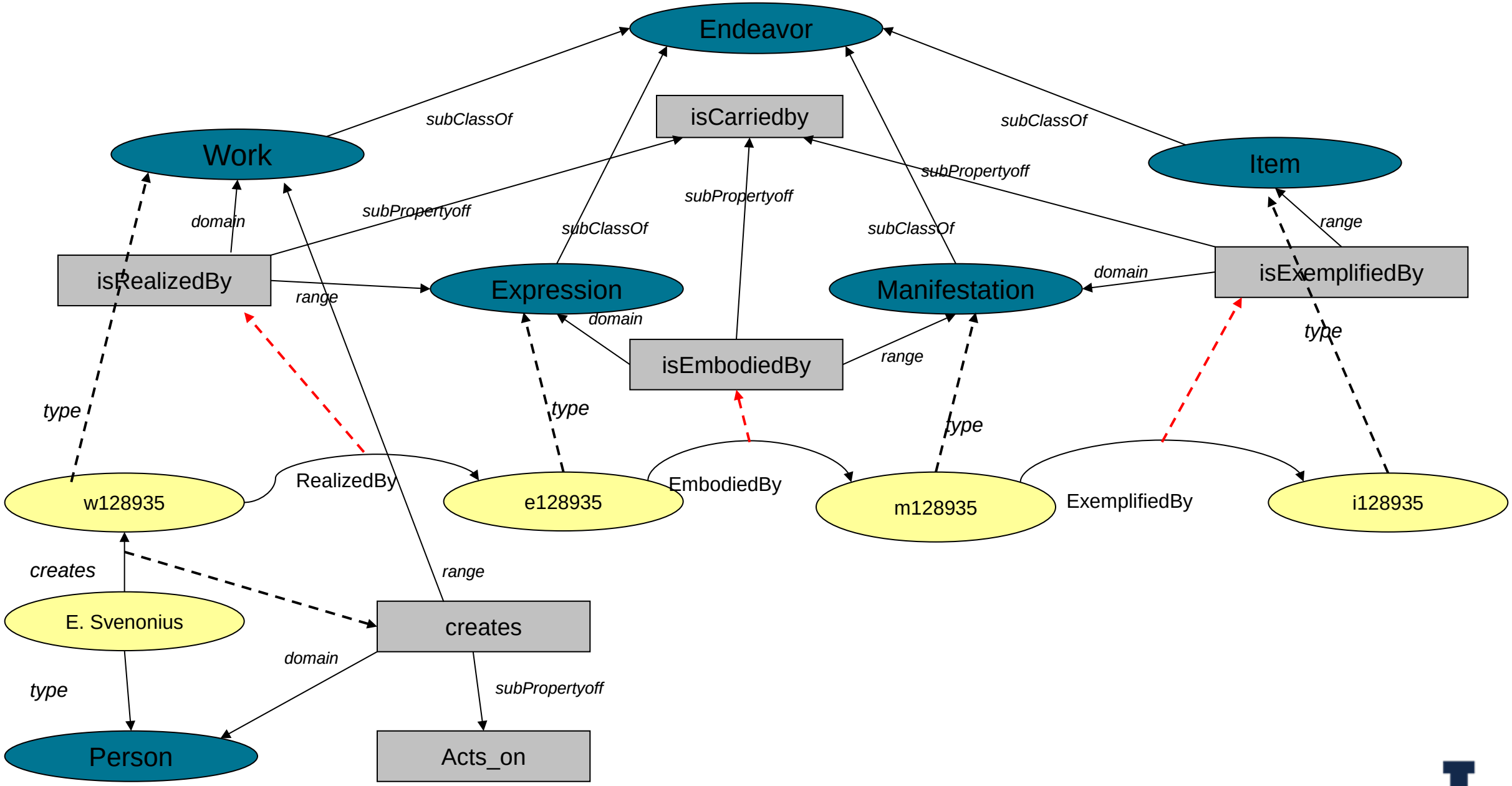


Connecting an Instance to the Schema: FRBR Entities

Here we add to the picture some “instance data,” four actual bibliographic entities: the work Intellectual Foundations, the text of the MIT Press 2000 edition, the edition itself, and an individual copy, all with their relevant inherent FRBR relationships asserted to hold between them.

All instance entities and relationships matched with the relevant schema level constructs

Notes: In RDFS the relationship between instance level property and the matching schema level property is identity, and so shown here with an unnamed red dotted arrow.


Endeavor
isCarriedby
Work
Item
subClassOf
subClassOf
subPropertyoff
subPropertyoff
subPropertyoff
domain
range
subClassOf
subClassOf
isRealizedBy
Expression
Manifestation
isExemplifiedBy
domain
range
domain
range
type
type
type
type
isEmbodiedBy
RealizedBy
EmbodiedBy
ExemplifiedBy
w128935
e128935
m128935
i128935
creates
range
E. Svenonius
creates
domain
type
subPropertyoff
Person
Acts_on

Data Model Relationships