LRM to Schema.org Cheat sheet v0.9.3

rene.voorburg@kb.nl / 2020-11-30 / work in progress

Summary of recent changes:

0.9 .2 to 0.9.3 (processed feedback from the KB bibliographical metadata expertgroup)

- fixed incorrect property for schema:alternateName
- added / changed template for persons and name formats.
- added clarifications regarding the use of datePublished dateCreated, dateModified
- added: context of use determines how to deal with inheritance

0.9 to 0.9.2 (processed feedback from Enno Meijers, Bob Coret, Jeffrey van der Hoeven, Meta van der Waal-Gentenaar)

- reverted proposed policy on 'generic typing' of digitized materials as just 'CreativeWorks';
- added clarification on **Res** entities and other vocabularies;
- added more context to the introduction;
- split up the core principles part from the introduction;
- fixed the range for schema:locationCreated in examples
- various minor updates and additions, typos fixed, etc.

- to 0.9 (processed feedback from Richard Wallis, Richard Ligtenberg)

- changed approach related to inheritance and data duplication
- always add 'CreativeWork' as a type, also for subtypes

Contents

1. Introduction

This document provides a set of guidelines, rules and examples for using the schema.org[1] ontology and vocabulary to describe bibliographical entities that are modelled according to the IFLA Library Reference Model[2] (LRM). Schema.org provides an RDF compliant and cross domain data model that has been broadly accepted as a general-purpose metadata standard, for use far beyond its original purpose of supplying structured data to search engines. That is why publishing bibliographical metadata in schema.org provides an excellent foundation for libraries to interlink and become interlinked with other knowledge domains. This helps to maximize the potential value library data has for society.

The use of schema.org in the bibliographical domain has been pioneered and advanced by Richard Wallis, amongst others in his role of the group chair of the bibframe2schema.org initiative[3], which provides a reference mapping for bibliographical data modeled according to BIBFRAME. The guidelines in this document are largely based on work done by him. However, instead of using BIBFRAME as a reference, this document provides an application profile for bibliographical metadata based on IFLA LRM. It was primarily written to support the goal to standardizing the bibliographical metadata provided by data.bibliotheken.nl[4], the linked data publication platform of the KB, the national library of the Netherlands[5].

In its current incarnation, this application profile is merely a set of guidelines, rules and examples, not a complete mapping from for example Pica+ or RDA-based metadata to schema.org. It is assumed the reader has some basic knowledge of both IFLA LRM and RDF.

[1] https://schema.org/
[2] IFLA: https://www.ifla.org/files/assets/cataloguing/frbr-lrm/ifla-lrm-august-2017_rev201712.pdf
[3] https://bibframe2schema.org/
[4] http://data.bibliotheken.nl
[5] https://www.kb.nl/

2. Core principles

Design for LRM classes

Central to this *application profile* is IFLA LRM and its classes **Work**, **Expression**, **Manifestation**, **Item** (WEMI) and **Agent**. In order to create and describe schema.org entities, the first question that should be asked is whether the entity at stake is a **Work**, **Expression**, **Manifestation**, **Item** or **Agent**. The answer to this question determines how an entity is typed in schema.org and what properties and relations can be used and how they should be interpreted. Using schema.org properties, its domain and range restrictions should be followed.

Usually, a dataset following this application profile should at least consist of the schema.org equivalent of **Manifestations**, and preferably the related **Items**. Ideally, the related **Work** and **Expression** level entities are included too. For datasets that would consist of singleton[6] **Manifestations** only, it will suffice to just create the **Item** descriptions.

Describe things & things describing things

This profile adheres to the Linked Data best practice that things are not the same thing as their descriptions. So, the page providing the metadata describing any WEMI entity should be distinguishable by having its own URI, thus allowing the modeler to make statements about not just the entity at stake but also about its **metadata**. This distinction becomes important when for example adding licensing or copyright information.

Don't rely on inheritance of properties or on inferred classes

Various properties that describe bibliographical entities are inherited from its parent WEMI-entities, or the other way around. However, for many users these mechanisms of inheritance will not be obvious. So, when publishing bibliographical data using this application profile, as a default it is recommended to assume the user has no knowledge of the inheritance mechanisms of the underlying LRM-model. Depending on the context of use, such a user should be supported by providing all relevant properties, even those that are usually considered to be inherited from a parent entity.
Similarly, to help users understanding the provided data, one should be explicit and provide all types of the described entities and not rely on implicit or inferred typing.

Don't be restrictive

In the spirit of Linked Data and the Open World Assumption, this application profile is not intended to be restrictive. Modelling needs beyond what is provided here may be solved using schema.org constructs not exemplified in this document. Even more, the modeler should not feel restricted to use the schema.org as the sole vocabulary. However, one is advised to apply solutions provided by the broader schema.org ontology wherever possible.

[6] A singleton **Manifestation** has a one-to-one relationship with its underlying **Item**, it is a **Manifestation** with a single **Item**.

3. Namespace URIs

As a reminder, the schema.org namespace declaration (all examples and snippets provided using Turtle serialization):

```
@prefix schema: <http://schema.org/>.
```

Note: Some examples use 's:' instead of 'schema:' for sake of brevity.

In addition to schema.org, example in this document use the Web Ontology Language (OWL) for a semantically strict 'sameAs'-relations. Also, the XML Schema Definition (XSD) is used for specifying date formats. Additionally, rdf-schema is used to enrich the description of person-entities.

```
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix rdfs: < http://www.w3.org/2000/01/rdf-schema# >.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
```

4. Entity classes

CreativeWorks as Work, Expression, Manifestation or Item

At the core of this bibliographical application profile stands the `schema:CreativeWork` class and its subclasses. Being based on IFLA LRM, each `schema:CreativeWork` entity exist as either a **Work**, an **Expression**, a **Manifestation** or an **Item**.

The following template should be used to create LRM WEMI entities in schema.org:

```
<Work> a schema:CreativeWork .

<Expression> a schema:CreativeWork, schema:ProductGroup.

<Manifestation> a schema:CreativeWork, schema:ProductModel .

<Item> a schema:CreativeWork, schema:IndividualProduct, schema:ArchiveComponent.
```

Some notes:

- The `schema:ProductGroup` approach for **Expressions** is not standard. No alternative approach rooted in schema.org is known. An alternative approach might be to add the relevant RDA (rdac) class in addition to the schema.org classes.
- Applying the `schema:ArchiveComponent` to **Items** allows the modeler to use `schema:holdingArchive` relation, otherwise it is not required.

In addition to using `schema:CreativeWork`, it is desirable to add one or more of its subclasses, fitting the entity at stake.

Serial publications

Schema.org has specific `schema:CreativeWork` subclasses for describing serial publications and its members or groups of members. Please use *table 1* as a reference.

Table 1 Classes for serial publications, groups and member

	Series:	*Volume*	*Issue*
Generic	`s:CreativeWorkSeries`	`s:PublicationVolume`	`s:PublicationIssue`
Books	`s:Bookseries`	`s:PublicationVolume`	`s:Book`
Periodicals	`s:Periodical`	`s:PublicationVolume`	`s:PublicationIssue`
Newspapers	`s:Newspaper`	`s:PublicationVolume`	`s:PublicationIssue`
Comic books	`s:ComicSeries`	`s:PublicationVolume`	`s:ComicIssue`

Some notes:

- Be as specific as possible, but don't be over specific. Mind that both `schema:ComicSeries` and `schema:Newspaper` are a *subclass* of `schema:Periodical` and `schema:CreativeWorkSeries` is the *superclass* of these.
- In this application profile, a `schema:PublicationVolume` is *not* seen as an LRM WEMI entity, so one should refrain from dual typing it using the `schema:Product` classes.

Agent, Collective Agent and Person

In contrast to IFLA LRM, no class for an **Agent** type entity exists in schema.org. However, classes for its subclasses **Person** or **Collective Agent** entity do exist. These are respectively `schema:Person` and `schema:Organization`. Use one of those for **Agents**.

Please note that the definition of the LRM **Person** and that of `schema:Person` are *not fully compatible*. In LRM the entity **Person** is restricted to real persons who live or are assumed to have lived. It explicitly excludes fictional persons, while the schema definition *does* include fictional persons. For example, in LRM , J.K. Rowling, Robert Galbraith and indeed Joanne Rowling are identical as a **Person**. In schema.org, it is allowed to consider these three *persona*[7] as three different `schema:Persons`. The approach of splitting a **Person** up into separate `schema:Person` entities for each *persona* can be used to retain the information by what pseudonym a publication was written, without having to rely on **Nomen**[8] entities.

The following example describes two books written by Joanne Rowling, one using the name "J.K. Rowling", the other using the name "Robert Galbraith".

```
<http://data.bibliotheken.nl/id/nosuch/whpphilstone>
  a schema:CreativeWork , schema:Book ;
  schema:name "Harry Potter and the Philosopher's Stone" ;
  schema:author <http://data.bibliotheken.nl/id/pers/jkrowling> .

<http://data.bibliotheken.nl/id/nosuch/wtroubledblood>
  a schema:CreativeWork , schema:Book ;
  schema:name "Troubled Blood" ;
  schema:author <http://data.bibliotheken.nl/id/pers/robertgalbraith> .

<http://data.bibliotheken.nl/id/pers/jkrowling> .
  a schema:Person ;
  schema:name " J.K. Rowling";
  schema:sameAs <http://data.bibliotheken.nl/id/pers/robertgalbraith> .

<http://data.bibliotheken.nl/id/pers/robertgalbraith>
  a schema:Person ;
  schema:name " Robert Galbraith" ;
  schema:sameAs <http://data.bibliotheken.nl/id/pers/jkrowling> .
```

Some notes:

- The relation `schema:sameAs` is used to link together the `schema:Persons` that are derived from one LRM **Person**.
- Mind that `schema:sameAs` does not have the same strict semantics as `owl:sameAs`[9] has. The semantics behind `owl:sameAs` would essentially result in merging the two `schema:Persons` into one, making it impossible to infer by what author name the books were published.

Metadata

When it is desirable to provide statements about the description of an entity, a **Metadata** entity should be used. This approach is not based on IFLA LRM but on linked data best practices. In this application profile, **Metadata** entities, or entities describing properties of descriptions of 'ordinary' entities should be dual typed as a `schema:Dataset` and a `schema:Webpage`. In the following example, a **Metadata** entity is used to provide statements regarding the *last modification date* and the *license* of the description of a book.

[7] *persona*: the aspect of someone's character that is presented to or perceived by others
[8] A **Nomen** in LRM is a name (essentially represented by a string) as an entity instead of a literal so additional attributes can be used to describe it. See further on in this chapter.
[9] http://www.w3.org/2002/07/owl#sameAs

```
<http://data.bibliotheken.nl/id/nbt/w1234567>
  a schema:CreativeWork , schema:Book ;
  schema:name "De Aanslag" ;
  schema:author <http://data.bibliotheken.nl/id/thes/p06854796X> ;
  schema:inLanguage "nl" ;
  schema:dateCreated "1982"^^xsd:gYear ;
  schema:mainEntityOfPage <http://data.bibliotheken.nl/doc/nbt/w1234567> .

<http://data.bibliotheken.nl/doc/nbt/w1234567>
  a schema:WebPage, schema:Dataset ;
  schema:mainEntity <http://data.bibliotheken.nl/id/nbt/w1234567> ;
  schema:isPartOf <http://data.bibliotheken.nl/id/dataset/nbt> ;
  schema:license <https://creativecommons.org/publicdomain/zero/1.0/> ;
  schema:publisher <http://data.bibliotheken.nl/id/thes/p76543> ;
  schema:dateModified "2019-06-21"^^xsd:date .
```

Some notes

- The relation `schema:mainEntityOfPage` is used to link the **Metadata** entity to the entity describing a book.
- When following the KB URI policy for linked data, the URI of the **Metadata** entity has "doc" as the first segment of the path, the actual object share the URI except for using "id" instead of "doc".

Nomen, Res or other entity classes

This application profile *does not* provide a schema.org counterpart for LRM **Nomen** entities. For names, simply string literals or URIs are used. However, in a limited way, string literals behave like entities in RDF. Specifically, RDF allows us to specify properties for a string like its language and its script (following BCP47[10]) . For example:

```
<URI_of_publisher> schema:name "Paskov Dom"@ru-Latn .
<URI_of_publisher> schema:name "Пашков Дом"@ru .
```

Note that in the approach propagated here of a `schema:Person` as a *persona*, the `schema:Person` entity can be used to cluster **Nomen**s (as literal strings). This in a manner much comparable to current practice in managing name authority records[11].

In accordance with how LRM defines the **Res** entity as "any entity in the universe of discourse[12] ", this application profile does allow the modeler to use other entity classes beyond those explicitly provided here. It is not required that these entities are described or even typed using schema.org. Implicitly, they may be considered to be typed as a `schema:Thing`. This will be specifically useful to link for example to external authorities or thesauri (usually provided using the SKOS vocabulary). Typically, these links will be made in schema.org using predicates like `schema:about`, `schema:genre`, `schema:audience` and `schema:sameAs`.

[10] https://tools.ietf.org/html/bcp47
[11] Cf IFLA p. 90 (https://www.ifla.org/files/assets/cataloguing/frbr-lrm/ifla-lrm-august-2017_rev201712.pdf).
[12] IFLA, p. 20 (https://www.ifla.org/files/assets/cataloguing/frbr-lrm/ifla-lrm-august-2017_rev201712.pdf).

5. Relations between WEMI entities

Hierarchical relations inside the WEMI-tree

The LRM WEMI entities of a publication stand in a hierarchical relation to each other. Table 2 shows how these relations are to be represented in schema.org.

Table 2 Hierarchical relations for WEMI entities in schema.org

	object:			
subject:	**Work**	**Expression**	**Manifestation**	**Item**
Work	-	**`s:workExample`**	`s:workExample`	`s:workExample`
Expression	**`s:exampleOfWork`**	-	**`s:workExample`**	`s:workExample`
Manifestation	`s:exampleOfWork`	**`s:exampleOfWork`**	-	**`s:workExample`**
Item	`s:exampleOfWork`	`s:exampleOfWork`	**`s:exampleOfWork`**	-

Note: Other than in RDA there are just two predicates that may be used respectively for any 'upwards' or 'downwards' relation. Relations shown in **bold** are preferred, but it is not forbidden to skip a (missing) WEMI level.

Usually, entities will link to their parent (using `schema:examleOfWork`), but is specifically for **Works** and **Expressions** it is a good practice to provide a link downwards to a *representative* child of the entity using `schema:workExample`. See figure 1 for an example.

Lateral relations

Lateral relations are relations for which both subject and object have *the same (implicit) LRM WEMI type*. They are relevant for example for describing derivations, translations, digitized items and part-whole relations.

Translations and derivations

Both at the **Work** and the **Expression** level entities might be based on another entity (having the same WEMI-type). A generic way to express this is to use `schema:isBasedOn`[13]. Specifically for translations, which occur at the **Expression** level, `schema:translationOfWork` can be used. An example can be found in figure 1.

[13] Mind that schema.org has no inverse relation for `schema:isBasedOn`.

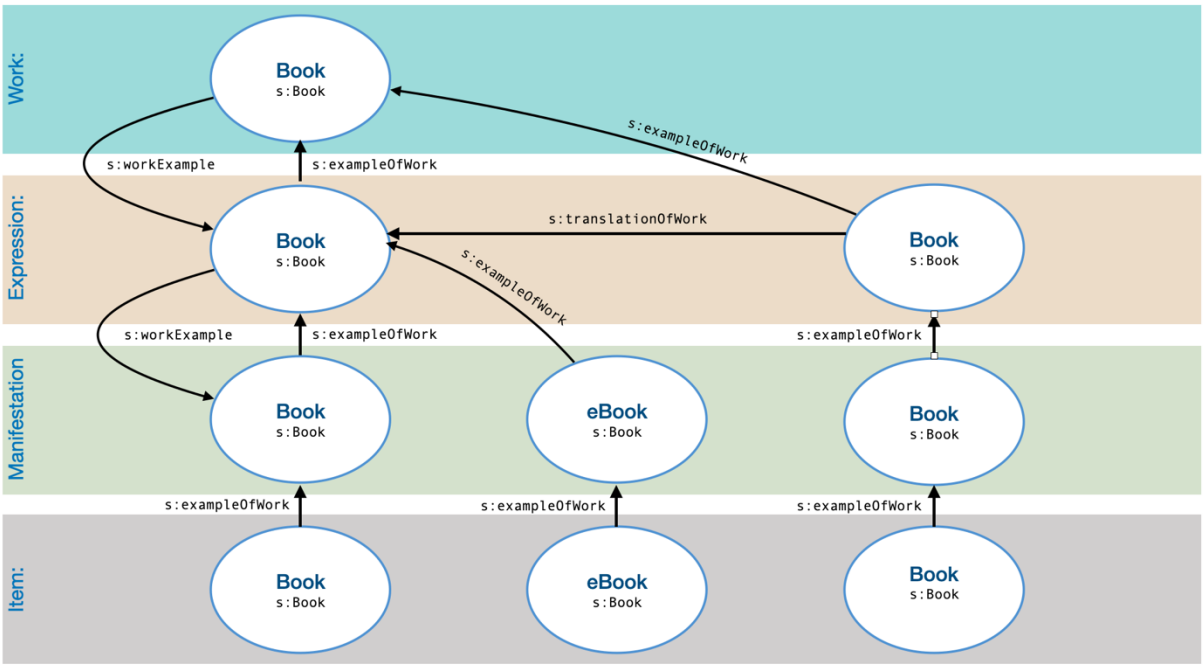

Figure 1 Example of a digital reprint and a translation of a simple monograph

Note: In figure 1 an e-book is typed in schema.org as a `schema:Book`, not as a `schema:EBook` since that is not a class in schema.org but an enumeration for with the `schema:bookFormat` property.

Digitized Items

Digitizing a printed **Item** will produce in (at least) one new **Item** and a new **Manifestation**. This will result in the following lateral relations:

```
<Digtal_Item> schema:encodesCreativeWork <Printed_Item> .

<Digital_Manifestation> schema:isBasedOn <Printed_Manifestation> .
```

Figure 2 shows an example of the entities and relations in schema.org when digitizing a simple monograph.

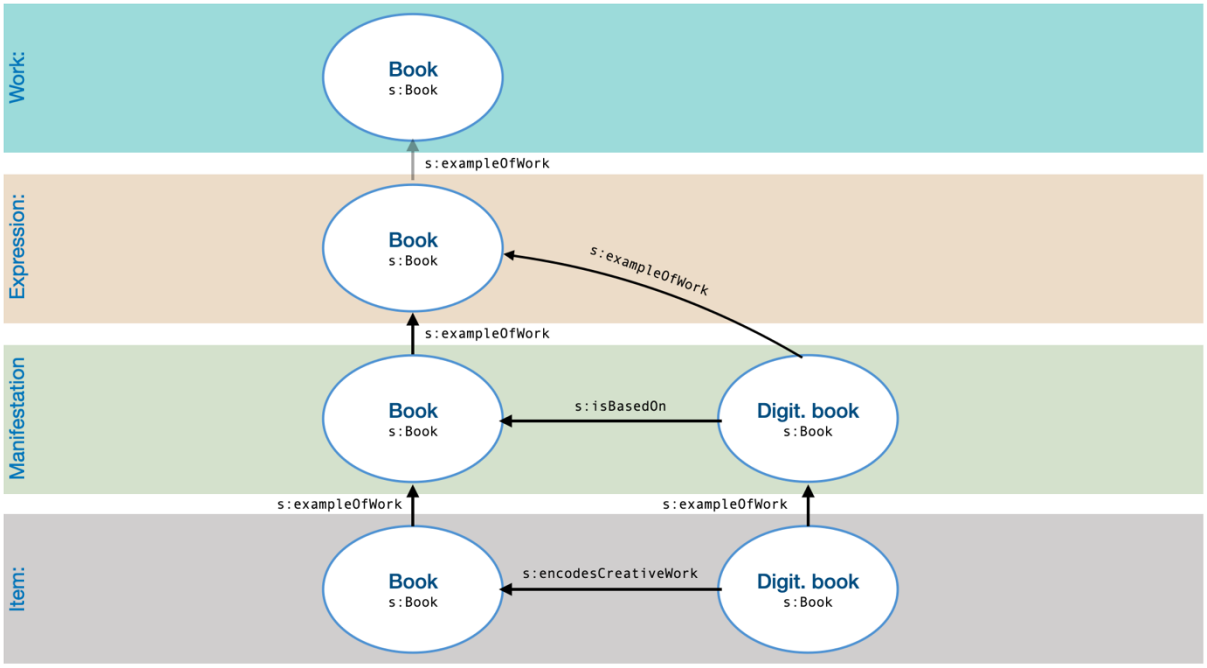

Figure 2 Digitizing a monograph

Part to whole relations

Another category of lateral relations are part to whole relations. For those `schema:isPartOf`, and when needed the inverse `schema:hasPart`, is used.

For example, a `schema:PublicationIssue` can be considered to be an part of a `schema:PublicationVolume`. See figure 3 for an example using `schema:BookSeries`. Figure 4 presents the entities and relations for a serial publication, specifically a `schema:Newspaper`.

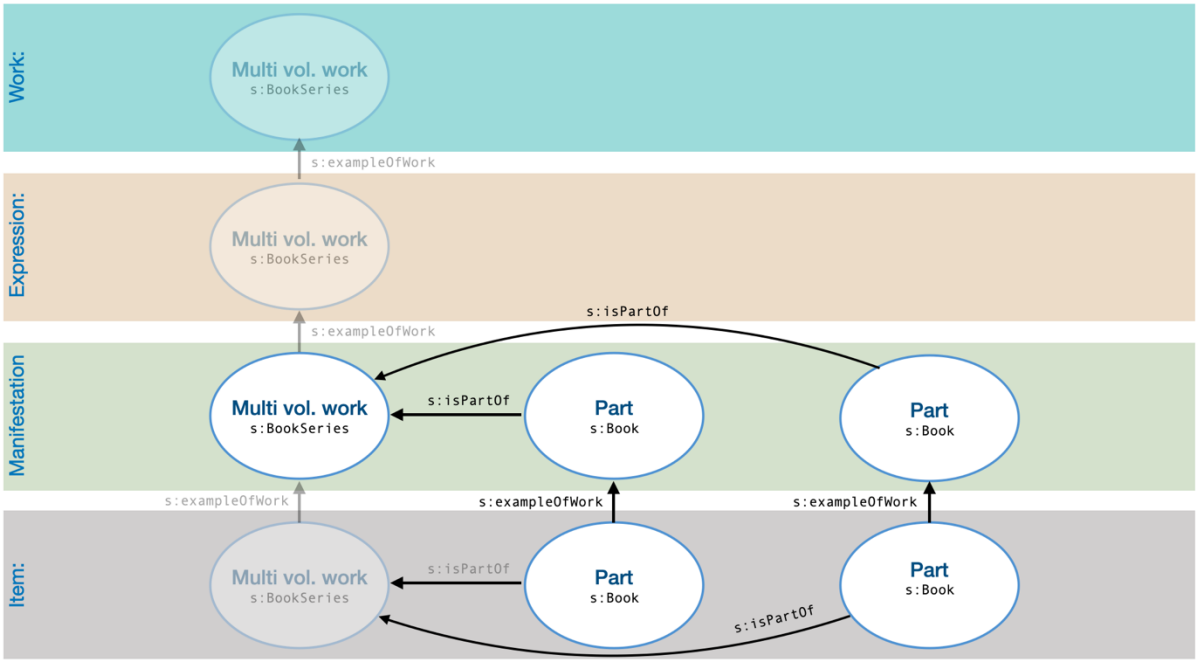

Figure 3 A multivolume bookseries using isPartOf relations

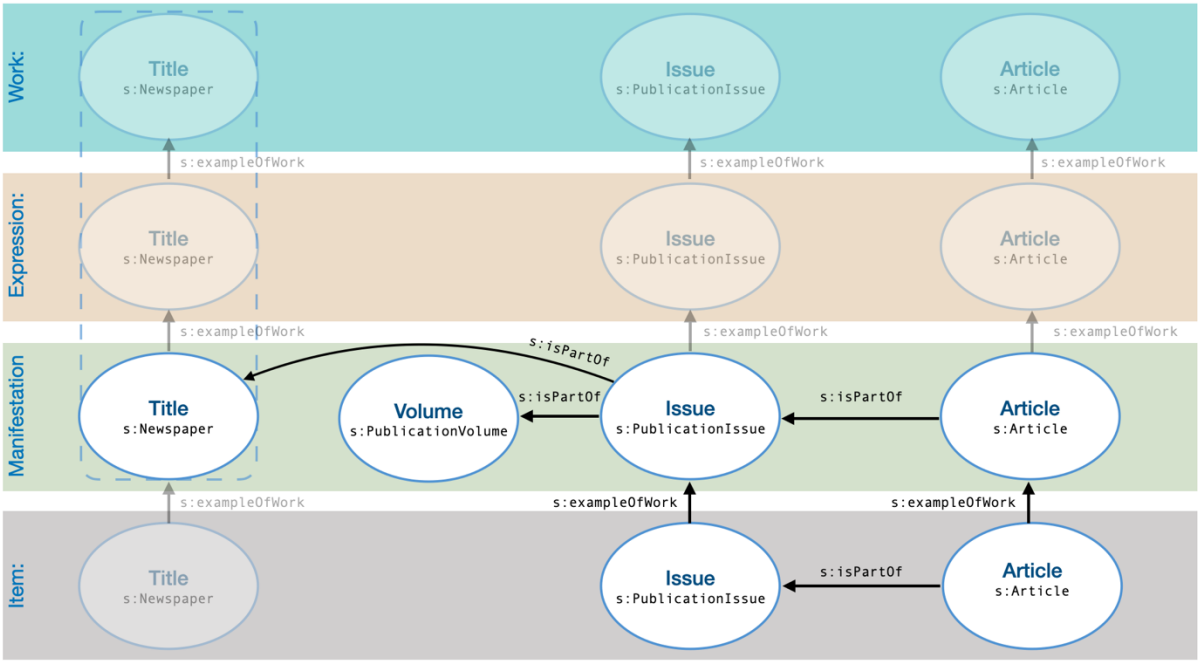

Figure 4 Part-whole relations in a serial publication

Notes:

- In figure 4 a **Volume** is just a `schema:PublicationVolume`, it will not be defined at a specific WEMI level entity.

6. Entity templates and clippings

Work entities

A **Work** is the intellectual or artistic content of a distinct creation. It is an abstract object that allows the grouping of related **Expressions**[14].

The following provides a template for a Work instance in schema.org:

```
<a_template_for_a_Work_instance>
  a schema:CreativeWork, schema:{subclass of CreativeWork} ;
  schema:name "{work title}";
  schema:alternateName "{alternative title}", "{yet another title}"  ;
  schema:author <URI_of_author> ;
  schema:inLanguage "{bcp47 language code}"
  schema:dateCreated "YYYY"^^xsd:gYear ;
  schema:locationCreated "<URI>" ;
  schema:about "subject, string", <{or_URI}> ;
  schema:genre "{name of genre}", <{or_URI}> ;
  schema:description "{a description of work at stake}" ;
  schema:workExample <{URI_of_representative_Expression}> .
```

All properties beyond the class and work title (`schema:name`) should be considered optional. In addition to the properties in the template above, lateral relations (`schema:isBasedOn`, `schema:hasPart` or `schema:isPartOf`) and hierarchical relations may be added. Usually, entities lower in the WEMI-tree will link to their parent, but is a good practice to provide a link to a representative child of the **Work**, using `schema:workExample`.

A `schema:identifier` might be supplied, but the best practice is to treat the URI of the instance as its persistent identifier.

Expression entities

An **Expression** is the specific intellectual or artistic form that a **Work** takes each time it is "realized". It excludes incidental aspects of physical form[15]. In schema.org we discriminate the **Expression** level from the other WEMI levels by dual typing entities as a `schema:ProductGroup`, in addition to the `schema:CreativeWork` class or subclass.

Some properties and relations that are (usually) meaningless when used at the **Work** level but that are specifically relevant to **Expressions** are:

```
schema:translator <{uri_of_translator}>
schema:editor <{uri_of_editor}>
schema:translator <{uri_of_translator}>
schema:license <{uri_of_license}>
schema:audience "{audience}", <{uri_of_audience}>
```

Specifically, these lateral relations are relevant for Expressions:

[14] IFLA LRM (https://www.ifla.org/files/assets/cataloguing/frbr-lrm/ifla-lrm-august-2017_rev201712.pdf) p 21.
[15] IFLA LRM (https://www.ifla.org/files/assets/cataloguing/frbr-lrm/ifla-lrm-august-2017_rev201712.pdf) p 23.

```
schema:translationOfWork <{uri_of_Expression}>
schema:isBasedOn <{uri_of_Expression}>
```

An example of a description of an instance of an **Expression**:

```
<an_example_of_an_Expression_instance>
  a schema:CreativeWork, schema:Book, schema:ProductGroup;
  schema:name "Kwaad Bloed";
  schema:author <http://data.bibliotheken.nl/id/pers/robertgalbraith> ;
  schema:exampleOfWork <http://data.bibliotheken.nl/id/w/trbldbld> ;
  schema:translator <http://data.bibliotheken.nl/id/pers/pqr> ;
  schema:translationOfWork <http://data.bibliotheken.nl/id/e/trbldbld> ;
  schema:inLanguage "nl" ;
  schema:dateCreated "2020"^^xsd:gYear ;
  schema:genre "thriller"@nl ;
  schema:description "Privédetective onderzoekt coldcases."@nl ;
  schema:workExample <http://data.bibliotheken.nl/id/nbt/p1234> .
```

Manifestation entities

A **Manifestation** entity is defined by IFLA as "A set of all carriers that are assumed to share the same characteristics as to intellectual or artistic content and aspects of physical form. That set is defined by both the overall content and the production plan for its carrier or carriers[16] ". Further, IFLA notes that "A manifestation is recognized from the common characteristics exhibited by the items resulting from the same production process. The specification of the production process is an intrinsic part of the manifestation".

For libraries, it is common to describe bibliographical materials at the **Manifestation** level. Each Manifestation should be typed as both a `schema:ProductModel` and a specific matching subclass of `schema:CreativeWork`. When no matching subclass exists, simply using `schema:CreativeWork` will do.

Properties that are introduced at the **Manifestation** level are related to the production process and the physical result of the process:

```
schema:publication [
  a schema:PublicationEvent ;
  schema:name "{publisher imprint as captured}" ;
  schema:startDate "{yyyy}"^^xsd:gYear ;
  schema:location "{place as literal <URI>}" ;
  schema:publishedBy <{URI_of_Agent_/publisher}> ;
] .
schema:isbn "{isbn}" ;
schema:issn "{issn}" ;
schema:numberOfPages "{literal / integer}" ;
schema:height: "{literal}" ;
schema:width: "{literal}" ;
schema:material "{literal_or_URI}";
schema:image "content_URL or schema:ImageOject" ;
schema:bookEdition "{bookedition}" ;
schema:contributor <{URI_of_contributor}> ;
```

[16] IFLA p. 25 (https://www.ifla.org/files/assets/cataloguing/frbr-lrm/ifla-lrm-august-2017_rev201712.pdf).

Some notes:

- The properties `schema:datePublished` and `schema:publisher` are not used in the context of **Manifestations** because only using a `schema:publicationEvent` entity additional properties as for example the place of publication can be registered.
- The `schema:PublicationEvent` is provided here as a blank node. No rules forbid to use a URI instead.

An example of a description of an instance of a **Manifestation**:

```
<an_example_of_an_instance_of_a_Manifestation>
  a schema:CreativeWork, schema:Book, schema:ProductModel ;
  schema:name "Het stenen bruidsbed: roman" ;
  schema:alternateName "Het stenen bruidsbed" ;
  schema:author <http://data.bibliotheken.nl/id/thes/p06854796X> ;
  schema:exampleOfWork <URI_of_parent_Expression> ;
  schema:isbn "9789023476825" ;
  schema:bookEdition "44" ;
  schema:publication [
    a schema:PublicationEvent ;
    schema:startDate "2013"^^xsd:gYear ;
    schema:location "Amsterdam;
    schema:publishedBy <URI_of_publisher> ;
  ] ;
  schema:numberOfPages "183" ;
  schema:inLanguage "nl" ;
  schema:genre "novel"@en .
```

Item entities

In LRM, an **Item** is the physical object. In this application profile, each **Item** is dual typed as a `schema:indivualProduct` type, in addition to `schema:CreativeWork` (or subclass). Further, if the `schema:holdingAchive` relation is to be used, the Item should also by typed as a `schema:ArchiveComponent`.

Following best practices for linked data, when an entity is an *information object,* following its URI should return the object itself. This could lead to the conclusion that for digital **Items**, its KB resolver link, which is advertised as a persistent URI, is simply the **Item's** URI. However, a digital **Item** is an information object, but a printed **Item** is not. For that very reason, printed **Items** do not have a KB resolver link. So to prevent inconsistent URIs, all Items should have a URI that adheres to the KB URI policy[17], for example `http://data.bibliotheken.nl/id/nbt/e127456283X`. For digital items, it is proposed to use `schema:contentUrl` for the resolver link leading to the actual digital object.

Properties specifically relevant for **Item** entities:

```
schema:holdingArchive <{URI_of_Organization}> ;
schema:itemLocation "location of Item, literal or URI" ;
schema:sku "{shelfmark}" ;
schema:contentUrl <URI_of_digital_object> ;
schema:contentSize "File size" ;
```

Usually an **Item** should be linked to its parent **Manifestation** using `schema:exampleOfWork`.

[17] add reference…

Two examples of a schema.org representations of imaginary **Items** are presented below. The first is a printed Item, the second is the result of a digitization of the first.

```
<http://data.bibliotheken.nl/id/nbt/e456283X>
  a schema:IndividualProduct ;
  a schema:CreativeWork, schema:Book ;
  a schema:ArchiveComponent;
  schema:name "Troubled Blood" ;
  schema:author <http://data.bibliotheken.nl/id/pers/robertgalbraith> ;
  schema:exampleOfWork <http://data.bibliotheken.nl/id/nbt/p192741446> ;
  schema:inLanguage "en" ;
  schema:publication [
    a schema:PublicationEvent ;
    schema:startDate "2020"^^xsd:gYear ;
    schema:location "London;
    schema:publishedBy [
      a schema:Organization ;
      schema:name "Sphere Books"
    ]
  ] ;
  schema:isbn "9781549106590" ;
  schema:genre "thriller" ;
  schema:holdingArchive  <http://data.bibliotheken.nl/id/thes/p075301482> ;
  schema:sku "GW A155684:3" .

<http://data.bibliotheken.nl/id/alba/e127456283X>
  a schema:IndividualProduct , schema:CreativeWork;
  schema:name "Troubled Blood";
  schema:author <http://data.bibliotheken.nl/id/pers/robertgalbraith> ;
  schema:exampleOfWork <http://data.bibliotheken.nl/id/nbt/p192741447> ;
  schema:inLanguage "en" ;
  schema:genre "thriller" ;
  schema:encodesCreativeWork <http://data.bibliotheken.nl/id/nbt/e456283X>;
  schema:contentUrl <http://resolver.host.org/resolve?urn=xyz:e1256283X> ;
  schema:contentSize "2134K" ;
  schema:dateCreated "2019-04-21"^^xsd:date .
  schema:datePublished "2019-06-02"^^xsd:date .
```

Some notes:

- In this example, both **Items** present properties that are inherited from the **Manifestation** level.
- The publisher in the example is essentially provided as a literal value, but in order to follow the range restrictions for the `schema:publishedBy` property, it is wrapped as a `schema:Organization` entity using a blank node.
- The digital **Item** has `schema:dateCreated` and `schema:datePublished` properties. Strictly in the context of a digital **Item**, they convey respectively when the very digital **Item** was created and when it was put online.
- See the discussion on the semantics of the `schema:dateCreated` and `schema:datePublished` properties, at the end of the section on **Metadata** entities.

Person entities

In this application profile, a `schema:Person` is in its core a *persona*, or a bundle of all names and name variants that belong to that persona. Personas that belong to the same LRM Person are linked together using `schema:sameAs`.

Schema.org does not prescribe how to format person names. In this application profile, `schema:name` is intended to provide the full literal name, so for example "`Robert Galbraith`", **not** "`Galbraith,`

Robert". For sorting purposes, Dutch[18] naming conventions hinder using simply the literal family name. The person "Jan van Dijk" has "van Dijk" as his family name, but that consists of a surname prefix "van" which should be ignored for sorting purposes, plus the base surname "Dijk". Schema.org doesn't provide a way to make this distinction explicit. This profile proposes to use rdfs:label to provide a sortable name. Typically, rdfs:label will provide the authorized access point, for example "Vondel, Joost van den (1587-1679)". In addition to schema.org, one might consider describing person names using the Person Name Vocabulary[19].

The following presents two descriptions of related persona.

```
<http://data.bibliotheken.nl/id/pers/robertgalbraith>
  a schema:Person ;
  rdfs:label "Galbraith, Robert"@en ;
  schema:name "Robert Galbraith"@en ;
  schema:sameAs <http://data.bibliotheken.nl/id/pers/jkrowling> .

<http://data.bibliotheken.nl/id/pers/jkrowling>
  a schema:Person ;
  rdfs:label "Rowling, J.K."@en ;
  schema:name "J.K. Rowling"@en, "Дж.К.Роулинг"@ru
  schema:alternateName "Rowling, Joanne"@en, "Роулинг, Джоан"@ru ;
  schema:familyName "Rowling"@en, "Роулинг"@ru ;
  schema:givenName "Joanne"@en, "Джоан"@ru ;
  schema:birthDate "1965"^^xsd:gYear ;
  schema:birthPlace "Yate" ;
  schema:gender schema:Female ;
  schema:description "J.K. Rowling is a writer and philanthropist."@en ;
  schema:sameAs <http://data.bibliotheken.nl/id/pers/robertgalbraith>.
  schema:sameAs <https://viaf.org/viaf/116796842> ;
  schema:sameAs <http://www.wikidata.org/entity/Q34660> .
```

Note: the schema:sameAs property is used link to VIAF and WikiData *entities*, not the related pages.

Some other relevant properties to use for schema:Persons are:

```
schema:deathDate "{literal}"
schema:deathPlace "{literal}"
```

Metadata entities

Metadata entities, or entities describing properties of descriptions of 'ordinary' entities should be dual typed as a schema:Dataset and a schema:Webpage. The relation schema:mainEntityOfPage is used to link to the **Metadata** entity. The following provides an example of a metadata instance:

```
<http://data.bibliotheken.nl/doc/nbt/w1234567>
  a schema:WebPage, schema:Dataset ;
  schema:isPartOf <http://data.bibliotheken.nl/id/dataset/nbt> ;
  schema:license <https://creativecommons.org/publicdomain/zero/1.0/> ;
  schema:publisher <http://data.bibliotheken.nl/id/thes/p76543> ;
  schema:dateModified "2019-06-21"^^xsd:date ;
  schema:dateCreated "2019-05-01"^^xsd:date ;
```

[18] Dutch, in the sense of 'from or in the Netherlands'. In Belgium, Flemish family names are sorted using the full literal family name, including the prefix, for example "Van Dijk".
[19] PNV: https://w3id.org/pnv#

```
  schema:datePublished 2019-06-01"^^xsd:date .
```

Some notes

- The dataset in this example is part of a larger dataset (using `schema:isPartOf`). Please follow the NDE guidelines for publishing datasets[20] that extend beyond a single metadata entity.
- The property `schema:datePublished` reflect the date the entity was published for the first time.
- The property `schema:dateModified` relates to publication date of the current *version* of the entity at stake.
- The `schema:dateCreated` property also relates to the current *version* of the entity. A dataset might have been generated (created of modified; hence `schema:dateCreated`) before the updated version is published (`schema:dateModified`).

[20] https://github.com/netwerk-digitaal-erfgoed/project-organisations-datasets/tree/master/publication-model