

# Data Model for Lexicography (DMLex) Version 1.0

Committee Specification Draft ~~02~~03

12 ~~January~~June 2024

## Specification URIs

This version:

<https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd03/dmlex-v1.0-csd03.html>

<https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd03/dmlex-v1.0-csd03.pdf> (Authoritative)

Previous version:

<https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd02/dmlex-v1.0-csd02.html>

<https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd02/dmlex-v1.0-csd02.pdf> (Authoritative)

~~Previous version:~~

~~(Authoritative)~~

Latest version:

<https://docs.oasis-open.org/lexidma/dmlex/v1.0/dmlex-v1.0.html>

<https://docs.oasis-open.org/lexidma/dmlex/v1.0/dmlex-v1.0.pdf> (Authoritative)

Technical Committee:

[OASIS Lexicographic Infrastructure Data Model and API \(LEXIDMA\) TC](#)

Chair:

Michal Měchura ([michmech@mail.muni.cz](mailto:michmech@mail.muni.cz)), [Masaryk University](#)

Editors:

David Filip ([glorfindel@mail.muni.cz](mailto:glorfindel@mail.muni.cz)), [Masaryk University](#)

Miloš Jakubiček ([milos.jakubicek@sketchengine.eu](mailto:milos.jakubicek@sketchengine.eu)), [Lexical Computing](#)

[Vojtěch Kovář](#) ([vojcek@mail.muni.cz](mailto:vojcek@mail.muni.cz)), [Masaryk University](#)

Simon Krek ([simon.krek@ijs.si](mailto:simon.krek@ijs.si)), [Jozef Stefan Institute](#)

John McCrae ([john.mccrae@universityofgalway.ie](mailto:john.mccrae@universityofgalway.ie)), [University of Galway](#)

Michal Měchura ([michmech@mail.muni.cz](mailto:michmech@mail.muni.cz)), [Masaryk University](#)

Additional artifacts:

This prose specification is one component of a Work Product that also includes declarative validation artifacts accessible from <https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd03/schemas/>:

- XML: <https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd03/schemas/XML/>
- JSON: <https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd03/schemas/JSON/>

RDF: <https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd03/schemas/RDF/>~~NONE AT THE MOMENT~~

•

Informative copies of third party schemas are provided:

<https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd03/schemas/informativeCopiesOf3rdPartySchemas/>

## Related Work:

This specification is related to:

- *No related specifications.*

## Declared namespaces:

This specification declares one or more namespaces. Namespace isn't considered an XML specific feature in this serialization independent specification.

*The core and modules namespace*

- <http://docs.oasis-open.org/lexidma/ns/dmlex-1.0>

## Key words:

The key words MUST, MUST NOT, REQUIRED, SHALL, SHALL NOT, SHOULD, SHOULD NOT, RECOMMENDED, MAY, and OPTIONAL in this document are to be interpreted as described in [BCP 14 \[RFC2119\]](#) and [\[RFC8174\]](#) if, and only if, they appear in all capitals, as shown here.

## Abstract:

This document defines the 1st version of a data model in support of the high-priority technical goals described in the LEXIDMA TC's charter, including:

- A serialization-independent Data Model for Lexicography (DMLex)
- An XML serialization of DMLex
- A JSON serialization of DMLex
- A relational database serialization of DMLex
- An RDF serialization of DMLex
- An informative NVH serialization of DMLex

## Status:

This document was last revised or approved by the LEXIDMA TC on the above date. The level of approval is also listed above. Check the "Latest version" location noted above for possible later revisions of this document. Any other numbered Versions and other technical work produced by the Technical Committee (TC) are listed at [https://www.oasis-open.org/committees/tc\\_home.php?wg\\_abbrev=lexidma#technical](https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=lexidma#technical).

TC members should send comments on this document to the TC's email list. Others should send comments to the TC's public comment list, after subscribing to it by following the instructions at the "[Send A Comment](#)" button on the TC's web page at <https://www.oasis-open.org/committees/lexidma/>.

This specification is provided under the [Non-Assertion](#) Mode of the [OASIS IPR Policy](#), the mode chosen when the Technical Committee was established. For information on whether any patents have been disclosed that may be essential to implementing this specification, and any offers of patent licensing terms, please refer to the Intellectual Property Rights section of the TC's web page (<https://www.oasis-open.org/committees/lexidma/ipr.php>).

Note that any machine-readable content ([Computer Language Definitions](#)) declared Normative for this Work Product is provided in separate plain text files. In the event of a discrepancy between any such plain text file and display content in the Work Product's prose narrative document(s), the content in the separate plain text file prevails.

### Citation format:

When referencing this specification the following citation format should be used:

[DMLex-1.0]

*Data Model for Lexicography Version 1.0*. Edited by David Filip, Miloš Jakubiček, Simon Krek, John McCrae, and Michal Měchura. 12 ~~January~~ ~~June~~ 2024. OASIS ~~Committee~~ ~~Committee~~ Specification Draft 0203. <https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd03/dmlex-v1.0-csd03.html>. Latest version: <https://docs.oasis-open.org/lexidma/dmlex/v1.0/dmlex-v1.0.html>.

### Notices:

Copyright © OASIS Open 2024.

All Rights Reserved. Distributed under the terms of the OASIS [IPR Policy](#)).

The name "OASIS" is a trademark of [OASIS](#)), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs.

For complete copyright information please see the [full Notices](#) section in an Appendix below.

---

## Table of Contents

### [1 Introduction \(Normative\)](#)

- [1.1 Modular structure of DMLex](#)
- [1.2 Implementing DMLex](#)
- [1.3 The metamodel behind DMLex](#)
  - [1.3.1 Object types](#)
  - [1.3.2 Properties](#)
  - [1.3.3 Tree structures](#)
  - [1.3.4 Relational structures](#)
  - [1.3.5 Unique identifiers](#)
- [1.4 Definitions](#)

### [2 Conformance](#)

### [3 DMLex Core](#)

- [3.1 Optional roots](#)
- [3.2 Fragment identification](#)
  - [3.2.1 DMLex fragment identification strings](#)
- [3.3 lexicographicResource](#)
- [3.4 entry](#)
- [3.5 partOfSpeech](#)
- [3.6 inflectedForm](#)
- [3.7 sense](#)
- [3.8 definition](#)
- [3.9 label](#)
- [3.10 pronunciation](#)
- [3.11 transcription](#)
- [3.12 example](#)

- [4 DMLex OPTIONAL Modules \(Normative\)](#)
  - [4.1 DMLex Crosslingual Module](#)
    - [4.1.1 Extensions to lexicographicResource](#)
    - [4.1.2 translationLanguage](#)
    - [4.1.3 Extensions to sense](#)
    - [4.1.4 headwordTranslation](#)
    - [4.1.5 headwordExplanation](#)
    - [4.1.6 Extensions to example](#)
    - [4.1.7 exampleTranslation](#)
    - [4.1.8 Extensions to partOfSpeech](#)
    - [4.1.9 Extensions to label](#)
    - [4.1.10 Extensions to pronunciation](#)
    - [4.1.11 Extensions to inflectedForm](#)
  - [4.2 DMLex Controlled Values Module](#)
    - [4.2.1 Extensions to lexicographicResource](#)
    - [4.2.2 definitionTypeTag](#)
    - [4.2.3 inflectedFormTag](#)
    - [4.2.4 labelTag](#)
    - [4.2.5 labelTypeTag](#)
    - [4.2.6 partOfSpeechTag](#)
    - [4.2.7 sourceIdentityTag](#)
    - [4.2.8 transcriptionSchemeTag](#)
    - [4.2.9 sameAs](#)
  - [4.3 DMLex Linking Module](#)
    - [4.3.1 Extensions to lexicographicResource](#)
    - [4.3.2 relation](#)
    - [4.3.3 member](#)
    - [4.3.4 relationType](#)
    - [4.3.5 memberType](#)
    - [4.3.6 Extensions to sameAs](#)
  - [4.4 DMLex Annotation Module](#)
    - [4.4.1 Extensions to entry](#)
    - [4.4.2 Extensions to headwordTranslation](#)
    - [4.4.3 placeholderMarker](#)
    - [4.4.4 Extensions to definition](#)
    - [4.4.5 Extensions to example](#)
    - [4.4.6 Extensions to exampleTranslation](#)
    - [4.4.7 headwordMarker](#)
    - [4.4.8 collocateMarker](#)
    - [4.4.9 Extensions to label](#)
  - [4.5 DMLex Etymology Module](#)
    - [4.5.1 Extensions to entry](#)
    - [4.5.2 etymology](#)
    - [4.5.3 etymon](#)
    - [4.5.4 etymonUnit](#)
    - [4.5.5 Extensions to lexicographicResource](#)
    - [4.5.6 etymonType](#)
    - [4.5.7 etymonLanguage](#)
    - [4.5.8 Extensions to partOfSpeech](#)
- [5 DMLex REQUIRED Serializations \(Normative\)](#)
  - [5.1 DMLex XML serialization \(Normative\)](#)
    - [5.1.1 Design Principles \(Informative\)](#)
    - [5.1.2 Whitespace handling \(Normative\)](#)
    - [5.1.3 Element order \(Normative\)](#)
    - [5.1.4 DMLex XML serialization elements \(Normative\)](#)
  - [5.2 DMLex JSON serialization \(Normative\)](#)
    - [5.2.1 Design principles](#)
    - [5.2.2 DMLex JSON serialization objects \(Normative\)](#)

- [5.3 DMLex RDF serialization \(Normative\)](#)
- [5.3.1 Design principles](#)
- [5.3.2 DMLex RDF serialization elements](#)
- [5.4 DMLex relational database serialization \(Normative\)](#)
- [5.4.1 Design principles \(Informative\)](#)
- [5.4.2 Database diagrams \(Informative\)](#)
- [5.4.3 DMLex relational database serialization tables \(Normative\)](#)

## Appendixes

### [A Informative material on serializations \(Informative\)](#)

- [A.1 Examples \(Informative\)](#)
- [A.1.1 A basic entry](#)
- [A.1.2 How to use `inflectedForm`](#)
- [A.1.3 Pronunciation given as transcription](#)
- [A.1.4 Pronunciation given as a sound file](#)
- [A.1.5 Pronunciation given both ways](#)
- [A.1.6 How to use `partOfSpeechTag` and `inflectedFormTag`](#)
- [A.1.7 Mapping controlled values to external inventories](#)
- [A.1.8 Defining a bilingual lexicographic resource](#)
- [A.1.9 Defining a multilingual lexicographic resource](#)
- [A.1.10 How to use `headwordTranslation` in a bilingual lexicographic resource](#)
- [A.1.11 How to use `headwordTranslation` in a multilingual lexicographic resource](#)
- [A.1.12 How to use `headwordExplanation`](#)
- [A.1.13 Modelling parts and wholes](#)
- [A.1.14 Modelling antonyms](#)
- [A.1.15 Modelling synonyms](#)
- [A.1.16 Modelling variants](#)
- [A.1.17 Modelling subsenses](#)
- [A.1.18 Modelling subentries \(at subsense level\)](#)
- [A.1.19 Modelling subentries \(at sense level\)](#)
- [A.1.20 Using `placeholderMarker`](#)
- [A.1.21 Using `placeholderMarker` in a bilingual lexicographic resource](#)
- [A.1.22 Using `headwordMarker`](#)
- [A.1.23 Using `collocateMarker`](#)
- [A.1.24 Modelling etymology](#)
- [A.1.25 Modelling etymology types and language](#)
- [A.2 DMLex NVH serialization \(Informative\)](#)
- [A.2.1 Design Principles \(Informative\)](#)
- [A.2.2 DMLex NVH serialization nodes \(Normative\)](#)
- [A.2.2.1 NVH node: `lexicographicResource`](#)
- [A.2.2.2 NVH node: `entry`](#)
- [A.2.2.3 NVH node: `partOfSpeech`](#)
- [A.2.2.4 NVH node: `inflectedForm`](#)
- [A.2.2.5 NVH node: `sense`](#)
- [A.2.2.6 NVH node: `definition`](#)
- [A.2.2.7 NVH node: `label`](#)
- [A.2.2.8 NVH node: `pronunciation`](#)
- [A.2.2.9 NVH node: `transcription`](#)
- [A.2.2.10 NVH node: `example`](#)
- [A.2.2.11 NVH node: `translationLanguage`](#)
- [A.2.2.12 NVH node: `headwordTranslation`](#)
- [A.2.2.13 NVH node: `headwordExplanation`](#)
- [A.2.2.14 NVH node: `exampleTranslation`](#)
- [A.2.2.15 NVH node: `definitionTypeTag`](#)
- [A.2.2.16 NVH node: `inflectedFormTag`](#)
- [A.2.2.17 NVH node: `labelTag`](#)
- [A.2.2.18 NVH node: `labelTypeTag`](#)

[A.2.2.19 NVH node: partOfSpeechTag](#)  
[A.2.2.20 NVH node: sourceIdentityTag](#)  
[A.2.2.21 NVH node: transcriptionSchemeTag](#)  
[A.2.2.22 NVH node: sameAs](#)  
[A.2.2.23 NVH node: relation](#)  
[A.2.2.24 NVH node: member](#)  
[A.2.2.25 NVH node: relationType](#)  
[A.2.2.26 NVH node: memberType](#)  
[A.2.2.27 NVH node: placeholderMarker](#)  
[A.2.2.28 NVH node: headwordMarker](#)  
[A.2.2.29 NVH node: collocateMarker](#)  
[A.2.2.30 NVH node: etymology](#)  
[A.2.2.31 NVH node: etymon](#)  
[A.2.2.32 NVH node: etymonUnit](#)  
[A.2.2.33 NVH node: etymonType](#)  
[A.2.2.34 NVH node: etymonLanguage](#)

## [B References \(Normative\)](#)

[B.1 Normative references \(Normative\)](#)

[B.2 Informative references \(Informative\)](#)

## [C Machine Readable Validation Artifacts \(Normative\)](#)

## [D DMLex UML diagram \(Normative\)](#)

## [E DMLex plain text description \(Informative\)](#)

## [F Specification Change Tracking \(Informative\)](#)

[F.1 Tracking of changes made in response to Public Reviews](#)

[F.1.1 Tracking of changes in response to the 2nd Public Review](#)

[F.1.2 Tracking of changes in response to the 1st Public Review](#)

## [G Acknowledgements \(Informative\)](#)

## [H Notices \(Informative\)](#)

---

# 1 Introduction (Normative)

DMLex is a data model for modelling dictionaries (here called *lexicographic resources*) in computer applications such as dictionary writing systems.

DMLex is a data model, not an encoding format. DMLex is abstract, independent of any markup language or formalism. At the same time, DMLex has been designed to be easily and straightforwardly implementable in XML, JSON, NVH, as a relational database, and as a Semantic Web triplestore.

## 1.1 Modular structure of DMLex

The DMLex specification is divided into a REQUIRED core with several OPTIONAL modules.

- [DMLex Core](#) allows you to model the basic entries-and-sense structure of a monolingual lexicographic resource.
- [DMLex Crosslingual Module](#) extends DMLex Core to model bilingual and multilingual lexicographic resources.
- [DMLex Controlled Values Module](#) extends DMLex Core to represent inventories of look-up values to be used as part-of-speech tags, usage label tags and others.
- [DMLex Linking Module](#) extends DMLex Core and allows you to model various kinds of relations between entries, senses and other objects, including semantic relations such as synonymy and antonymy and presentational relations such as subentries and subsenses, both within a single lexicographic resource and across multiple lexicographic resources.
- [DMLex Annotation Module](#) extends DMLex Core to allow the modelling of inline markup on various objects such as example sentences, including the modelling of collocations and corpus patterns.

- [DMLex Etymology Module](#) extends DMLex Core to allow the modelling of etymological information in dictionaries.

## 1.2 Implementing DMLex

DMLex is an abstract data model which can be implemented in many different programming environments and serialization languages. In this document, we specify REQUIRED serializations for:

[XML](#),  
[JSON](#),  
[RDF](#), and  
[relational databases](#).

An informative serialization specification is provided for:

[NVH](#).

## 1.3 The metamodel behind DMLex

The DMLex data model is defined in this document through the following concepts.

### 1.3.1 Object types

DMLex defines the existence of certain *object types* such as `entry` and `definition`. In every DMLex-conformant data set, each “object” (for example each XML element or each database table) must be an instance of one of these object types.

The names of object types are unique within the scope of the entire data model, including all the modules. An object type defined in one module is guaranteed not to name-conflict with another object type in another module.

### 1.3.2 Properties

For every object type, DMLex defines which properties it can have, whether the properties are required or optional, what their arities are (for example “zero or more”), and what kinds of values it can contain.

There are two kinds of properties: those that contain literal values and those that contain objects.

Some properties are defined to contain literal values such as strings and numbers. The following types of literal values are used in DMLex:

- normalised string: a string that contains no new lines, does not start or end with a whitespace, contains no block of ASCII whitespace more than a single space and is non-empty.
- number: including negative numbers and floating-point numbers
- non-negative integer number
- boolean: a true/false value
- URI
- IETF language code
  - IRI

non-empty string,  
number,

unique identifier  
and reference to a unique identifier

- [: an Internationalized Resource Identifier](#)
- [closed list of possible values](#)

Some properties are defined to contain objects (of types defined in DMLex), for example an object of type `entry` can contain objects of type `sense`. In such cases, the name of the property is the same as the name of the object type. For example, the `entry` object type has a property called `sense` which (with an arity of “zero or more”) contains `sense` objects.

The names of properties are unique within the scope of the object type they belong to but not necessarily outside it.

### 1.3.3 Tree structures

The mechanics of objects which (through their properties) contain other objects mimics the tree-like structure commonly used in lexicography to represent dictionary entries, typically using XML.

Because of this, this document sometimes refers to objects being the “children”, “parents”, “descendants” or “ancestors” of other objects, or objects “belonging to” other objects. This is to be understood as follows.

- “Object A is the parent of object B” and “object B is a child of object A” both mean that object A has a property whose value is object B.
- “Object A is an ancestor of object B” and “object B is a descendant of object A” means that there is a chain of parent-child links between object A and object B.
- “Object B belongs to object A” is the same as “object B is a descendant of object A” and “object A is an ancestor of object B”.

The tree structures created by DMLex are guaranteed to have similar properties as in XML and many other tree-structured formalisms, including importantly the constraint that every object has no more than one parent.

Additionally, tree structures created by DMLex are guaranteed to be non-recursive: an object can never be an ancestor of another object of the same type.

### 1.3.4 Relational structures

In addition to tree structures, which are used in DMLex to model the basic entries-and-senses skeleton of a lexicographic resource, DMLex also allows (through the `relation` object type defined in the Linking Module) the modelling of non-tree relations between objects, that is, relations connecting objects which belong to different parents.

Non-tree relations are used in DMLex to model cross-references between entries and senses, both within a single lexicographic resource and across multiple lexicographic resources. Non-tree relations are also used in DMLex to model certain things which have traditionally been modelled with tree structures in lexicography, such as the hierarchical arrangement of entries and subentries, or senses and subsenses. In DMLex, these phenomena are modelled as non-tree relations in order to avoid recursion in the tree structures and to minimize the number of object types in the data model.

### 1.3.5 Unique identifiers

Some object types defined in DMLex have a property or a combination of properties mandated to be unique. Designating a property as unique means that its value must be unique within the scope of its parent object



type. Designating a combination of properties as unique means that the ordered composition of its values must be unique within the scope of their parent object type. In the case that all unique properties are missing, listingOrder MUST be used to verify uniqueness. Such unique properties in DMLex serve two purposes:

- To enable objects to participate in relational structures. Relational structures (as opposed to tree structures) are modelled in DMLex as tuples (triples, ...) of unique identifiers.
- To enable addressing and accessing individual objects, for example individual senses or individual example sentences, outside the context of the entries they are in. See [Fragment identification](#) for details.

Serializations of DMLex may choose to assign arbitrary unique identifiers (e.g. a random integer or a hash of the property value) to each property or combination of properties.

## 1.4 Definitions

### Agent

any application or tool that , reads, edits, writes, processes, stores, renders or otherwise handles *DMLex Instances*.

*Agent* is the most general application conformance target that subsumes all other specialized user agents disregarding whether they are defined in this specification or not.

### Enrich, Enriching

the process of adding metadata and resources to *DMLex Instances*

*Processing Requirements*

- *Enriching* MAY happen at the time of *Creation*.

### Enricher, Enricher Agent

any *Agent* that performs the *Enriching* process

### Create, Creation

the process of encoding lexicographic data as *DMLex Instances*

### Creator, Creator Agent

any *Agent* that performs the *Creation* process

### Modify, Modification

the process of changing core and module based structural and inline objects that were previously created by other *Writers*

*Processing Requirements*

- DMLex Instances MAY be *Modified* and *Enriched* at the same time.

### Note

*Creator* or *Enricher* knowledge of the *DMLex Instance* is not assumed while *Modifying*. However, modifications can only happen in the same serialization.

### **Modifier, Modifier Agent**

an *Agent* that performs the *Modification* process

### **Writer, Writer Agent**

an *Agent* that *Creates*, generates, or otherwise writes *DMLex Instances* for whatever purpose, including but not limited to *Creator*, *Modifier*, and *Enricher Agents*.

### **Note**

Since DMLex serializations are intended as editing and exchange formats rather than processing formats, many applications will need to generate *DMLex Instances* from their internal processing formats, even in cases when they are processing *DMLex Instances* provided by other *Writers*.

### **UNIQUE**

The term UNIQUE shall be interpreted as given in the section on [Unique identifiers](#).

## **2 Conformance**

1. *DMLex Instances Conformance*
  - a. Conformant *DMLex Instances* MUST be well formed and valid instances according to one of the normative DMLex Serialization Specifications.
  - b. *DMLex Instances* MAY contain custom extensions, provided that those extensions do not provide features or parts of features that are provided by the [DMLex Core](#) or any of the [DMLex Modules](#). Extensions MUST be serialized in a way conformant and compatible with the pertaining DMLex Serialization Specifications.
2. *Application Conformance*
  - a. DMLex *Writers* MUST create conformant *DMLex Instances* to be considered DMLex compliant.
  - b. *Agents* processing conformant *DMLex Instances* that contain custom extensions are not REQUIRED to understand and process non-DMLex objects or attributes. However, conformant applications SHOULD preserve existing custom extensions when processing conformant *DMLex Instances*, provided that the objects that contain custom extensions are not removed according to DMLex Processing Requirements or the extension's own processing requirements.
  - c. All *Agents* MUST comply with Processing Requirements for otherwise unspecified *Agents* or without a specifically set target *Agent*.
  - d. Specialized *Agents* defined in this specification - this is *Creator*, *Writer*, *Modifier*, and *Enricher Agents* - MUST comply with the Processing Requirements targeting their specifically defined type of *Agent* on top of Processing Requirements targeting all *Agents* as per point c. above.
  - e. DMLex is an object model explicitly designed for exchanging data in the same serialization among various *Agents*. Thus, a conformant DMLex application MUST be able to accept *DMLex Instances Created*, *Modified*, or *Enriched* by a different application, provided that:
    - i. The processed files are conformant *DMLex Instances* according to the same DMLex Serialization Specification,
    - ii. in a state compliant with all relevant Processing Requirements.
3. *Backwards Compatibility*
  - a. N/A (no previous versions)

### **Note**

*DMLex Instances* cannot be conformant to this specification w/o being conformant to a specific serialization.

## Warning

Exchange of *DMLex Instances* data between and among DMLex serializations is out of scope of the DMLex Version 1.0 specification. Although, the serializations are semantically compatible, they don't have to ~~posses~~**possess** compatible addressing and ~~fragment~~**fragment** identification mechanisms, mainly due to the fact that each serialization is expected to use its own native id data types, uniqueness scopes, and addressing methods.

## 3 DMLex Core

The DMLex Core provides data types for modelling monolingual dictionaries (called *lexicographic resources* in DMLex) where headwords, definitions and examples are all in one and the same language. DMLex Core gives you the tools you need to model simple dictionary entries which consist of the following data types:

- [lexicographicResource](#)
- [entry](#)
- [partOfSpeech](#)
- [inflectedForm](#)
- [sense](#)
- [definition](#)
- [label](#)
- [pronunciation](#)
- [transcription](#)
- [example](#)

### 3.1 Optional roots

When exchanging data encoded in a DMLex serialization which has the concept of a "root" or top-level object, such as XML, JSON or NVH, the object types `lexicographicResource` and `entry` can serve as such roots.

### 3.2 Fragment identification

Incomplete parts of DMLex objects represent valid fragments as long as it is possible to identify their complete source DMLex object. For a particular serialization, this can be achieved by:

- Making sure the fragment contains all [unique properties](#) of the DMLex parent-child hierarchy that are necessary to determine its source object.
- Using a serialization-specific addressing mechanism (e.g. XPath for XML, JMESPath for JSON or an NVH query) to provide the location of the fragment in the source DMLex object.

#### 3.2.1 DMLex fragment identification strings

DMLex provides a recommended method for addressing DMLex objects present on-line, useful for linking (cf. Section 4.3, "DMLex Linking Module") and general interoperability. Implementing this method is not REQUIRED for conformance.

Every fragment SHOULD be assigned a unique fragment identification string, composed of `lexicographicResource.uri`, with protocol identification prefix (such as `http://` or `https://`)

removed, and a sequence of identifiers that uniquely determines the path in the DMLex tree structure. The DMLex fragment identification string of the root object `lexicographicResource` is the value of its attribute `lexicographicResource.uri`, with protocol identification prefix (such as `http://` or `https://`) removed. The fragment identification strings of its direct children are constructed as follows:

`lexicographicResource.uri/objectTypeName/objectID`

(We define below how object IDs are created.)

The DMLex fragment identification strings of descendant objects are constructed by appending the children's type names and IDs to the fragment identification strings of their direct parents, using "/" as the delimiter. In other words, the full template for a fragment identification string looks as follows:

`lexicographicResource.uri/objectTypeName/objectID/child1TypeName/child1ID/child2TypeName/child2ID/...`

For example, a particular `sense` (which is a property of `entry`) is assigned the following fragment identification string:

`lexicographicResource.uri/entry/entryID/sense/senseID`

A fragment identification string of an `example` (which is a property of `sense`, which is a property of `entry`) has the following structure:

`lexicographicResource.uri/entry/entryID/sense/senseID/example/exampleID`

### **3.2.1.1 Object IDs**

For the purpose of creating DMLex fragment identification strings, each object is assigned a unique ID relative to its parent, based on values of its properties declared as `UNIQUE`. Multiple situations can occur:

1. The object type has a single `UNIQUE` property with an arity of "exactly one", and the value of the property is a string or a number. In this case, the object ID is the string or the number, with the following modifications performed in that particular order:
  - every `"\"` (ASCII character 5C) is replaced by `"\"`
  - every `"~"` (ASCII character 7E) is replaced by `"~"`
  - every `" "` (ASCII character 5F) is replaced by `"\"`
  - every `"0"` (zero, ASCII character 30) is replaced by `"\0"`
  - all IRI-unsafe characters (outside the `inreserved` class according to [RFC 3987]) are percent-encoded according to [RFC 3986~~3-4~~]
2. The object type has a single `UNIQUE` property with an arity of "exactly one", and the value of the property is a child DMLex object. In this case, the object ID is the same as the object ID of the child object. (Note: this case actually does not occur in the specification as such; we list it here to streamline the description of the following cases.)
3. The object type has a single `UNIQUE` property with an arbitrary arity. In this case, all the partial single values or child object IDs are constructed according to the steps 1. and 2., and the resulting object ID is their concatenation using `" "` (ASCII character 5F) as a separator. The order of the partial values is driven by the `listingOrder` of the respective objects. If this procedure returns an empty string (which can happen in case of `UNIQUE` attributes that allow the arity of zero), the string `"0"` (zero, ASCII character 30) is used instead of the empty string.
4. The object type has multiple `UNIQUE` properties. In this case, all the partial values or child object IDs are constructed according to the steps 1., 2. and 3., and the resulting object ID is their concatenation using `"~"` (ASCII character 5F) as a separator. The order of the partial values is driven by the order of the properties as given in this specification. (Note: all attributes marked as `UNIQUE` need to be represented in the ID, as empty values are replaced by `"0"` according to step 3. No empty IDs are allowed.)

5. In specific situations it may happen there are multiple different objects with all the UNIQUE properties empty, i.e. multiple objects with duplicate IDs (the same sequence of zeros) emerge as the result of the step 4. One example of such a situation is multiple senses without indicators or definitions, but with different translations. In that case, and only in that case, the value of listingOrder is concatenated to the sequence of zeros, to distinguish between the duplicate IDs. If there is only one such object, listingOrder is not concatenated to the sequence of zeros.

DMLex does not define the structure of DMLex fragment identification strings for object types without UNIQUE properties.

### **3.2.1.2 DMLex fragment identification string examples**

Particular examples of DMLex fragment identification strings can then look as follows:

- www.example.com/lexicon/entry/cat~1~noun
- www.example.com/lexicon/entry/cat~1~noun/sense/0~small%20furry%20animal (Here we assume that the sense's indicator is empty and it has one definition which says "small furry animal").
- www.example.com/lexicon/entry/cat~1~noun/sense/0~small%20furry%20animal/example/I%20have%20two%20dogs%20and%20a%20cat.
- www.example.com/lexicon/entry/cat~1~noun/sense/0~0 (Here we assume that both the sense's definition and its indicator are empty, and there is only one such sense.)
- www.example.com/lexicon/entry/cat~1~noun/sense/0~02 (Here we assume that both the sense's definition and its indicator are empty, there are multiple such senses, and this is the sense number 2, of all this entry's senses.)

## **3.3 lexicographicResource**

Represents a dictionary. A lexicographic resource is a dataset which can be used, viewed and read by humans as a dictionary and – simultaneously – ingested, processed and understood by software agents as a machine-readable dataset.

### **Note**

The correct name of this data type in DMLex is *lexicographic*, not *lexical*, resource.

### *Properties*

- title OPTIONAL (zero or one). ~~Non-empty~~ **Normalised** string. A human-readable title of the lexicographic resource.
- uri **REQUIRED** ~~OPTIONAL~~ (zero or one). The URI of the lexicographic resource, identifying it on the Web.
- langCode REQUIRED (exactly one). The IETF language code of the language that this lexicographic resource describes.
- [entry](#) OPTIONAL (zero or more)

### *Comments*

- langCode identifies the language of headwords, definitions and examples in this dictionary. DMLex is based on the assumption that all headwords in a lexicographic resource are in the same language, and that definitions and examples, if any are included in the lexicographic resource, are

in that language too. The `langCode` property of `lexicographicResource` informs potential users of the lexicographic resource which language that is.

- Ideally, a lexicographic resource should include at least one entry. However, DMLex specifies that `entry` is optional in `lexicographicResource` to allow for the existence of lexicographic resources which are not yet complete.
- The `lexicographicResource` data type does not prescribe properties for detailed metadata about the lexicographic resource, implementers are advised to refer to the properties in the [Dublin Core elements namespace](#) as for basic metadata.
- DMLex allows lexicographic resources to contain zero entries. This is to enable the existence of lexicographic resources which are work in progress, and lexicographic resources whose only purpose is to contain relations (defined in the [DMLex Linking Module](#)) that link entries from other lexicographic resources.

#### Example 1. XML

```
<lexicographicResource uri="..." langCode="...">
  <title>...</title>
  <entry.../>
</lexicographicResource>
```

#### Example 2. JSON

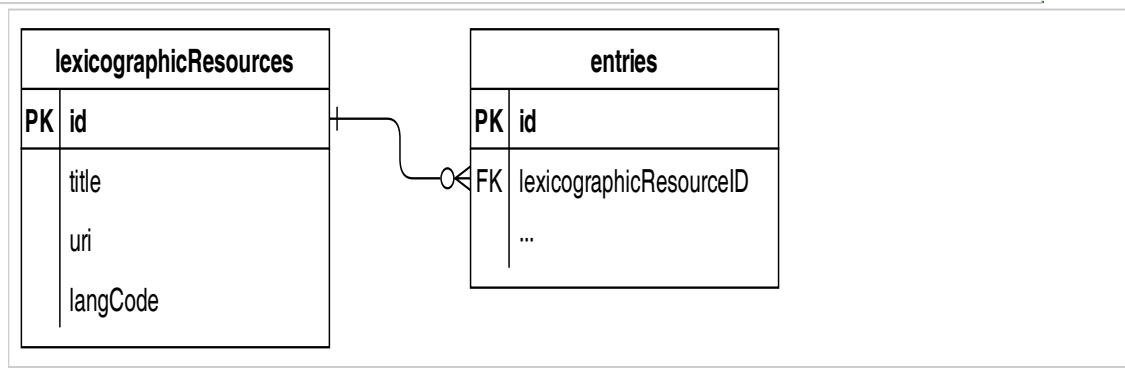
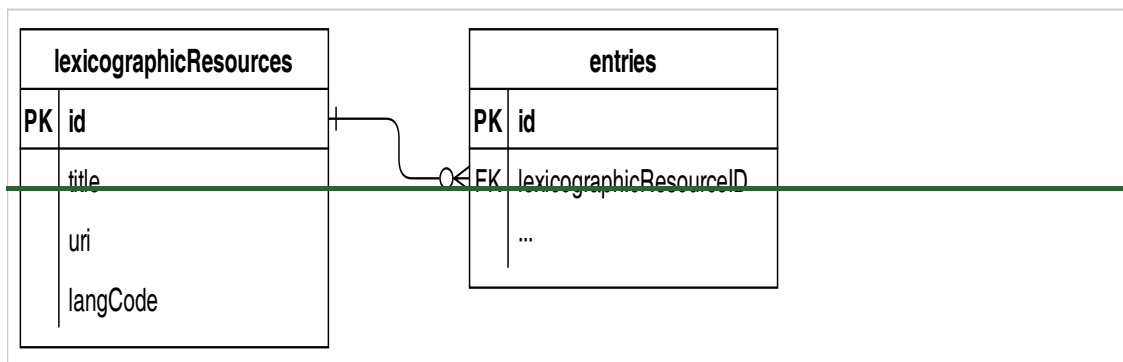
```
{
  "title": "...",
  "langCode": "...",
  "entries": [...]
}
```

#### Example 3. RDF

```
@prefix dmlex: <http://wwwhttps://docs.oasis-open.org/to-be-confirmedlexidma/dmlex>/v1.0/schemas/RDF/dmlex.ttl#> .

<#id> a dmlex:LexicographicResource ;
  dmlex:title "...";
  dmlex:uri "...";
  dmlex:langCode "...";
  dmlex:entry <entry1> , ... .
```

#### Example 4. Relational database



### 3.24 entry

Represents a dictionary entry. An entry contains information about one headword.

*Property of*

- [lexicographicResource](#)

*Properties*

- **headword** REQUIRED (exactly one) and UNIQUE (in combination with other unique properties if present). ~~Non-empty~~ **Normalised** string. The entry's headword.
- **homographNumber** OPTIONAL (zero or one) and UNIQUE (in combination with other unique properties if present). **Number**. The entry's homograph number, as a guide to distinguish entries with the same headword.
- **partOfSpeech** OPTIONAL (zero or more) and UNIQUE (in combination with other unique properties if present).
- **label** OPTIONAL (zero or more).
- **pronunciation** OPTIONAL (zero or more).
- **inflectedForm** OPTIONAL (zero or more).
- **sense** OPTIONAL (zero or more).

#### Note

DMLex Core does not have a concept of "subentry". To model subentries (i.e. entries inside entries) in a lexicographic resource, object types from the Linking Module should be used.

#### Note

The headword can be a single word, a multi-word expression, or any expression in the source language which is being described by the entry.

### Note

DMLex allows only one headword per entry. Things such as variant headwords do not exist in DMLex. However, the [DMLex Linking Module](#) does make it possible to represent the existence of variants by treating them as separate headwords of separate entries, and linking the entries using a type of link which will cause the entries to be placed together when shown to human users. See [Section A.1.16, "Modelling variants"](#) for an example using the English words "colour" and "color".

### Note

DMLex allows entries to contain zero senses. An entry with zero senses is an entry which declares the existence of its headword and, optionally, gives some information about the headword's morphosyntactical properties (part of speech, inflected forms etc.) but does not say anything about its semantics, about its meaning. Such entries can exist as entries which are work in progress, or as entries whose only purpose is to serve as members in entry-to-entry relations (defined in the [DMLex Linking Module](#)) such as spelling variants (example: [Section A.1.16, "Modelling variants"](#)).

### Note

Entries in DMLex do not have an explicit listing order. An application can imply a listing order from a combination of the headword and the homograph number, such that the headword is the primary sorting key and the homograph number (for entries that have one) is the secondary sorting key.

### Note

Ideally, each entry should have exactly one part-of-speech label. However, DMLex allows more than one `partOfSpeech` in `entry` in order to allow for exceptional cases when the lexicographer has decided to treat multiple part-of-speech readings of a headword in a single entry. Example: English words which denote nationalities ("Czech", "German") and which can function both as nouns and as adjectives.

#### Example 5. XML

```
<entry homographNumber="...">
  <headword>...</headword>
  <partOfSpeech.../>
  <label.../>
  <pronunciation.../>
  <inflectedForm.../>
  <sense.../>
</entry>
```

#### Example 6. JSON

```
{
  "headword": "...",
  "homographNumber": "...",
```



```

"partsOfSpeech": [...],
"labels": [...],
"pronunciations": [...],
"inflectedForms": [...],
"senses": [...]
}

```

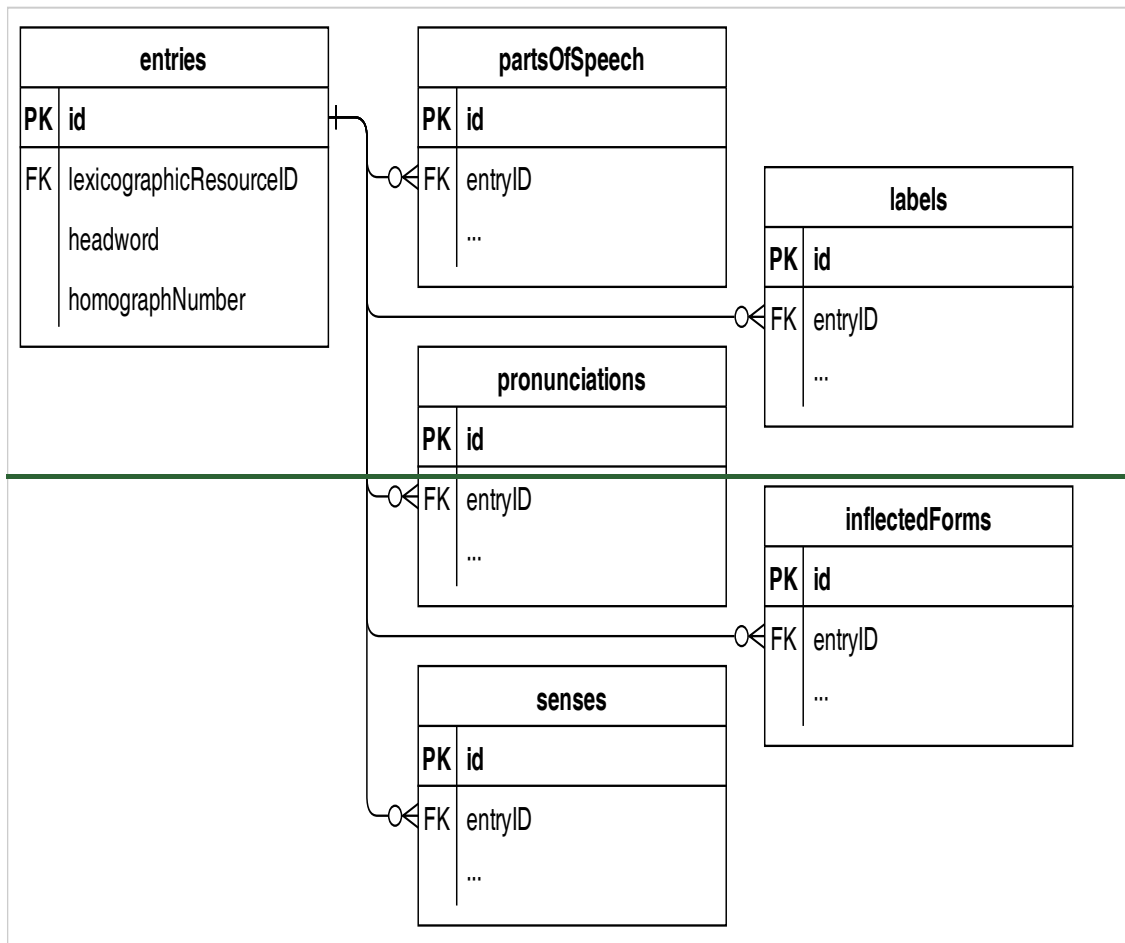
**Example 7. RDF**

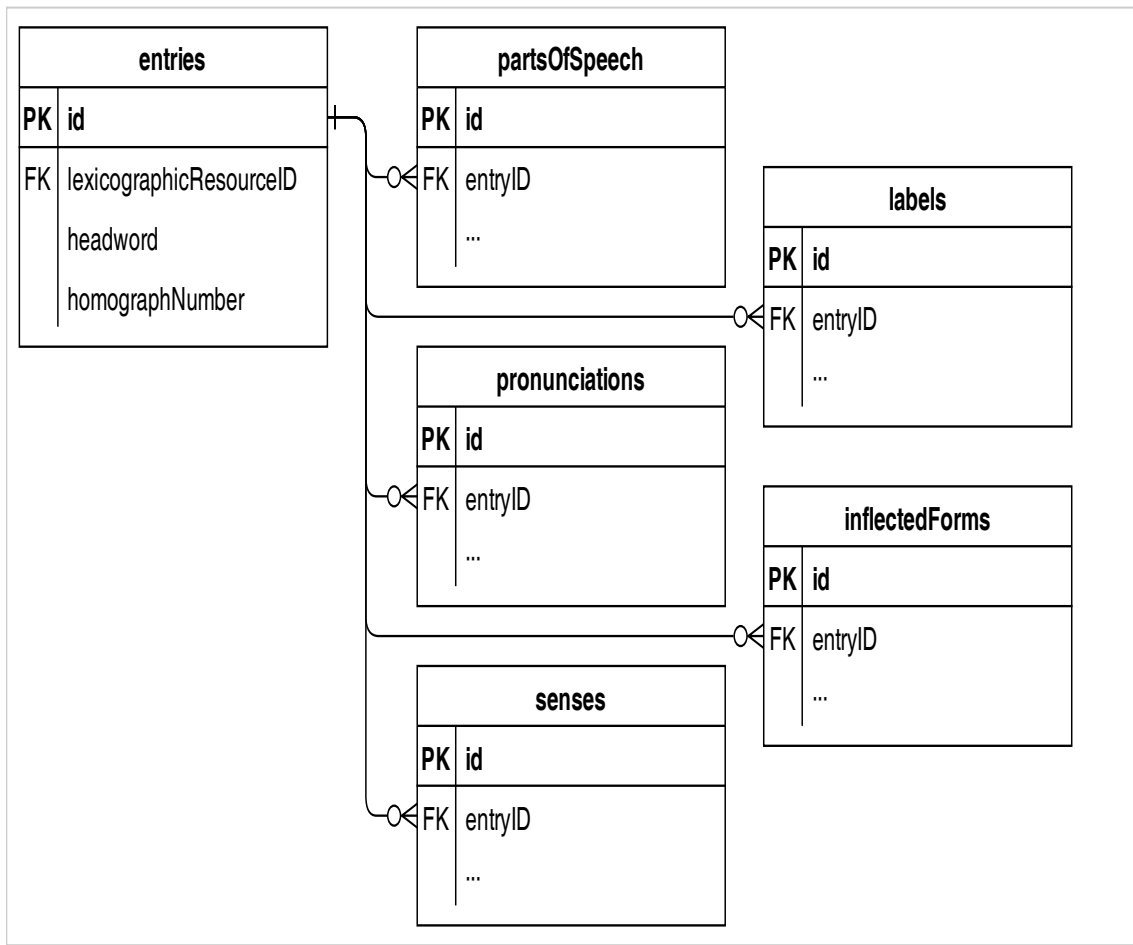
```

<id> a dmlex:Entry ;
  dmlex:headword "...";
  dmlex:homographNumber ... ;
  dmlex:partOfSpeech ... ;
  dmlex:label ... ;
  dmlex:pronunciation ... ;
  dmlex:inflectedForm ... ;
  dmlex:sense ... .

```

**Example 8. Relational database**





### 3.35 `partOfSpeech`

Represents a part-of-speech label.

Property of

- [entry](#)

Properties

- `tag` REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ **Normalised** string. An abbreviation, a code or some other string of text which identifies the part-of-speech label, for example `n` for noun, `v` for verb, `adj` for adjective. The `partOfSpeechTag` object type can be used to explain the meaning of the part-of-speech tags, to constrain which part-of-speech tags are allowed to occur in the lexicographic resource, and to map them onto external inventories and ontologies.
- `listingOrder` REQUIRED (exactly one). Number. The position of this part-of-speech label among other part-of-speech labels of the same entry. This can be implicit from the serialization.

Comments

- The way to model other grammatical properties of the headword besides part of speech, such as gender (of nouns) or aspect (of verbs) in DMLex is to combine them with the part of speech into a

single part-of-speech tag, for example noun-masc and noun-fem, or v-perf and v-imperf.

*Example 9. XML*

```
<partOfSpeech tag="..."/>
```

*Example 10. JSON*

Implemented as a string

```
"..."
```

*Example 11. RDF*

```
<entry> dmlex:partOfSpeech [  
  a dmlex:PartOfSpeech ;  
  dmlex:tag "...";  
  dmlex:listingOrder 1 ] .
```

*Example 12. Relational database*

partsOfSpeech	
PK	id
FK	entryID
	tag
	listingOrder

partsOfSpeech	
PK	id
FK	entryID
	tag
	listingOrder

Represents one (of possibly many) inflected forms of the headword. Example: [Section A.1.2, “How to use `inflectedForm`”](#).

*Property of*

- [entry](#)

*Properties*

- `text` REQUIRED (exactly one) and UNIQUE (in combination with other unique properties if present). ~~Non-empty~~ **Normalised** string. The text of the inflected form.
- `tag` OPTIONAL (zero or one) and UNIQUE (in combination with other unique properties if present). ~~Non-empty~~ **Normalised** string. An abbreviation, a code or some other string of text which identifies the inflected form, for example `pl` for plural, `gs` for genitive singular, `com` for comparative. The `inflectedFormTag` object type can be used to explain the meaning of the inflection tags, to constrain which inflection tags are allowed to occur in the lexicographic resource, and to map them onto external inventories and ontologies.
- `label` OPTIONAL (zero or more).
- `pronunciation` OPTIONAL (zero or more).
- `listingOrder` REQUIRED (exactly one). Number. The position of this inflected form among other inflected forms of the same entry. This can be implicit from the serialization.

*Example 13. XML*

```
<inflectedForm tag="...">
  <text>...</text>
  <label.../>
  <pronunciation.../>
</inflectedForm>
```

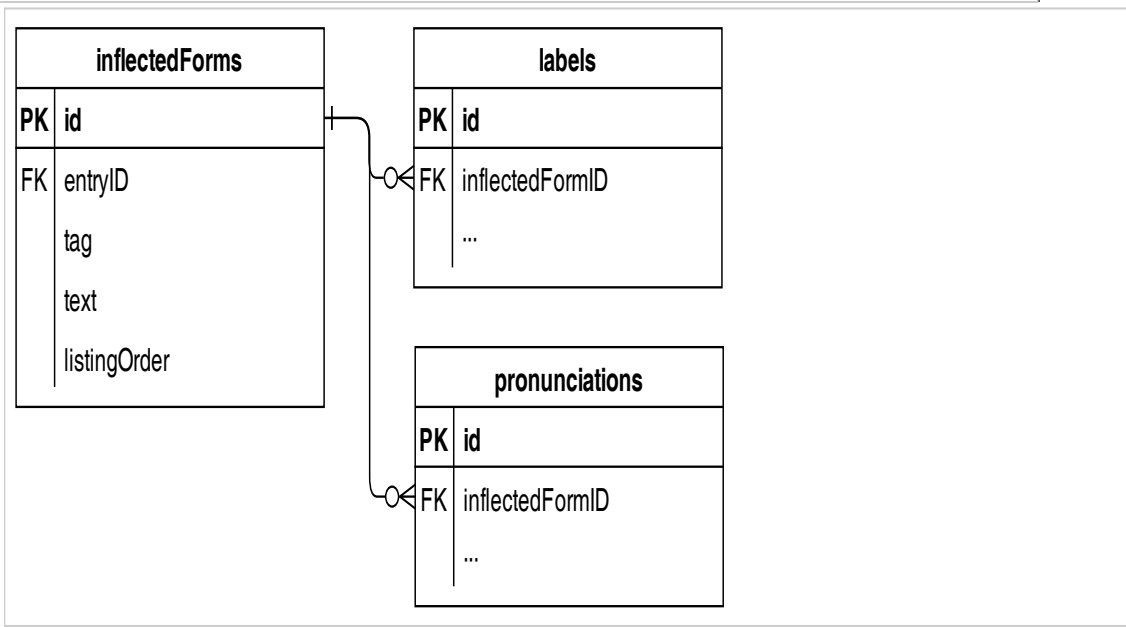
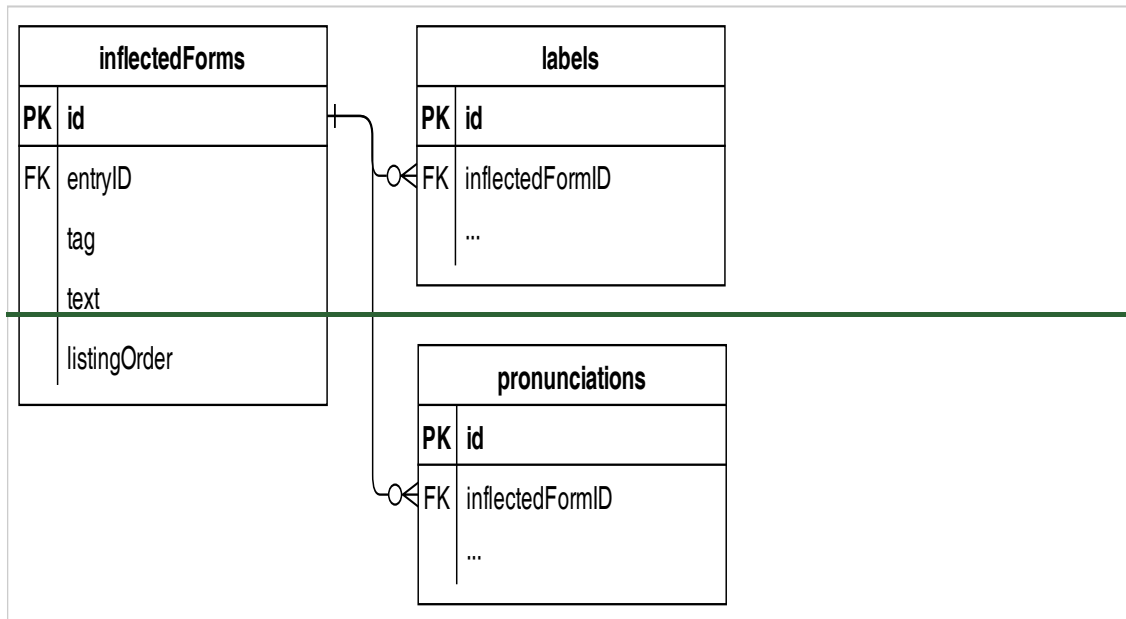
*Example 14. JSON*

```
{
  "tag": "...",
  "text": "...",
  "labels": [...],
  "pronunciations": [...]
}
```

*Example 15. RDF*

```
<entry> dmlex:inflectedForm [
  dmlex:text "...";
  dmlex:tag "...";
  dmlex:listingOrder 1;
  dmlex:label ...;
  dmlex:pronunciation ... ] .
```

Example 16. Relational database



Comments

- The `inflectedForm` object type is intended to model the **inflectional morphology** of a headword. To model derivational morphology, for example feminine forms of masculine nouns, the recommended way to do that in DMLex is to create separate entries for the two words, and link them using the [Linking Module](#).

3.57 sense

Represents one of possibly many meanings (or meaning potentials) of the headword.

### Property of

- [entry](#)

### Properties

- `listingOrder` REQUIRED (exactly one). Number. The position of this sense among other senses of the same entry. ~~Can~~ [This can](#) be implicit from the serialization.
- `indicator` OPTIONAL (zero or one) and UNIQUE. [Normalised string](#). A short statement, in the same language as the headword, that gives an indication of the meaning of a sense and permits its differentiation from other senses in the entry. Indicators are sometimes used in dictionaries instead of or in addition to definitions.
- `label` OPTIONAL (zero or more).
- `definition` OPTIONAL (zero or more) and UNIQUE.
- `example` OPTIONAL (zero or more).

### Comments

- The properties of **entry** are, apart from `sense`, formal properties of the headword such as orthography, morphology, syntax and pronunciation. A **sense** is a container for statements about the headword's semantics. DMLex deliberately makes it impossible to include morphological information at sense level. It is impossible in DMLex to model an entry where each sense has slightly different morphological properties (e.g. a noun has a weak plural in one sense and a strong plural in another). Such phenomena need to be treated as two entries (homographs) and can be linked using the Linking Module to make sure they are always shown together to human users.

### Example 17. XML

```
<sense>
  <indicator>...</indicator>
  <label.../>
  <definition.../>
  <example.../>
</sense>
```

### Example 18. JSON

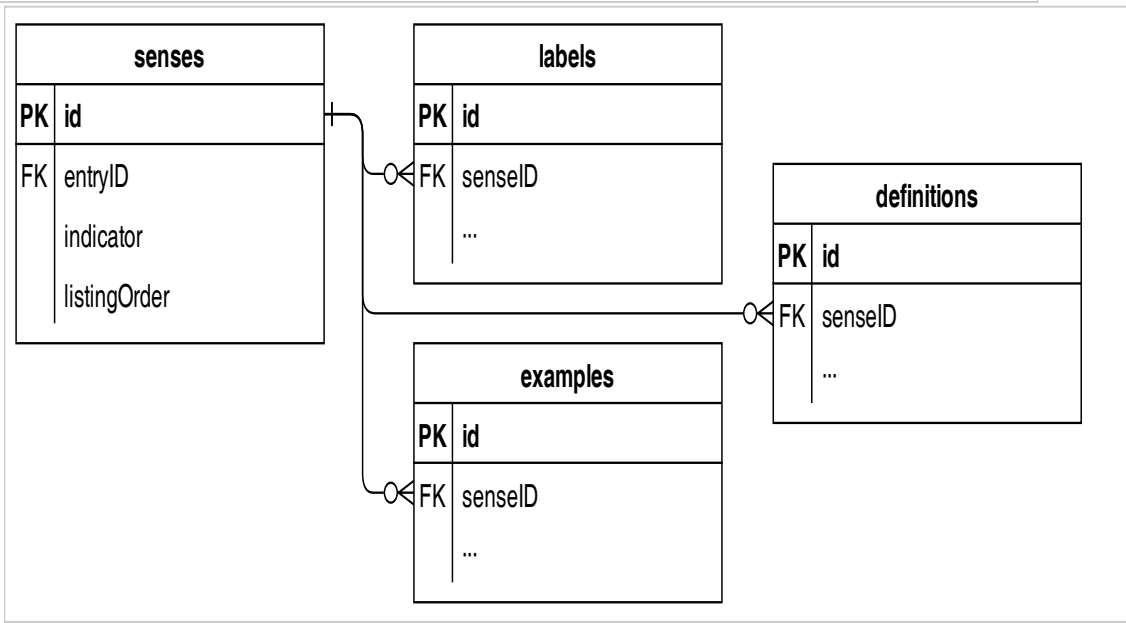
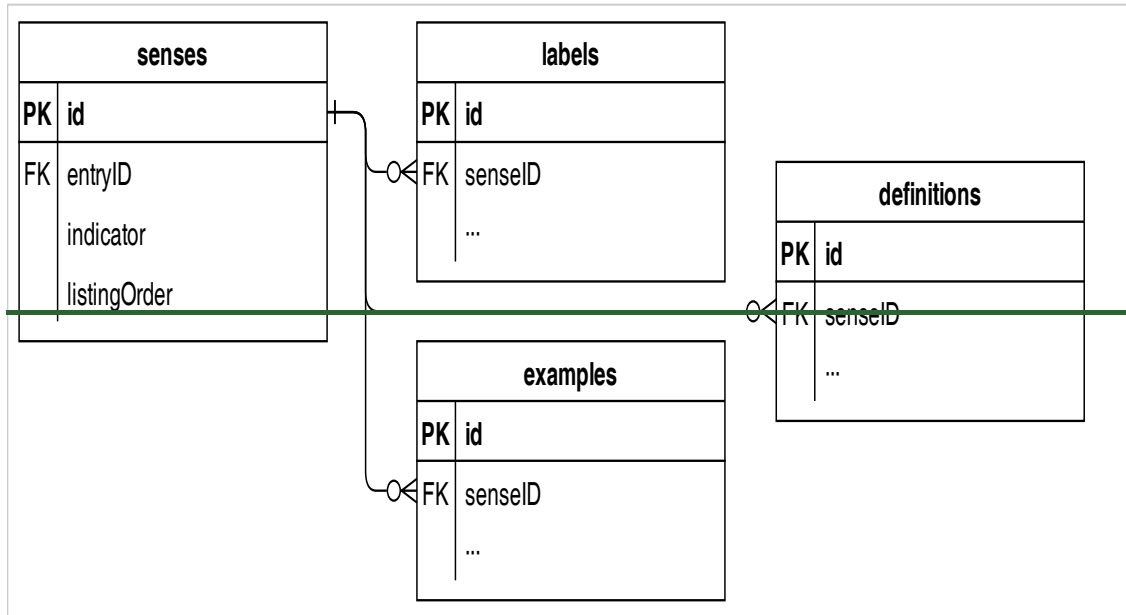
```
{
  "indicator": "...",
  "labels": [...],
  "definitions": [...],
  "examples": [...]
}
```

### Example 19. RDF

```
<id> a dmlex:Sense ;
  dmlex:listingOrder 1 ;
  dmlex:indicator "...";
```

```
dmlex:label ... ;
dmlex:definition ... ;
dmlex:example ... .
```

Example 20. Relational database



### 3.68 definition

Represents one of possibly several definitions of a sense.

Property of

- [sense](#)

#### Properties

- text REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ **Normalised** string. A statement, in the same language as the headword, that describes and/or explains the meaning of a sense. In DMLex, the term definition encompasses not only formal definitions, but also less formal explanations.
- definitionType OPTIONAL (zero or one). **Normalised string**. If a sense contains multiple definitions, indicates the difference between them, for example that they are intended for different audiences. The [definitionTypeTag](#) object type can be used to constrain and/or explain the definition types that occur in the lexicographic resource.
- listingOrder REQUIRED (exactly one). Number. The position of this definition among other definitions of the same sense. This can be implicit from the serialization.

#### Example 21. XML

```
<definition definitionType="...">...</="...">
  <text>...</text>
</definition>
```

#### Example 22. JSON

```
{
  "text": "...",
  "definitionType": "..."
}
```

#### Example 23. RDF

```
<sense> dmlex:definition [
  a dmlex:Definition ;
  dmlex:text "...";
  dmlex:definitionType "...";
  dmlex:listingOrder 1 ] .
```

#### Example 24. Relational database



definitions	
PK	id
FK	senseID
	text
	definitionType
	listingOrder

definitions	
PK	id
FK	senseID
	text
	definitionType
	listingOrder

### 3.7.9 label

Represents a restriction on its parent such as temporal (old-fashioned, neologism), regional (dialect), register (formal, colloquial), domain (medicine, politics) or grammar (singular-only).

*Property of*

- [entry](#)
- [sense](#)
- [inflectedForm](#)
- [pronunciation](#)
- [example](#)

*Properties*

- `tag` REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ **Normalised** string. An abbreviation, a code or some other string of text which identifies the label, for example `neo` for neologism, `colloq` for colloquial, `polit` for politics. The [labelTag](#) object type can be used to explain the meaning of the labels, to constrain which labels are allowed to occur in the lexicographic resource, and to map them onto external inventories and ontologies.
- `listingOrder` REQUIRED (exactly one). Number. The position of this label among other labels of the same entry. This can be implicit from the serialization.

*Comments*

- A label applies to the object that it is a child of. When the label is a child of [entry](#), then it applies to the headword in all its senses. When the label is a child of [sense](#), then it applies to the headword in that sense only (**not** including any subsenses linked to it using the [Linking Module](#)). When the label is a child of [inflectedForm](#), then it applies only to that inflected form of the headword (in all senses). When the label is a child of [pronunciation](#), then it applies only to that pronunciation of the headword (in all senses).

*Example 25. XML*

```
<label tag="..."/>
```

*Example 26. JSON*

Implemented as a string

```
"..."
```

*Example 27. RDF*

```
<entry> dmlex:label [
  a dmlex:Label ;
  dmlex:tag "...";
  dmlex:listingOrder 1 ] .
```

*Example 28. Relational database*

labels	
<b>PK</b>	id
FK	entryID
FK	senseID
FK	inflectedFormID
FK	pronunciationID
FK	exampleID
	tag
	listingOrder

labels	
<b>PK</b>	<b>id</b>
FK	entryID
FK	senseID
FK	inflectedFormID
FK	pronunciationID
FK	exampleID
	tag
	listingOrder

### 3.810 pronunciation

Represents the pronunciation of its parent. Examples: [Section A.1.3, "Pronunciation given as transcription"](#), [Section A.1.4, "Pronunciation given as a sound file"](#), [Section A.1.5, "Pronunciation given both ways"](#).

*Property of*

- [entry](#)
- [inflectedForm](#)

*Properties*

- At least one of:
  - `soundFile` OPTIONAL (zero or one) and UNIQUE. An *Internationalized Resource Identifier (IRI)* pointing to a file which contains a sound recording of the pronunciation
  - [transcription](#) OPTIONAL (zero or more) and UNIQUE. **Normalised string.**
- `listingOrder` REQUIRED (exactly one). Number. The position of this pronunciation object among other pronunciation objects of the same entry. This can be implicit from the serialization.
- [label](#) OPTIONAL (zero or more).

#### Note

The value of the `soundFile` property is an IRI. This means, among other things, that it can be an internet URL or a relative file path, and can contain non-ASCII characters.

*Example 29. XML*

```
<pronunciation soundFile="...">
  <transcription.../>
  <label.../>
</pronunciation>
```

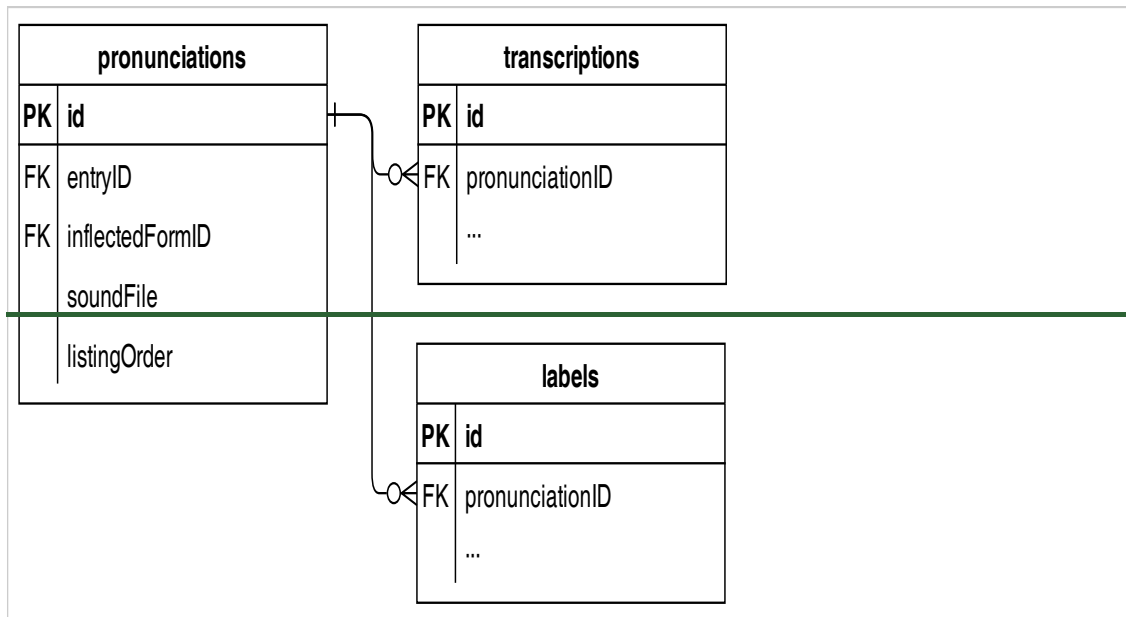
Example 30. JSON

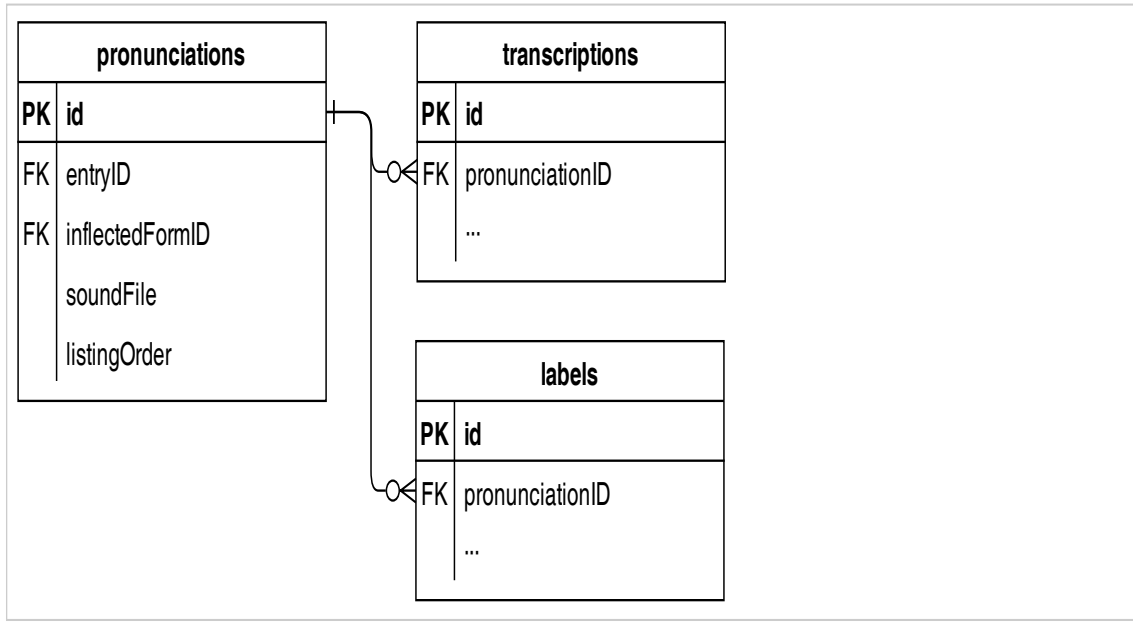
```
{  
  "soundFile": "...",  
  "transcriptions": [...],  
  "labels": [...]  
}
```

Example 31. RDF

```
<entry> dmlex:pronunciation [  
  a dmlex:Pronunciation ;  
  dmlex:soundFile <...> ;  
  dmlex:transcription ... ;  
  dmlex:listingOrder 1 ;  
  dmlex:label ... ] .
```

Example 32. Relational database





### 3.9.11 transcription

Represents the transcription of a pronunciation in some notation such as IPA.

Property of

- [pronunciation](#)

Properties

- text REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ **Normalised** string. The actual transcription.
- scheme OPTIONAL (zero or one). IETF language tag. Identifies the transcription scheme used here. Example: en-fonipa for English IPA. This can be implicit if the lexicographic resource uses only one transcription scheme throughout. The [transcriptionSchemeTag](#) object type can be used to define which transcription schemes are allowed in the lexicographic resource.
- listingOrder REQUIRED (exactly one). Number. The position of this transcription object among transcriptions of the same pronunciation. This can be implicit from the serialization.

Example 33. XML

```
<transcription scheme="...">...</="...">
  <text>...</text>
</transcription>
```

Example 34. JSON

```
{
  "text": "...",
  "scheme": "..."
```

```
}
```

*Example 35. RDF*

```
<pronunciation> dmlex:transcription [  
  a dmlex:Transcription ;  
  dmlex:scheme "...";  
  dmlex:listingOrder 1 ] .
```

*Example 36. Relational database*

transcriptions	
PK	id
FK	pronunciationID
	text
	scheme
	listingOrder

transcriptions	
PK	id
FK	pronunciationID
	text
	scheme
	listingOrder

**3.10.12** [example](#)

Represents a sentence or other text fragment which illustrates the headword being used.

*Property of*

- [sense](#)

*Properties*

- text REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ [Normalised](#) string. The example itself.

- `sourceIdentity` OPTIONAL (zero or one). [Normalised string](#). An abbreviation, a code or some other string of text which identifies the source. The `sourceIdentityTag` object type can be used to explain the meaning of the source identifiers, to constrain which source identifiers are allowed to occur in the lexicographic resource, and to map them onto external inventories and ontologies.
- `sourceElaboration` OPTIONAL (zero or one). [Non-empty Normalised string](#). A free-form statement about the source of the example. If `sourceIdentity` is present, then `sourceElaboration` can be used for information about where in the source the example can be found: page number, chapter and so on. If `sourceIdentity` is absent then `sourceElaboration` can be used to fully name the source.
- `label` OPTIONAL (zero or more).
- `soundFile` OPTIONAL (zero or one). [An Internationalized Resource Identifier \(IRI pointer\) pointing to a file, such as a filename or a URI, containing which contains](#) a sound recording of the example.
- `listingOrder` REQUIRED (exactly one). Number. The position of this example object among examples of the same sense. This can be implicit from the serialization.

**Example 37. XML**

```
<example sourceIdentity="..." sourceElaboration="..." soundFile="...">
  <text>...</text>
  <label.../>
</example>
```

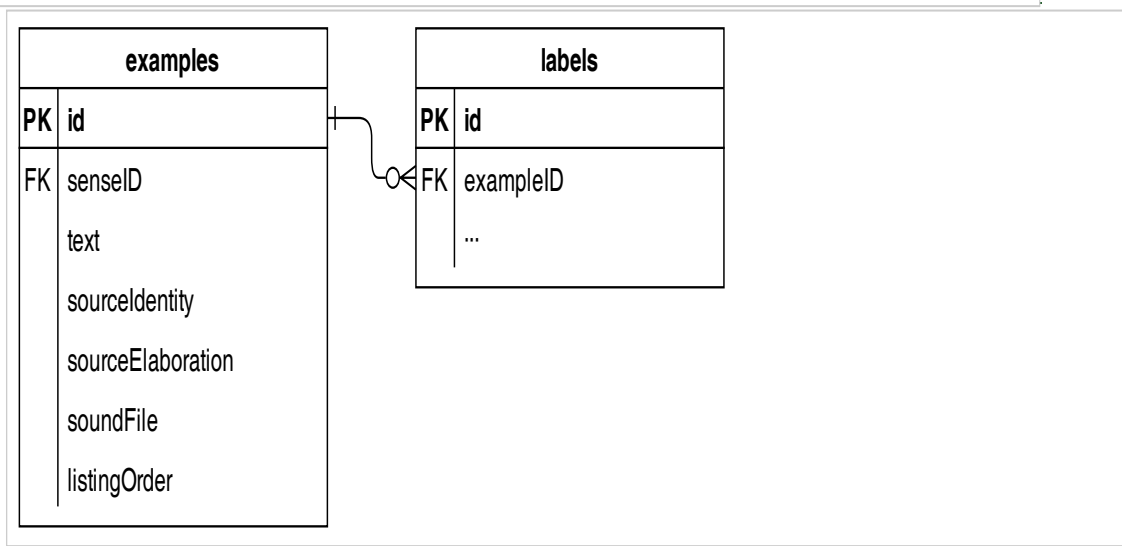
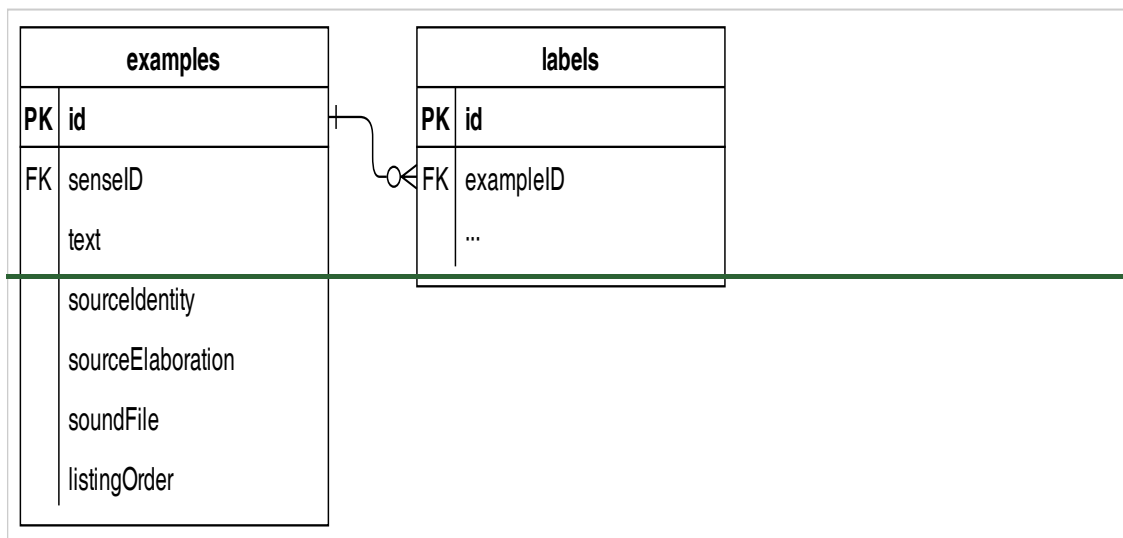
**Example 38. JSON**

```
{
  "text": "...",
  "sourceIdentity": "...",
  "sourceElaboration": "...",
  "labels": [...],
  "soundFile": "..."
}
```

**Example 39. RDF**

```
<sense> dmlex:example [
  a dmlex:Example ;
  dmlex:text "...";
  dmlex:sourceIdentity "...";
  dmlex:sourceElaboration "...";
  dmlex:label ... ;
  dmlex:soundFile <...> ;
  dmlex:listingOrder 1 ] .
```

**Example 40. Relational database**



## 4 DMLex OPTIONAL Modules (Normative)

### 4.1 DMLex Crosslingual Module

DMLex's Multilingual Module extends the Core and turns a monolingual lexicographic resource into a bilingual or multilingual one. A bilingual or multilingual lexicographic resource is a lexicographic resource with multiple (two or more) languages: the headwords and the examples are in one language (called "the language of the lexicographic resource" in DMLex) and their translations are in one or more other languages (called the *translation languages* in DMLex).

#### 4.1.1 Extensions to `lexicographicResource`

Extends the `lexicographicResource` object type from the [Core](#).

*Additional properties*

- `translationLanguage` REQUIRED (one or more)



#### Example 41. XML

```
<lexicographicResource ...>
  ...
  <translationLanguage.../>
</lexicographicResource>
```

#### Example 42. JSON

```
{
  ...,
  "translationLanguages": [...]
}
```

#### Example 43. RDF

```
<#lexicographicResource> dmlex:translationLanguage ...
```

### 4.1.2 translationLanguage

Represents one of the languages in which translations are given in this lexicographic resource. Examples: [Section A.1.8, “Defining a bilingual lexicographic resource”](#), [Section A.1.9, “Defining a multilingual lexicographic resource”](#).

#### Property of

- [lexicographicResource](#)

#### Properties

- langCode REQUIRED (exactly one) and UNIQUE. The IETF language code of the language.
- listingOrder REQUIRED (exactly one). Number. Sets the order in which translations (of headwords and examples) should be shown. It outranks the listing order given in [headwordTranslation](#), [headwordExplanation](#) and [exampleTranslation](#) objects.

#### Example 44. XML

```
<translationLanguage langCode="" />
```

#### Example 45. JSON

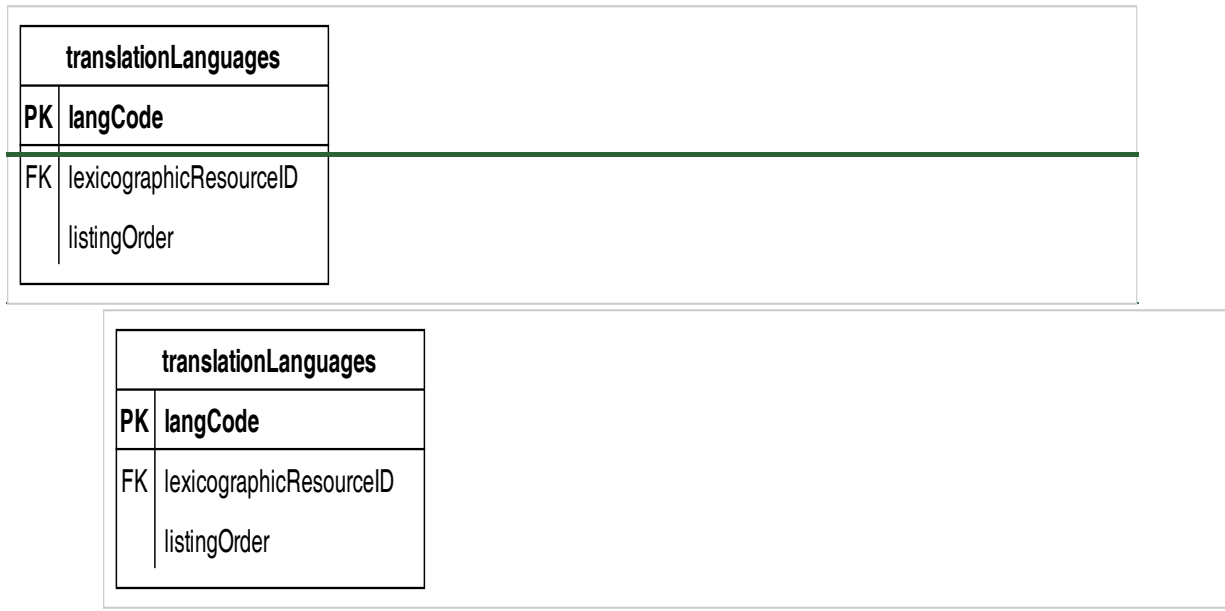
Implemented as a string

```
"..."
```

*Example 46. RDF*

```
<#lexicographicResource> dmlex:translationLanguage [
  dmlex:langCode ... ;
  dmlex:listingOrder 0 ] .
```

*Example 47. Relational database*



### 4.1.3 Extensions to *sense*

Extends the [sense](#) object type from the [Core](#).

*Additional properties*

- [headwordExplanation](#) OPTIONAL (zero or more)
- [headwordTranslation](#) OPTIONAL (zero or more)

*Example 48. XML*

```
<sense ...>
  ...
  <headwordExplanation.../>
  <headwordTranslation.../>
  ...
</sense>
```

*Example 49. JSON*

```
{
  ...
  "headwordExplanations": [...],
  "headwordTranslations": [...],
  ...
}
```

*Example 50. RDF*

```
<#sense>
  dmlex:headwordExplanation ... ;
  dmlex:headwordTranslation ... .
```

#### 4.1.4 headwordTranslation

Represents one of possibly multiple translations of a headword. Examples: [Section A.1.10, “How to use headwordTranslation in a bilingual lexicographic resource”](#), [Section A.1.11, “How to use headwordTranslation in a multilingual lexicographic resource”](#).

*Property of*

- [sense](#)

*Properties*

- `text` REQUIRED (exactly one) and UNIQUE (in combination with `langCode` if present). ~~Non-empty~~ **Normalised** string.
- `langCode` OPTIONAL (zero or one) if only one translation language exists in the lexicographic resource, REQUIRED (exactly one) and UNIQUE (in combination with `text`) otherwise. IETF language tag. Indicates the language of this translation. The [translationLanguage](#) datatype can be used to explain the meaning of the language codes that appear here and/or to constrain which language codes are allowed.
- `listingOrder` REQUIRED (exactly one). Number. The position of this translation among other translations of the same sense in the same language. ~~Can~~ **This can** be implicit from the serialization.
- [partOfSpeech](#) OPTIONAL (zero or more).
- [label](#) OPTIONAL (zero or more).
- [pronunciation](#) OPTIONAL (zero or more).
- [inflectedForm](#) OPTIONAL (zero or more).

*Example 51. XML*

```
<headwordTranslation langCode="...">
  <text>...</text>
  <partOfSpeech.../>
  <label.../>
  <pronunciation.../>
  <inflectedForm.../>
</headwordTranslation>
```

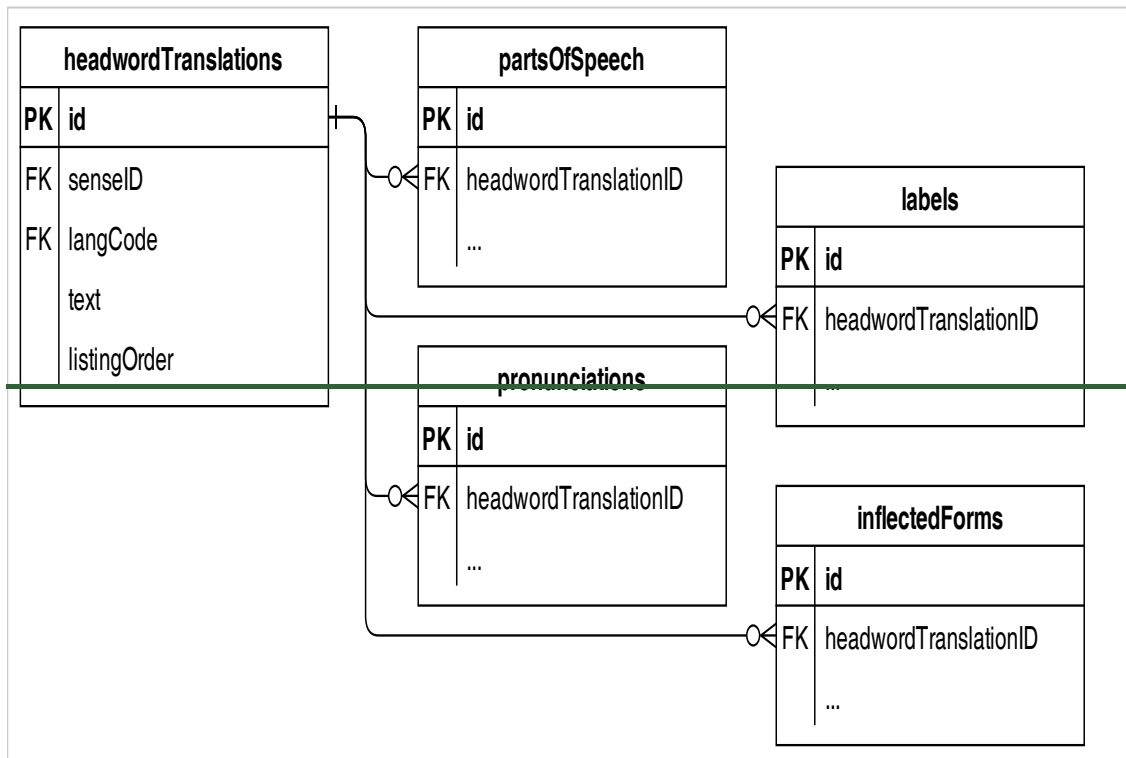
Example 52. JSON

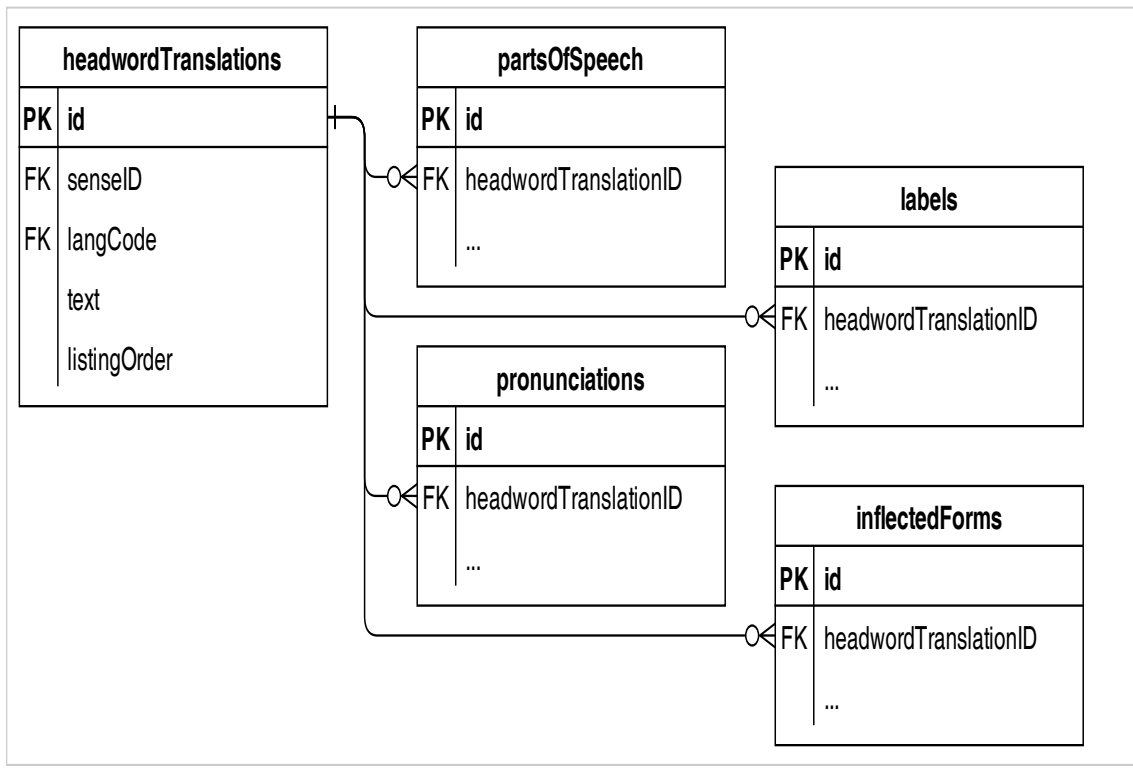
```
{
  "langCode": "...",
  "text": "...",
  "partsOfSpeech": [...],
  "labels": [...],
  "pronunciations": [...],
  "inflectedForms": [...]
}
```

Example 53. RDF

```
<#sense> dmlex:headwordTranslation [
  dmlex:langCode "...";
  dmlex:text "...";
  dmlex:partOfSpeech ...;
  dmlex:label ...;
  dmlex:pronunciation ...;
  dmlex:inflectedForm ... ] .
```

Example 54. Relational database





#### 4.1.5 headwordExplanation

Represents a statement in the translation language which explains (but does not translate) the meaning of the headword. Example: [Section A.1.12, "How to use headwordExplanation"](#).

*Property of*

- [sense](#)

*Properties*

- text REQUIRED (exactly one) and UNIQUE (in combination with langCode if present). ~~Non-empty~~ Normalised string.
- langCode OPTIONAL (zero or one) if only one translation language exists in the lexicographic resource, REQUIRED (exactly one) and UNIQUE otherwise (in combination with text). IETF language tag. Indicates the language in which this explanation is written. The [translationLanguage](#) datatype can be used to explain the meaning of the language codes that appear here and/or to constrain which language codes are allowed.

*Comments*

- It is assumed that there will always be a maximum of one headwordExplanation per translation language in each sense. For this reason, headwordExplanation does not have a listingOrder.

*Example 55. XML*

```
<headwordExplanation langCode="...">...</="...">
  <text>...</text>
</headwordExplanation>
```

Example 56. JSON

```
{
  "langCode": "...",
  "text": "...",
}
```

Example 57. RDF

```
<#sense> dmlex:headwordExplanation [
  dmlex:langCode "...";
  dmlex:text "..."] .
```

Example 58. Relational database

headwordExplanations	
PK	id
FK	senseID
FK	langCode
	text

headwordExplanations	
PK	id
FK	senseID
FK	langCode
	text

#### 4.1.6 Extensions to [example](#)

Extends the [example](#) object type from the [Core](#).

*Additional properties*

- [exampleTranslation](#) OPTIONAL (zero or more)

*Example 59. XML*

```
<example ...>
  ...
  <exampleTranslation.../>
</example>
```

*Example 60. JSON*

```
{
  ...,
  "exampleTranslations": [...]
}
```

*Example 61. RDF*

```
<#example> dmlex:exampleTranslation ... .
```

### 4.1.7 [exampleTranslation](#)

Represents the translation of an example.

*Property of*

- [example](#)

*Properties*

- `text` REQUIRED (exactly one) and UNIQUE (in combination with `langCode` if present). ~~Non-empty~~ **Normalised** string.
- `langCode` OPTIONAL (zero or one) if only one translation language exists in the lexicographic resource, REQUIRED (exactly one) and UNIQUE (in combination with `text`) otherwise. IETF language tag. Indicates the language of this translation. The [translationLanguage](#) datatype can be used to explain the meaning of the language codes that appear here and/or to constrain which language codes are allowed.
- `label` OPTIONAL (zero or more).
- `soundFile` OPTIONAL (zero or one). **An Internationalized Resource Identifier (IRI) pointing to a file, such as a filename or a URL, containing** ~~A pointer to a file, such as a filename or a URL, containing~~ **pointing to a file which contains** a sound recording of the translation.
- `listingOrder` REQUIRED (exactly one). Number. The position of this translation among other translations of the same example in the same language. ~~Can~~ **This can** be implicit from the serialization.

*Example 62. XML*

```

<exampleTranslation langCode="..." soundFile="...">
  <text>...</text>
  <label.../>
</exampleTranslation>

```

*Example 63. JSON*

```

{
  "langCode": "...",
  "text": "...",
  "labels": [...],
  "soundFile": "..."
}

```

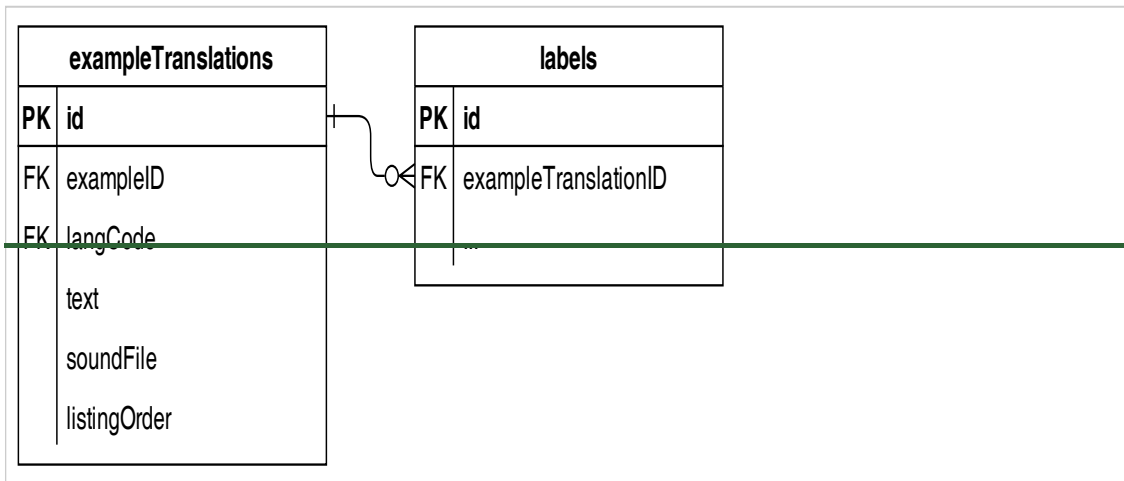
*Example 64. RDF*

```

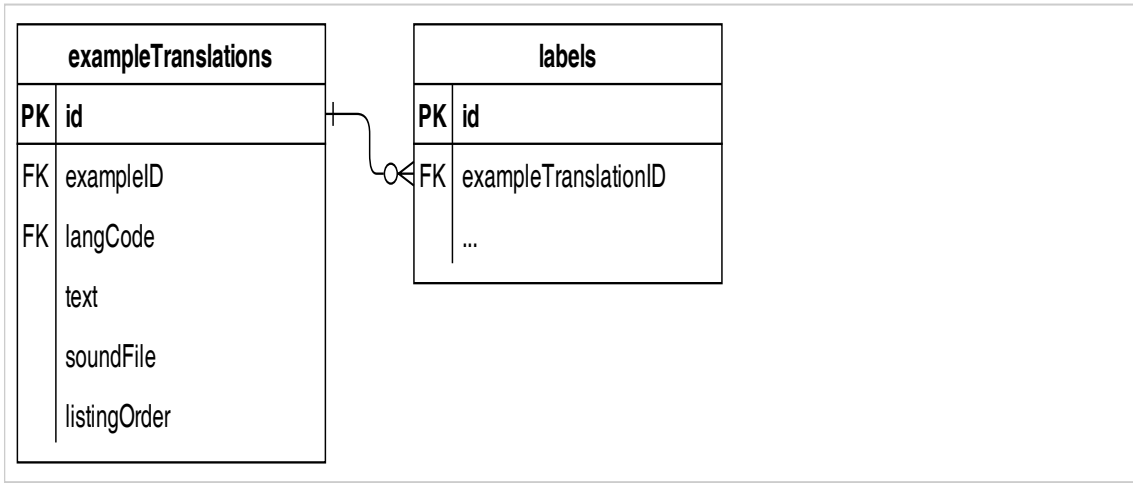
<#example> dmlex:exampleTranslation [
  dmlex:langCode "...";
  dmlex:text "...";
  dmlex:label ...;
  dmlex:soundFile "..." ] .

```

*Example 65. Relational database*







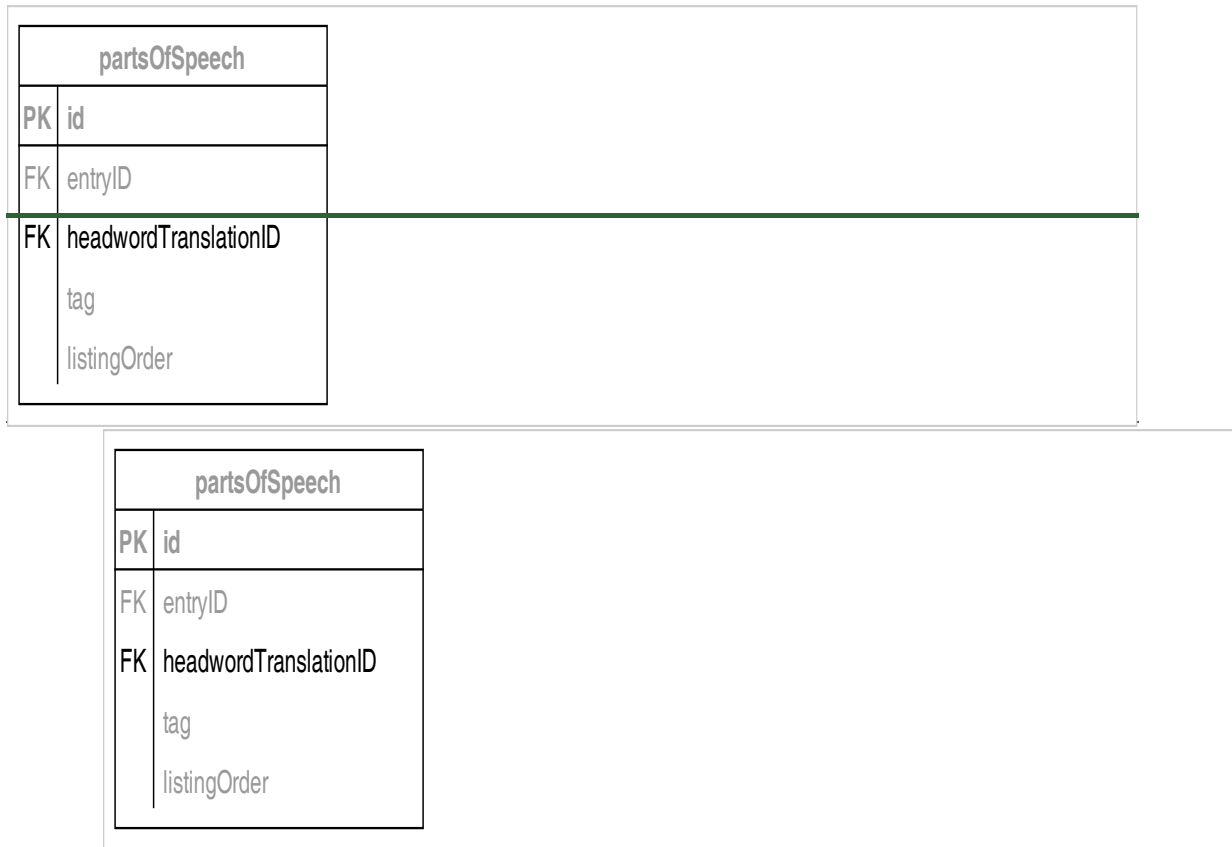
#### 4.1.8 Extensions to [partOfSpeech](#)

Extends the [partOfSpeech](#) object type from the [Core](#).

Can additionally be a property of

- [headwordTranslation](#)

Example 66. Relational database



#### 4.1.9 Extensions to `label`

Extends the [label](#) object type from the [Core](#).

Can additionally be a property of

- [headwordTranslation](#)
- [exampleTranslation](#)

*Example 67. Relational database*

labels	
PK	id
FK	entryID
FK	senseID
FK	inflectedFormID
FK	pronunciationID
FK	exampleID
FK	headwordTranslationID
FK	exampleTranslationID
	tag
	listingOrder

labels	
<b>PK</b>	id
FK	entryID
FK	senseID
FK	inflectedFormID
FK	pronunciationID
FK	exampleID
FK	headwordTranslationID
FK	exampleTranslationID
	tag
	listingOrder

#### 4.1.10 Extensions to [pronunciation](#)

Extends the [pronunciation](#) object type from the [Core](#).

*Can additionally be a property of*

- [headwordTranslation](#)

*Example 68. Relational database*

pronunciations	
<b>PK</b>	id
FK	entryID
FK	inflectedFormID
FK	headwordTranslationID
	soundFile
	listingOrder

pronunciations	
PK	id
FK	entryID
FK	inflectedFormID
FK	headwordTranslationID
	soundFile
	listingOrder

#### 4.1.11 Extensions to `inflectedForm`

Extends the [inflectedForm](#) object type from the [Core](#).

Can additionally be a property of

- [headwordTranslation](#)

*Example 69. Relational database*

inflectedForms	
PK	id
FK	entryID
FK	headwordTranslationID
	tag
	text
	listingOrder

inflectedForms	
PK	id
FK	entryID
FK	headwordTranslationID
	tag
	text
	listingOrder

## 4.2 DMLex Controlled Values Module

DMLex's Controlled Values Module extends the Core and makes it possible to represent inventories from which the values of various properties come from, such as [parts of speech](#), [labels](#), [inflected form tags](#) and others.

### *Comment*

- Treating controlled values as constraints in an implementation of DMLex, for example as business rules in a dictionary-writing system, is OPTIONAL.

### 4.2.1 Extensions to `lexicographicResource`

Extends the [lexicographicResource](#) object type from the [Core](#).

#### *Additional properties*

- [definitionTypeTag](#) OPTIONAL (zero or more)
- [inflectedFormTag](#) OPTIONAL (zero or more)
- [labelTag](#) OPTIONAL (zero or more)
- [labelTypeTag](#) OPTIONAL (zero or more)
- [partOfSpeechTag](#) OPTIONAL (zero or more)
- [sourceIdentityTag](#) OPTIONAL (zero or more)
- [transcriptionSchemeTag](#) OPTIONAL (zero or more)

#### *Example 70. XML*

```
<lexicographicResource ...>
  ...
  <definitionTypeTag.../>
  <inflectedFormTag.../>
  <labelTag.../>
  <labelTypeTag.../>
  <partOfSpeechTag.../>
  <sourceIdentityTag.../>
</lexicographicResource>
```

```
<transcriptionSchemeTag.../>
</lexicographicResource>
```

#### Example 71. JSON

```
{
  ...,
  "definitionTypeTags": [...],
  "inflectedFormTags": [...],
  "labelTags": [...],
  "labelTypeTags": [...],
  "partOfSpeechTags": [...],
  "sourceIdentityTags": [...],
  "transcriptionSchemeTags": [...]
}
```

#### Example 72. RDF

```
<#lexicographicResource>
  dmlex:definitionTypeTag ... ;
  dmlex:inflectedFormTag ... ;
  dmlex:labelTag ... ;
  dmlex:labelTypeTag ... ;
  dmlex:partOfSpeechTag ... ;
  dmlex:sourceIdentityTag ... ;
  dmlex:transcriptionSchemeTag ... .
```

### 4.2.2 definitionTypeTag

Represents one (of many) possible values for definitionType of [definition](#).

#### Property of

- [lexicographicResource](#)

#### Properties

- tag REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ **Normalised** string. An abbreviation, a code or some other string of text.
- description OPTIONAL (zero or one). ~~Non-empty~~ **Normalised** string. A human-readable description of what the tag means.
- [sameAs](#) OPTIONAL (zero or more).

#### Example 73. XML

```
<definitionTypeTag tag="..."
  <description>...</description>
  <sameAs.../>
```

```
</definitionTypeTag>
```

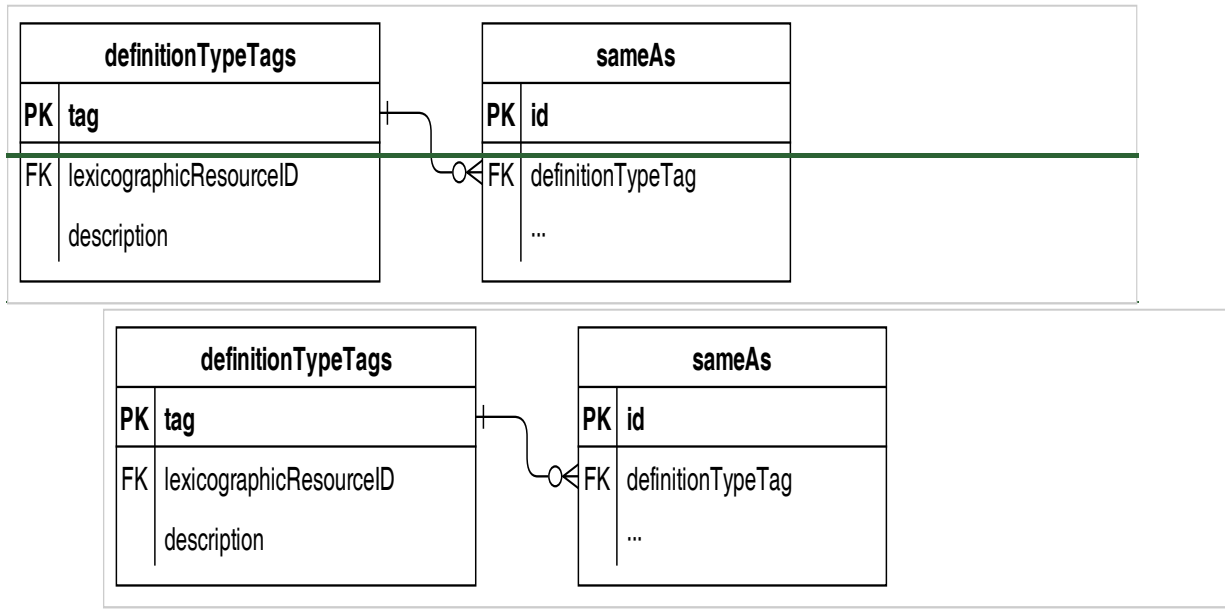
*Example 74. JSON*

```
{  
  "tag": "...",  
  "description": "...",  
  "sameAs": [...]  
}
```

*Example 75. RDF*

```
<lexicographicResource> dmlex:definitionTypeTag [  
  a dmlex:DefinitionTypeTag ;  
  dmlex:tag "...";  
  dmlex:description "...";  
  owl:dmlex:sameAs ... ] .
```

*Example 76. Relational database*



### 4.2.3 *inflectedFormTag*

Represents one (of many) possible values for **tag** of *inflectedForm*. Example: [Section A.1.6, "How to use partOfSpeechTag and inflectedFormTag"](#).

Property of

- [lexicographicResource](#)

## Properties

- tag REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~Normalised string. An abbreviation, a code or some other string of text.
- description OPTIONAL (zero or one). ~~Non-empty~~Normalised string. A human-readable description of what the tag means.
- for OPTIONAL (zero or one). ~~Non-empty~~Normalised string. A statement about constraints and recommendation on using this tag. This can be a human-readable description such as "for use with German nouns only" or a machine-readable instruction in any format understandable to the DMLex implementation, such as `partOfSpeech=noun AND language=de`.
- [sameAs](#) OPTIONAL (zero or more).

### Example 77. XML

```
<inflectedFormTag tag="..." for="...">
  <description>...</description>
  <sameAs.../>
</inflectedFormTag>
```

### Example 78. JSON

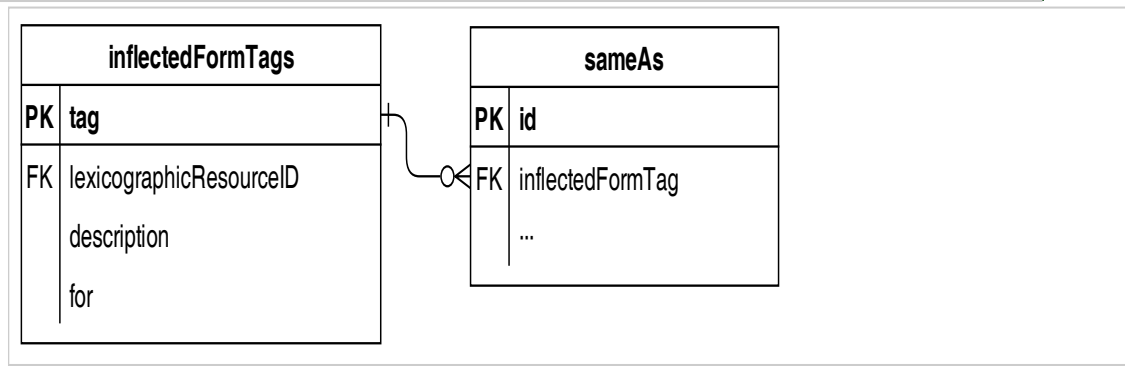
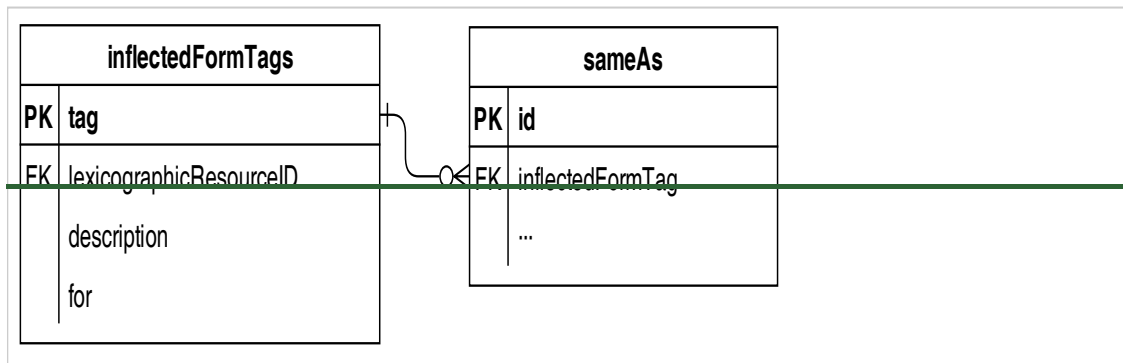
```
{
  "tag": "...",
  "description": "...",
  "for": "...",
  "sameAs": [...]
}
```

### Example 79. RDF

```
<lexicographicResource> dmlex:inflectedFormTag [
  a dmlex:InflectedFormTag ;
  dmlex:tag "...";
  dmlex:description "...";
  dmlex:for "...";
  owl:dmlex:sameAs ... ] .
```

### Example 80. Relational database





#### 4.2.4 labelTag

Represents one (of many) possible values for tag of [label](#).

Property of

- [lexicographicResource](#)

Properties

- tag REQUIRED (exactly one) and UNIQUE. **Non-emptyNormalised** string. An abbreviation, a code or some other string of text.
- description OPTIONAL (zero or one). **Non-emptyNormalised** string. A human-readable description of what the tag means.
- typeTag OPTIONAL (zero or one). **Non-emptyNormalised** string. An abbreviation, a code or some other string of text which identifies the type of the label, for example `temporal` for temporal labels (archaic, neologism etc) or `dialect` for labels of dialects. The [labelTypeTag](#) object type can be used to explain the meaning of the type tags, to constrain which type tags are allowed to occur in the lexicographic resource, and to map them onto external inventories and ontologies.
- for OPTIONAL (zero or one). **Non-emptyNormalised** string. A statement about constraints and recommendation on using this tag. This can be a human-readable description such as "for use with German nouns only" or a machine-readable instruction in any format understandable to the DMLex implementation, such as `partOfSpeech=noun AND language=de`.
- [sameAs](#) OPTIONAL (zero or more).

Example 81. XML

```
<labelTag tag="..." typeTag="..." for="...">
```

```

<description>...</description>
<sameAs.../>
</labelTag>

```

Example 82. JSON

```

{
  "tag": "...",
  "description": "...",
  "typeTag": "...",
  "for": "...",
  "sameAs": [...]
}

```

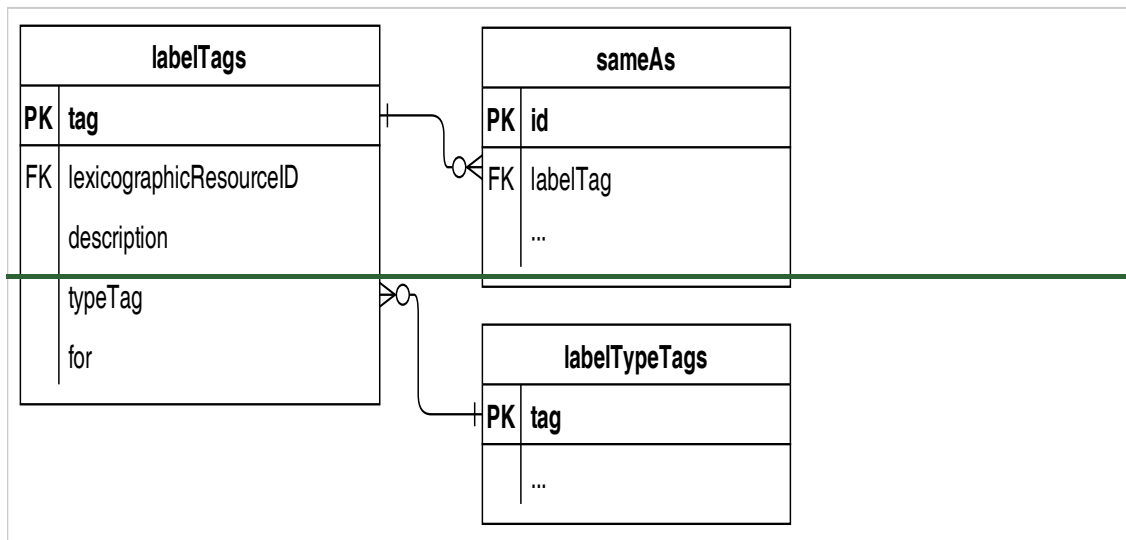
Example 83. RDF

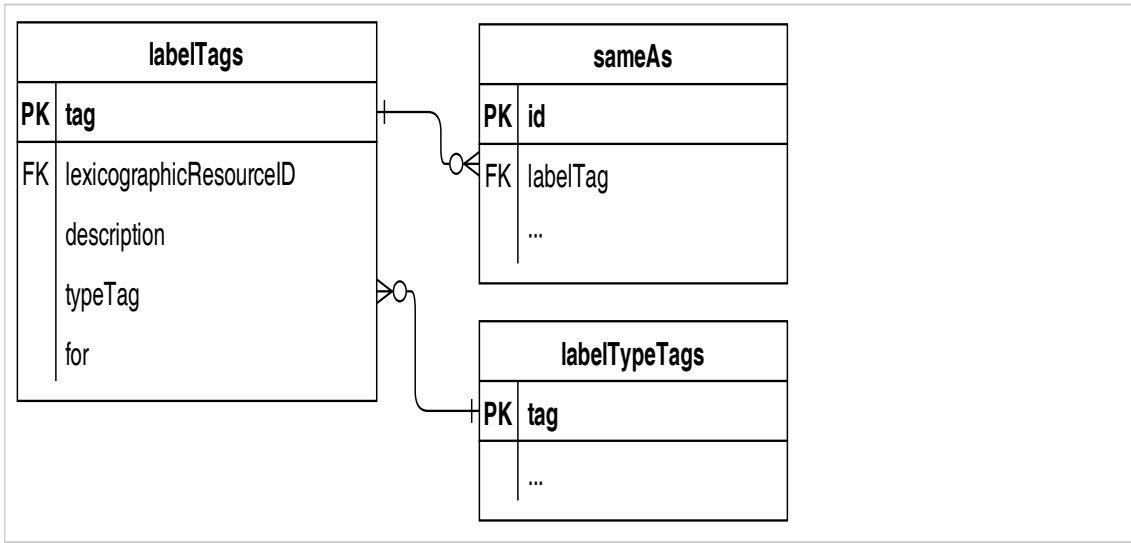
```

<lexicographicResource> dmlex:labelTag [
  a dmlex:LabelTag ;
  dmlex:tag "...";
  dmlex:description "...";
  dmlex:typeTag "...";
  dmlex:for "...";
  owl:dmlex:sameAs ... ] .

```

Example 84. Relational database





#### 4.2.5 labelTypeTag

Represents one (of many) possible values for `typeTag` of [labelTag](#).

Property of

- [lexicographicResource](#)

Properties

- `tag` REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ **Normalised** string. An abbreviation, a code or some other string of text.
- `description` OPTIONAL (zero or one). ~~Non-empty~~ **Normalised** string. A human-readable description of what the type tag means.
- [sameAs](#) OPTIONAL (zero or more).

Example 85. XML

```
<labelTypeTag tag="...">
  <description>...</description>
  <sameAs.../>
</labelTypeTag>
```

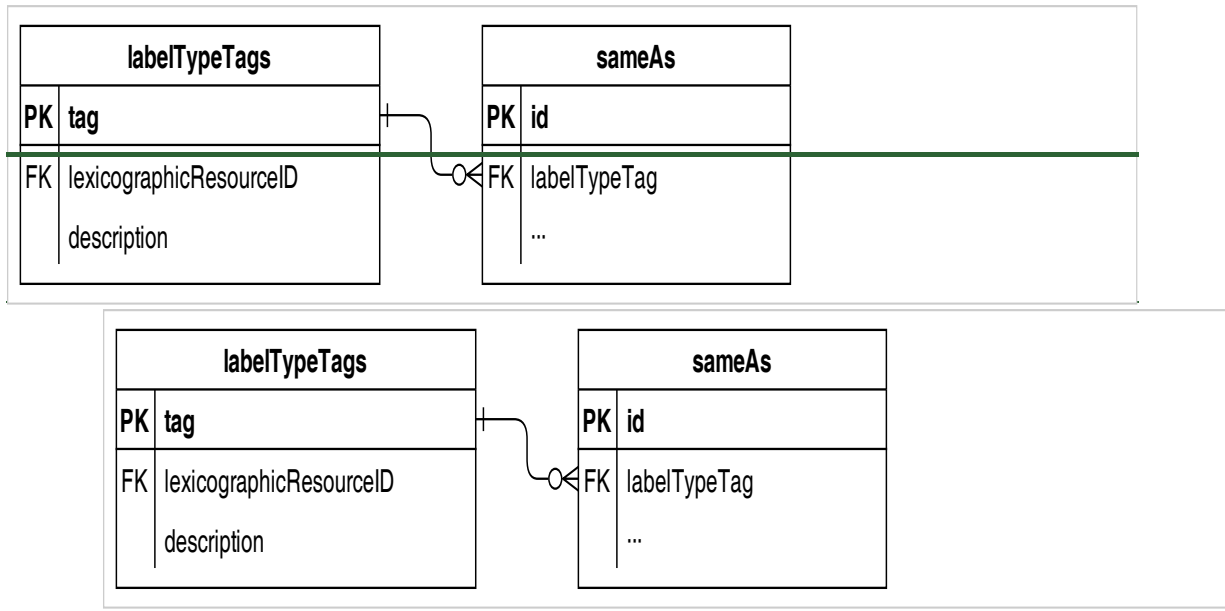
Example 86. JSON

```
{
  "tag": "...",
  "description": "...",
  "sameAs": [...]
}
```

### Example 87. RDF

```
<lexicographicResource> dmlex:labelTypeTag [  
  a dmlex:LabelTypeTag ;  
  dmlex:tag "..."  
  dmlex:description "..."  
  owl:dmlex:sameAs ... ] .
```

### Example 88. Relational database



### 4.2.6 partOfSpeechTag

Represents one (of many) possible values for tag of [partOfSpeech](#). Example: [Section A.1.6. "How to use partOfSpeechTag and inflectedFormTag"](#).

#### Property of

- [lexicographicResource](#)

#### Properties

- tag REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~Normalised string. An abbreviation, a code or some other string of text.
- description OPTIONAL (zero or one). ~~Non-empty~~Normalised string. A human-readable description of what the tag means.
- for OPTIONAL (zero or one). ~~Non-empty~~Normalised string. A statement about constraints and recommendation on using this tag. This can be a human-readable description such as "for use in German and Czech only" or a machine-readable instruction in any format understandable to the DMLex implementation, such as language=de OR language=cs.
- [sameAs](#) OPTIONAL (zero or more).

**Example 89. XML**

```
<partOfSpeechTag tag="..." for="...">
  <description>...</description>
  <sameAs.../>
</partOfSpeechTag>
```

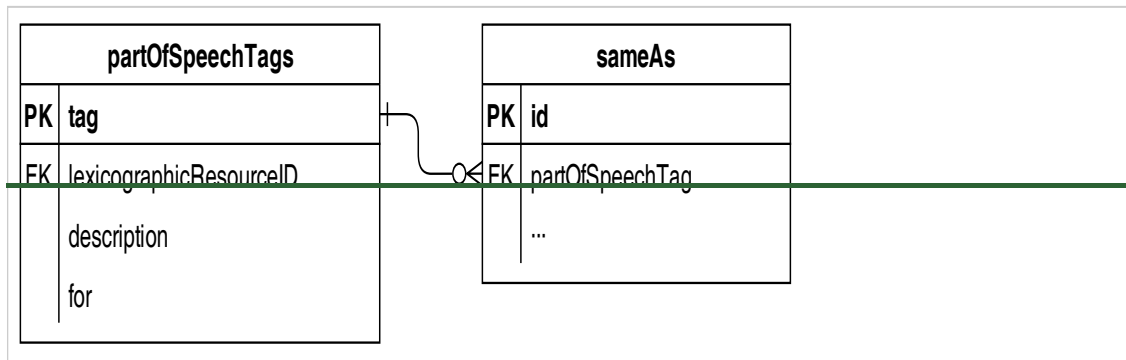
**Example 90. JSON**

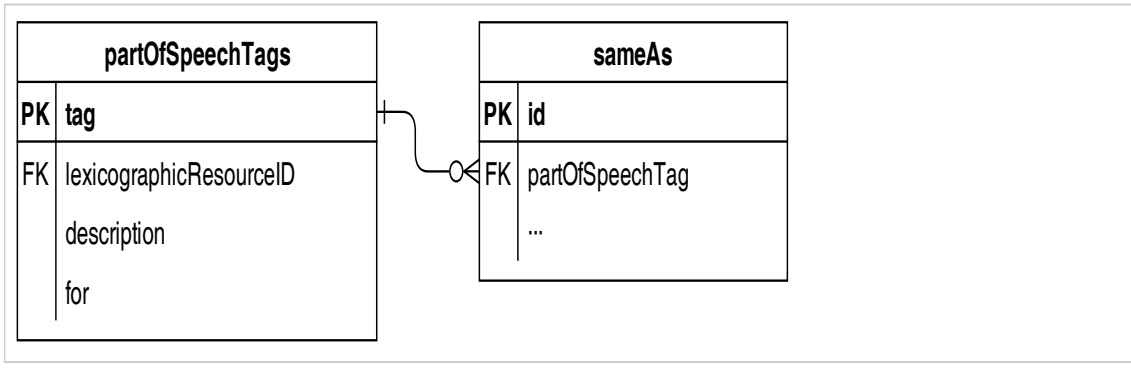
```
{
  "tag": "...",
  "description": "...",
  "for": "...",
  "sameAs": [...]
}
```

**Example 91. RDF**

```
<lexicographicResource> dmlex:partOfSpeechTag [
  a dmlex:PartOfSpeechTag ;
  dmlex:tag "...";
  dmlex:description "...";
  dmlex:for "...";
  owl:dmlex:sameAs ... ] .
```

**Example 92. Relational database**





#### 4.2.7 sourceIdentityTag

Represents one (of many) possible values for sourceIdentity of [example](#).

*Property of*

- [lexicographicResource](#)

*Properties*

- tag REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ **Normalised** string. An abbreviation, a code or some other string of text.
- description OPTIONAL (zero or one). ~~Non-empty~~ **Normalised** string. A human-readable description of what the tag means.
- [sameAs](#) OPTIONAL (zero or more).

*Example 93. XML*

```

<sourceIdentityTag tag="..."
  <description>...</description>
  <sameAs.../>
</sourceIdentityTag>
  
```

*Example 94. JSON*

```

{
  "tag": "...",
  "description": "...",
  "sameAs": [...]
}
  
```

*Example 95. RDF*

```

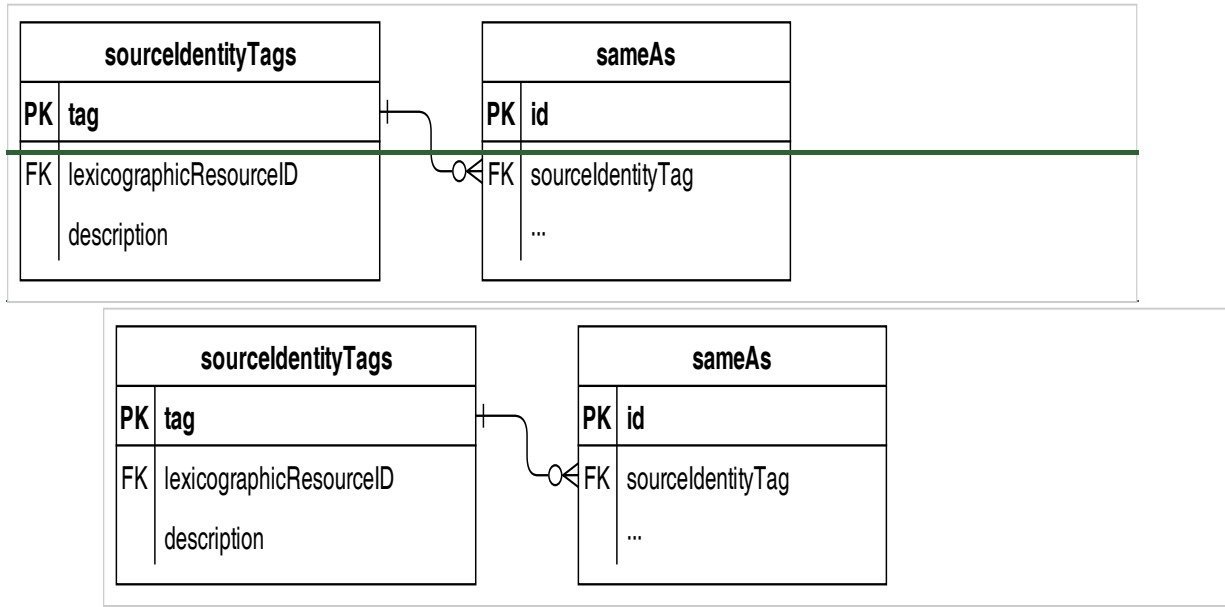
<lexicographicResource> dmlex:sourceIdentityTag [
  a dmlex:SourceIdentityTag ;
  
```

```

dmlex:tag "...";
dmlex:description "...";
owl:dmlex:sameAs ... ] .

```

Example 96. Relational database



#### 4.2.8 transcriptionSchemeTag

Represents one (of many) possible values for scheme of [transcription](#).

Property of

- [lexicographicResource](#)

Properties

- tag REQUIRED (exactly one) and UNIQUE. An IETF language tag.
- description OPTIONAL (zero or one). ~~Non-empty~~Normalised string. A human-readable description of what the tag means.
- for OPTIONAL (zero or one). ~~Non-empty~~Normalised string. A statement about constraints and recommendation on using this tag. This can be a human-readable description such as "for use in German and Czech only" or a machine-readable instruction in any format understandable to the DMLex implementation, such as language=de OR language=cs.

Comment

- The transcriptionSchemeTag does not have a sameAs property because the tag itself - which is an IETF language tag - defines fully what the tag means.

Example 97. XML

```
<transcriptionSchemeTag tag="..." for="...">
  <description>...</description>
</transcriptionSchemeTag>
```

*Example 98. JSON*

```
{
  "tag": "...",
  "description": "...",
  "for": "..."
}
```

*Example 99. RDF*

```
<lexicographicResource> dmlex:transcriptionSchemeTag [
  a dmlex:PartOfSpeechTag ;
  dmlex:tag "...";
  dmlex:description "...";
  dmlex:for "..."] .
```

*Example 100. Relational database*

transcriptionSchemeTags	
PK	tag
FK	lexicographicResourceID
	description
	for

transcriptionSchemeTags	
PK	tag
FK	lexicographicResourceID
	description
	for

#### 4.2.9 sameAs



Represents the fact that the parent object is equivalent to an item available from an external authority.  
Example: [Section A.1.7, "Mapping controlled values to external inventories"](#).

*Property of*

- [definitionTypeTag](#)
- [inflectedFormTag](#)
- [labelTag](#)
- [labelTypeTag](#)
- [partOfSpeechTag](#)
- [sourceIdentityTag](#)

*Properties*

- `uri` REQUIRED (exactly one) and UNIQUE. The URI of an item in an external inventory.

*Example 101. XML*

```
<sameAs uri="..."/>
```

*Example 102. JSON*

Implemented as a string

```
"..."
```

*Example 103. Relational database*

sameAs	
PK	id
FK	definitionTypeTag
FK	inflectedFormTag
FK	labelTag
FK	partOfSpeechTag
FK	sourceIdentityTag
	uri

sameAs	
<b>PK</b>	id
FK	definitionTypeTag
FK	inflectedFormTag
FK	labelTag
FK	partOfSpeechTag
FK	sourceIdentityTag
	uri

### 4.3 DMLex Linking Module

DMLex's Linking Module can be used to construct *relations* between objects which "break out" of the tree-like parent-and-child hierarchy constructed from datatypes from the Core and from other modules. The Linking Module can be used to create relations between senses which are synonyms or antonyms, between entries whose headwords are homonyms or spelling variants, between senses which represent superordinate and subordinate concepts (eg. hypernyms and hyponyms, holonyms and meronyms), between entries and subentries, between senses and subsenses, and many others.

Each relation is represented in DMLex by an instance of the `relation` datatype. A relation brings two or more *members* together. The fact that an object (such as a sense or an entry) is a member of a relation is represented in DMLex by an instance of the `member` datatype.

The Linking Module can be used to set up relations between objects inside the same lexicographic resource, or between objects residing in different lexicographic resources.

For linking, some type of reference IDs of linked objects are needed (cf. the `ref` property in Section 4.3.3, "member"). DMLex does not prescribe the exact form of these IDs, however, a recommended method for creating unique IRIs for DMLex objects is available in Section 3.2.1, "DMLex fragment identification strings", which may be useful especially when linking objects from different lexicographic resources on the Web.

Examples: [Section A.1.13, "Modelling parts and wholes"](#), [Section A.1.14, "Modelling antonyms"](#), [Section A.1.15, "Modelling synonyms"](#), [Section A.1.16, "Modelling variants"](#), [Section A.1.17, "Modelling subsenses"](#), [Section A.1.18, "Modelling subentries \(at subsense level\)"](#), [Section A.1.19, "Modelling subentries \(at sense level\)"](#).

#### 4.3.1 Extensions to `lexicographicResource`

Extends the [lexicographicResource](#) object type from the [Core](#).

*Additional properties*

- [relation](#) OPTIONAL (zero or more)
- [relationType](#) OPTIONAL (zero or more)

*Example 104. XML*

```
<lexicographicResource ...>
  ...
  <relation.../>
  <relationType.../>
</lexicographicResource>
```

*Example 105. JSON*

```
{
  ...,
  "relations": [...],
  "relationTypes": [...]
}
```

*Example 106. RDF*

```
<#lexicographicResource>
  dmlex:relation ... ;
  dmlex:relationType ... .
```

### 4.3.2 relation

Represents the fact that a relation exists between two or more objects.

*Property of*

- [lexicographicResource](#)

*Properties*

- **type** REQUIRED (exactly one). ~~Non-empty~~ **Normalised** string. Specifies what type of relation it is, for example a relation between synonyms or a relation between a sense and a subsense. Optionally, [relationType](#) objects can be used to explain those types and to constrain which types of relations are allowed to exist in the lexicographic resource.
- **description** OPTIONAL (zero or one). ~~Non-empty~~ **Normalised** string. A human-readable explanation of this relation.
- **member** REQUIRED (two or more).

*Example 107. XML*

```
<relation type="...">
  <description>...</description>
  <member.../>
</relation>
```

*Example 108. JSON*

```

{
  "type": "...",
  "description": "...",
  "members": [...]
}

```

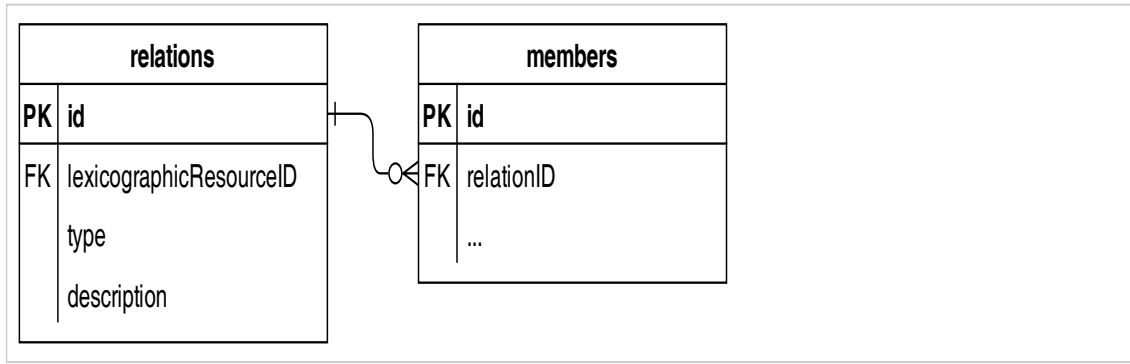
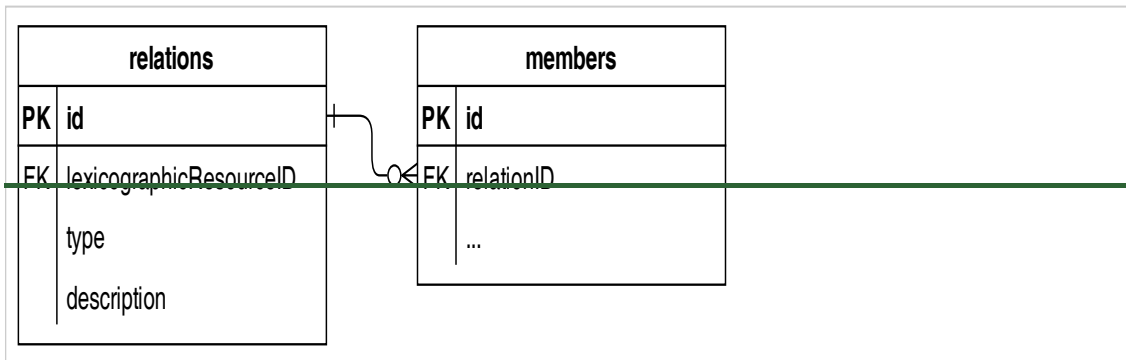
Example 109. RDF

```

<#lexicographicResource> dmlex:relation [
  a dmlex:Relation ;
  dmlex:type "...";
  dmlex:description "...";
  dmlex:member ... ] .

```

Example 110. Relational database



### 4.3.3 member

Represents the fact that an object, such as an entry or a sense, is a member of a relation.

Property of

- [relation](#)

Properties

- `ref` REQUIRED (exactly one) and UNIQUE (in combination with other unique properties if present). Reference to an object, such as an entry or a sense. [The IRI addressing mechanism described in Section 3.2.1, "DMLex fragment identification strings" can be used \(but is not REQUIRED\)](#).
- `role` OPTIONAL (zero or one) and UNIQUE (in combination with other unique properties if present). ~~Non-empty~~ [Normalised](#) string. An indication of the role the member has in this relation: whether it is the hypernym or the hyponym (in a hyperonymy/hyponymy relation), or whether it is one of the synonyms (in a synonymy relation), and so on. You can use [memberType](#) objects to explain those roles and to constrain which relations are allowed to contain which roles, what their object types are allowed to be (eg. entries or senses) and how many members with this role each relation is allowed to have.
- `listingOrder` REQUIRED (exactly one). Number. The position of this member among other members of the same relation. When showing members of the relation to human users (for example: when listing the synonyms in a synonymy relation), the members should be listed in this order. This can be implicit from the serialization.
- `obverseListingOrder` OPTIONAL (at most one). Number. The position of this relation among other relations this member is involved in. When an object - such as an entry or a sense - is a member of several relations (for example: when a sense is a member of a synonymy relation and also of an antonymy relation) then, when showing the object (the entry or the sense) to human users, the relations should be listed in this order (for example: the synonyms first, the antonyms second).

*Example 111. XML*

```
<member ref="..." role="..." obverseListingOrder="..." />
```

*Example 112. JSON*

```
{
  "ref": "...",
  "role": "...",
  "obverseListingOrder": "..."
}
```

*Example 113. RDF*

```
<#relation> dmlex:member [
  a dmlex:Member ;
  dmlex:ref "...";
  dmlex:role "...";
  dmlex:listingOrder 0 ;
  dmlex:obverseListingOrder 0 ] .
```

*Example 114. Relational database*

members	
PK	id
FK	relationID
FK	memberEntryID
FK	memberSenseID
	role
	listingOrder
	obverseListingOrder

members	
PK	id
FK	relationID
FK	memberEntryID
FK	memberSenseID
	role
	listingOrder
	obverseListingOrder

#### 4.3.4 `relationType`

Represents one of possible values for the type of [relation](#).

*Property of*

- [lexicographicResource](#)

*Properties*

- type REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ Normalised string.
- description OPTIONAL (zero or one). ~~Non-empty~~ Normalised string. A human-readable explanation of this relation type.
- scopeRestriction OPTIONAL (zero or one). ~~Non-empty string~~. Specifies restrictions on member of relations of this type. The possible values are:
  - sameEntry: members must be within of the same entry
  - sameResource: members must be within the same lexicographicResource
  - any: no restriction

- [memberType](#) OPTIONAL (zero or more).
- [sameAs](#) OPTIONAL (zero or more).

*Example 115. XML*

```
<relationType type="..." scopeRestriction="...">
  <description>...</description>
  <memberType.../>
  <sameAs.../>
</relationType>
```

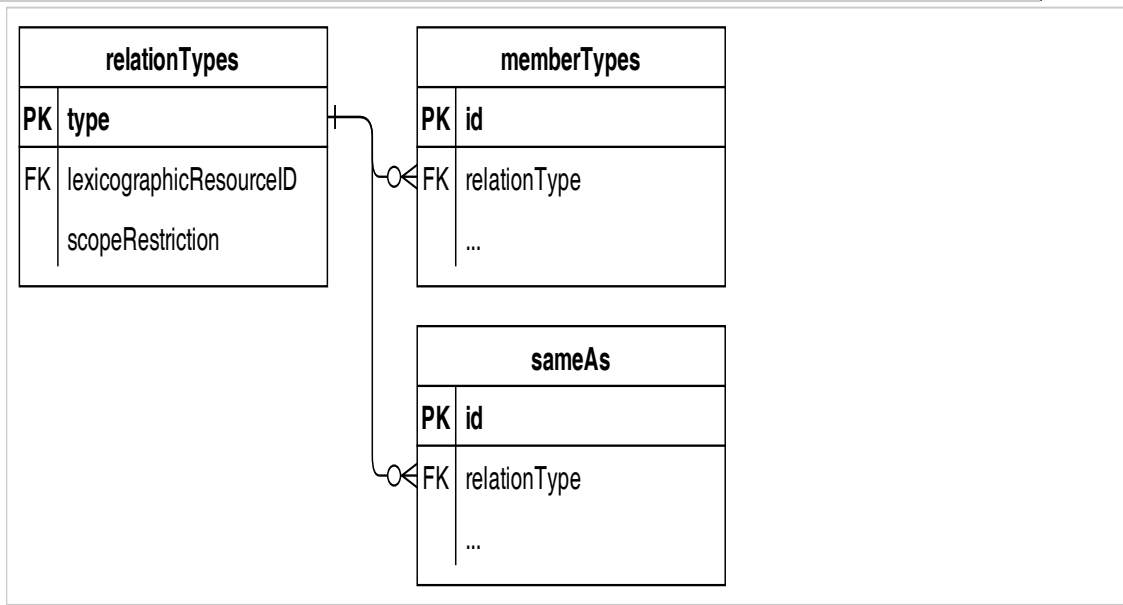
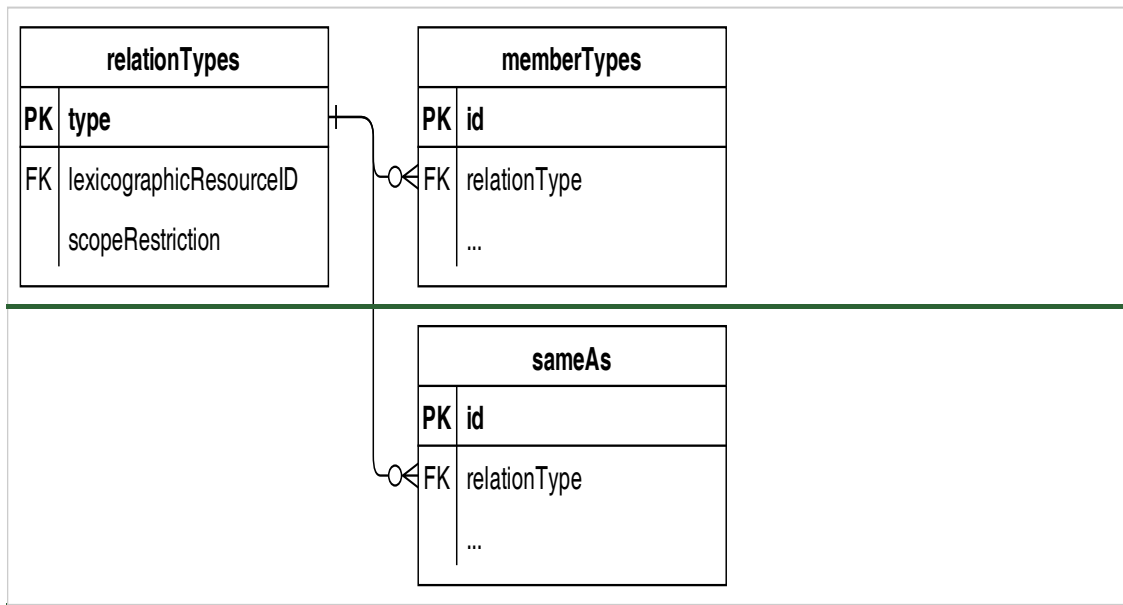
*Example 116. JSON*

```
{
  "type": "...",
  "scopeRestriction": "...",
  "memberTypes": [...],
  "sameAs": ["..."]
}
```

*Example 117. RDF*

```
<#lexicographicResource> dmlex:relationType [
  a dmlex:RelationType ;
  dmlex:type "...";
  dmlex:scopeRestriction "...";
  dmlex:memberType ... ;
  owldmlex:sameAs ... ] .
```

*Example 118. Relational database*



### 4.3.5 memberType

Represents one of possible values for the `role` of [member](#), as well as various restrictions on members having this role.

*Property of*

- [relationType](#)

*Properties*

- `role` OPTIONAL (zero or one) and UNIQUE-String (in combination with other unique properties if present). Normalised string.



- description OPTIONAL (zero or one). ~~Non-empty~~[Normalised](#) string. A human-readable explanation of this member role.
- type REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ [\(in combination with other unique properties if present\)](#). [Normalised](#) string. A restriction on the types of objects that can have this role. The possible values are:
  - sense: the object that has this role must be a [sense](#).
  - entry: the object that has this role must be an [entry](#).
  - collocate: the object that has this role must be an [collocateMarker](#) (from the [Linking module](#)).
- min OPTIONAL (zero or one). [Non-negative integer](#). ~~Number~~. Says that relations of this type must have at least this many members with this role. If omitted then there is no lower limit (effectively, zero).
- max OPTIONAL (zero or one). ~~Number~~. [Non-negative integer](#). Says that relations of this type may have at most this many members with this role. If omitted then there is no upper limit.
- hint OPTIONAL (zero or one). ~~Non-empty string~~. Suggests what machine agents should do when showing this member to a human user (in the context of another member of the same relation). The possible values are:
  - embed: Members that have this role should be shown in their entirety, i.e. the entire entry or the entire sense. This is suitable for relations between entries and subentries, or senses and subsenses.
  - navigate: Members that have this role should not be shown in their entirety. A navigable (e.g. clickable) link should be provided instead. This is suitable for relations between synonyms, antonyms, holonyms/heteronyms and similar.
  - none: Members that have this role should not be shown.
- [sameAs](#) OPTIONAL (zero or more).

**Example 119. XML**

```
<memberType role="..." type="..." min="..." max="..." hint="...">
  <description></description>
  <sameAs.../>
</memberType>
```

**Example 120. JSON**

```
{
  "role": "...",
  "description": "...",
  "type": "...",
  "min": "...",
  "max": "...",
  "hint": "...",
  "sameAs": [...]
}
```

**Example 121. RDF**

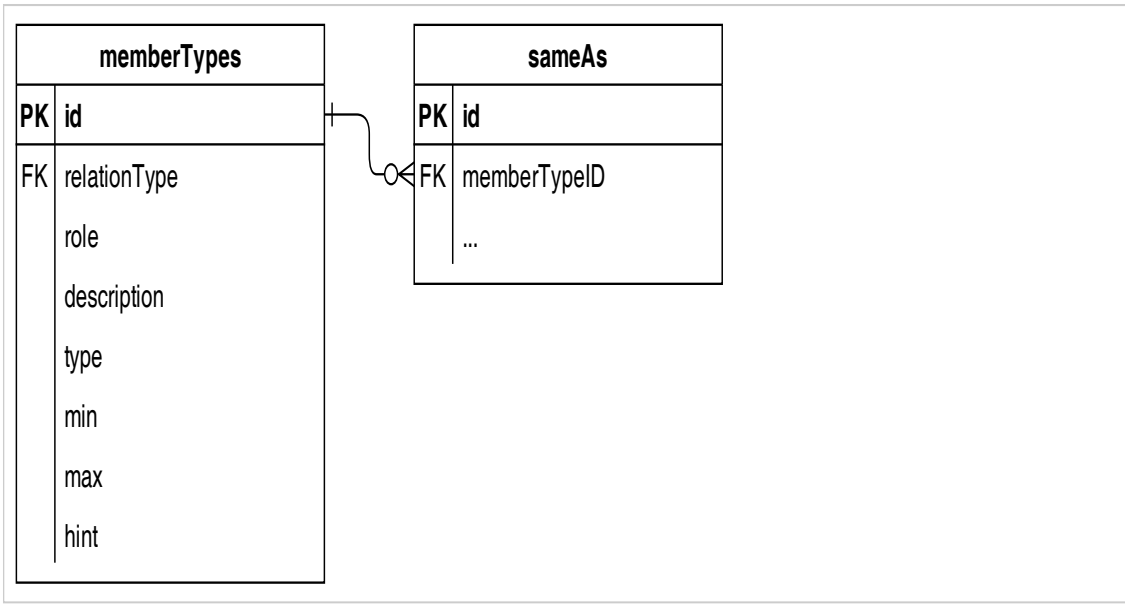
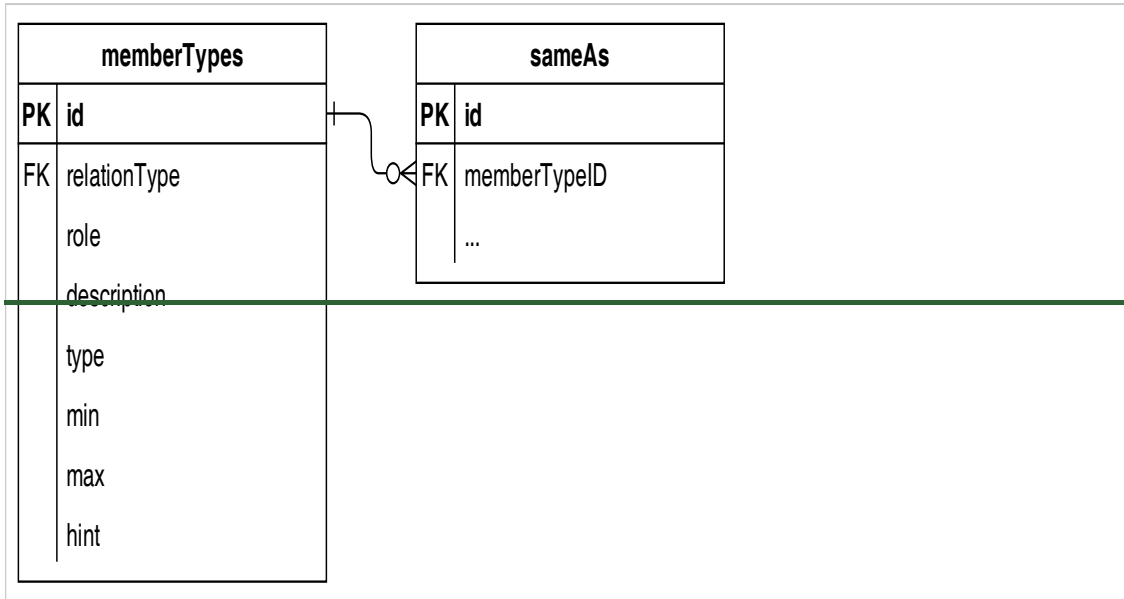
```
<#relationType> dmlex:type [
  a dmlex:MemberType ;
  dmlex:role "...";
  dmlex:description "...";
```

```

dmlex:memberType "...";
dmlex:min 0;
dmlex:max 0;
dmlex:hint "...";
owl:dmlex:sameAs ... ] .

```

Example 122. Relational database



#### 4.3.6 Extensions to `sameAs`

Extends the `sameAs` object type from the [Controlled Values Module](#).

Can additionally be a property of

- [relationType](#)
- [memberType](#)

Example 123. Relational database



## 4.4 DMLex Annotation Module

This module makes it possible to mark up substrings inside the string values of certain objects and to attach properties to them.

It is up to the implementer to decide how to implement annotations, whether as *inline markup* (as XML) or as *stand-off annotations* (for example through start and end indexes).

### 4.4.1 Extensions to `entry`

Extends the [entry](#) object type from the [Core](#).

*Additional properties*

- [placeholderMarker](#) OPTIONAL (zero or more)

Example 124. XML

```
<headword>
  ...<placeholderMarker>...</placeholderMarker>...
</headword>
```

*Example 125. JSON*

```
{
  ...,
  "headword": "...",
  "placeholderMarkers": [...],
  ...
}
```

*Example 126. RDF*

```
<id> a dmlex:Entry ;
  dmlex:headword "...";
  dmlex:placeholderMarker ... .
```

#### 4.4.2 Extensions to `headwordTranslation`

Extends the [headwordTranslation](#) object type from the [Crosslingual module](#).

*Additional properties*

- [placeholderMarker](#) OPTIONAL (zero or more)

*Example 127. XML*

```
<headwordTranslation>
  <text>
    ...<placeholderMarker>...</placeholderMarker>...
  </text>
</headwordTranslation>
```

*Example 128. JSON*

```
{
  "text": "...",
  "placeholderMarkers": [...],
  ...
}
```

*Example 129. RDF*

```
<id> a dmlex:HeadwordTranslation ;
```

```
dmlex:text "... " ;
dmlex:placeholderMarker ... .
```

### 4.4.3 placeholderMarker

Marks up a substring inside a headword or inside a headword translation which is not part of the expression itself but stands for things that can take its place. An application can use the inline markup to format the placeholders differently from the rest of the text, to ignore the placeholder in full-text search, and so on.

Examples: [Section A.1.20, "Using placeholderMarker"](#), [Section A.1.21, "Using placeholderMarker in a bilingual lexicographic resource"](#).

*Property of*

- [entry](#)
- [headwordTranslation](#)

*Properties*

- `startIndex` (~~required~~). **REQUIRED and UNIQUE**. Non-negative integer. The index of the first character of the substring according to the number of Unicode code points. The first character of the string has index 0. This property may be implicit if using in-line markup.
- `endIndex` (~~required~~). **REQUIRED and UNIQUE**. Non-negative integer. The index of the last character after the substring according to the number of Unicode code points. This property may be implicit if using in-line markup.

*Example 130. XML*

```
<placeholderMarker>...</placeholderMarker>
```

*Example 131. JSON*

```
{
  "startIndex": ...,
  "endIndex": ...
}
```

*Example 132. RDF*

```
<#headword> dmlex:placeholderMarker [
  dmlex:startIndex 0 ;
  dmlex:endIndex 1 ] .
```

*Example 133. Relational databases*

placeholderMarkers	
PK	id
FK	entryID
FK	headwordTranslationID startIndex endIndex

placeholderMarkers	
PK	id
FK	entryID
FK	headwordTranslationID startIndex endIndex

#### 4.4.4 Extensions to definition

Extends the [definition](#) object type from the [Core](#).

*Additional properties*

- [headwordMarker](#) OPTIONAL (zero or more)
- [collocateMarker](#) OPTIONAL (zero or more)

*Example 134. XML*

```
<definition...>
  <text>
    ...
    <headwordMarker>...</headwordMarker>
    ...
    <collocateMarker...>...</collocateMarker>
    ...
  </text>
</definition>
```

*Example 135. JSON*

```

{
  "text": "...",
  "headwordMarkers": [...],
  "collocateMarkers": [...],
  ...
}

```

*Example 136. RDF*

```

<#sense> dmlex:definition [
  dmlex:text "...";
  dmlex:headwordMarker ...;
  dmlex:collocateMarker ... ] .

```

#### 4.4.5 Extensions to `example`

Extends the [example](#) object type from the [Core](#).

*Additional properties*

- [headwordMarker](#) OPTIONAL (zero or more)
- [collocateMarker](#) OPTIONAL (zero or more)

*Example 137. XML*

```

<example>
  <text>
    ...
    <headwordMarker>...</headwordMarker>
    ...
    <collocateMarker...>...</collocateMarker>
    ...
  </text>
</example>

```

*Example 138. JSON*

```

{
  "text": "...",
  "headwordMarkers": [...],
  "collocateMarkers": [...],
  ...
}

```

*Example 139. RDF*

```
<#sense> dmlex:example [
  dmlex:text "...";
  dmlex:headwordMarker ...;
  dmlex:collocateMarker ... ] .
```

#### 4.4.6 Extensions to `exampleTranslation`

Extends the [exampleTranslation](#) object type from the [Crosslingual module](#).

*Additional properties*

- [headwordMarker](#) OPTIONAL (zero or more)
- [collocateMarker](#) OPTIONAL (zero or more)

*Example 140. XML*

```
<exampleTranslation>
  <text>
    ...
    <headwordMarker>...</headwordMarker>
    ...
    <collocateMarker...>...</collocateMarker>
    ...
  </text>
</exampleTranslation>
```

*Example 141. JSON*

```
{
  "text": "...",
  "headwordMarkers": [...],
  "collocateMarkers": [...],
  ...
}
```

*Example 142. RDF*

```
<#example> dmlex:exampleTranslation [
  dmlex:text "...";
  dmlex:headwordMarker ...;
  dmlex:collocateMarker ... ] .
```

#### 4.4.7 `headwordMarker`

Marks up a substring inside an example, inside an example translation or inside a definition which corresponds to the headword (or to a translation of the headword). An application can use the inline markup



to highlight the occurrence of the headword for human readers through formatting. Example: [Section A.1.22, "Using headwordMarker"](#).

#### Property of

- [definition](#)
- [example](#)
- [exampleTranslation](#)

#### Properties

- `startIndex` (~~required~~). **REQUIRED and UNIQUE**. Non-negative integer. The index of the first character of the substring according to the number of Unicode code points. The first character of the string has index 0. This property may be implicit if using in-line markup.
- `endIndex` (~~required~~). **REQUIRED and UNIQUE**. Non-negative integer. The index of the last character after the substring according to the number of Unicode code points. This property may be implicit if using in-line markup

#### Example 143. XML

```
<headwordMarker>...</headwordMarker>
```

#### Example 144. JSON

```
{  
  "startIndex": ...,  
  "endIndex": ...  
}
```

#### Example 145. RDF

```
<#definition> dmlex:headwordMarker [  
  dmlex:startIndex 0 ;  
  dmlex:endIndex 0 ] .
```

#### Example 146. Relational databases

headwordMarkers	
PK	id
FK	definitionID
FK	exampleID
FK	exampleTranslationID
	startIndex
	endIndex

headwordMarkers	
PK	id
FK	definitionID
FK	exampleID
FK	exampleTranslationID
	startIndex
	endIndex

#### 4.4.8 collocateMarker

Marks up a substring other than the headword inside an example, inside an example translation or inside a definition. An application can use the inline markup to highlight collocates or constituents. Example: [Section A.1.23, "Using collocateMarker"](#).

*Property of*

- [definition](#)
- [example](#)
- [exampleTranslation](#)

*Properties*

- `startIndex` ~~(required)~~ **REQUIRED and UNIQUE**. Non-negative integer. The index of the first character of the substring according to the number of Unicode code points. The first character of the string has index 0. This property may be implicit if using in-line markup.
- `endIndex` ~~(required)~~ **REQUIRED and UNIQUE**. Non-negative integer. The index of the last character after the substring according to the number of Unicode code points. This property may be implicit if using in-line markup.
- `lemma` **OPTIONAL (zero or one)** ~~and UNIQUE. Non-empty~~. **Normalised** string. The lemmatized form of the collocate. An application can use it to provide a clickable link for the user to search for

the lemma in the rest of the lexicographic resource or on the web. (If you want to link the collocate explicitly to a specific entry or to a specific sense in your lexicographic resource, or even in an external lexicographic resource, you can use the Linking Module for that.)

- [label](#) OPTIONAL (zero or more). [CanThis can](#) be used to communicate facts about the role or type of the item in the sentence, for example its syntactic role (subject, direct object etc.), its semantic role (agent, affected etc) or its semantic type (human, institution etc.) The [labelTag](#) object type can be used to explain and/or constrain the collocate labels that are allowed to appear in the lexicographic resource.

*Example 147. XML*

```
<collocateMarker id="..." lemma="...">...<label  
tag="..."/></collocateMarker>
```

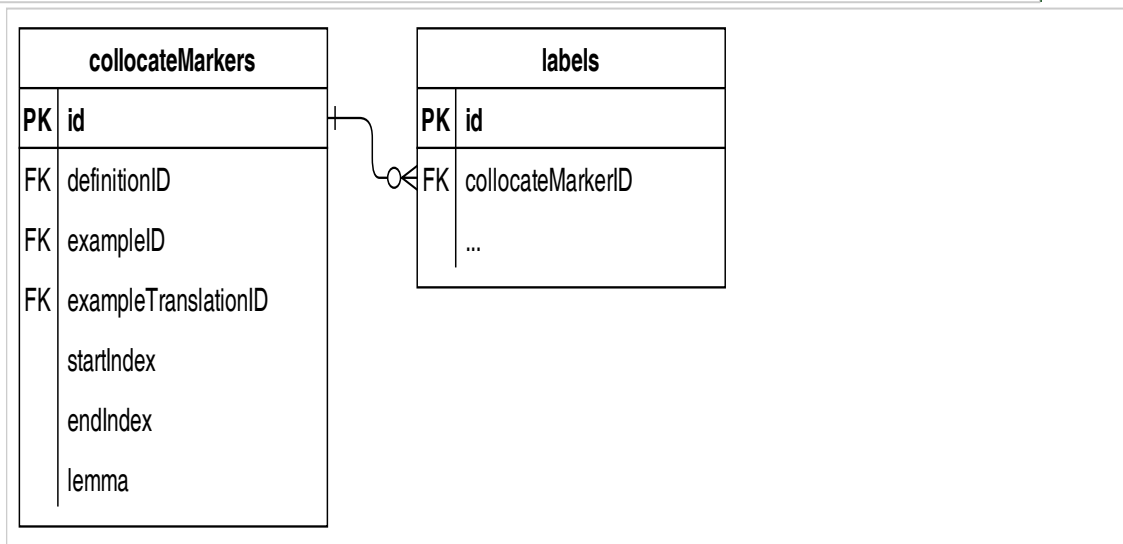
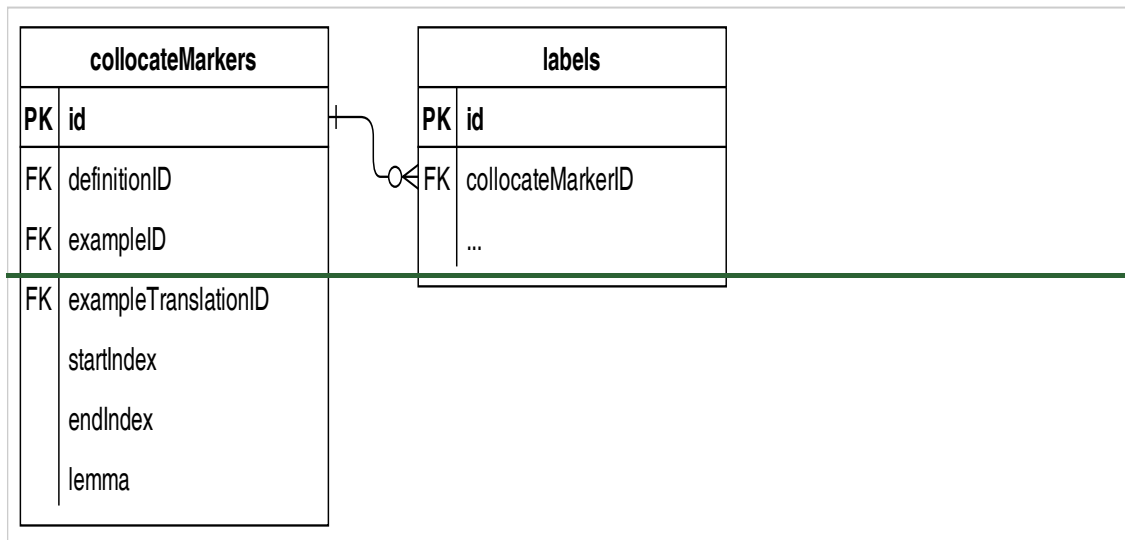
*Example 148. JSON*

```
{  
  "id": "...",  
  "startIndex": ...,  
  "endIndex": ...,  
  "lemma": "...",  
  "labels": ["..."]  
}
```

*Example 149. RDF*

```
<#definition> dmlex:itemMarker [  
  dmlex:id "...";  
  dmlex:startIndex 0 ;  
  dmlex:endIndex 0 ;  
  dmlex:lemma "...";  
  dmlex:label ... ] .
```

*Example 150. SQL*



#### 4.4.9 Extensions to `label`

Extends the `label` object type from the [Core](#).

*Can additionally be a property of*

- [collocateMarker](#)

*Example 151. Relational database*

labels	
PK	id
	...
FK	collocateMarkerID

labels	
PK	id
	...
FK	collocateMarkerID

## 4.5 DMLex Etymology Module

Extends DMLex Core to support the modelling of etymological information in lexicographic resources.

Example: [Section A.1.24, "Modelling etymology"](#).

### 4.5.1 Extensions to `entry`

Extends the `entry` object type from the [Core](#).

*Additional properties*

- `etymology` OPTIONAL (zero or more) Represents a chain of historical derivations of the entry's headword.

#### Note

If an entry contains more than one etymology object, then the different etymology objects represent different hypotheses about the origin of the headword.

*Example 152. XML*

```
<entry>
  ...
  <etymology>...</etymology>
</entry>
```

*Example 153. JSON*

```
{
  ...,
  "etymologies": [...]
}
```

*Example 154. RDF*

```
<#entry> dmlex:etymology ... ;
  dmlex:etymonType ... .
```

## 4.5.2 etymology

Represents a chain of historical derivations of a word.

*Property of*

- [entry](#)

*Properties*

- At least one of:
  - `description` OPTIONAL (zero or one) and **UNIQUE**. Normalised string. A plain-text form of the etymology, which may contain notes about the etymology. This may be used instead of or alongside a structured list of origin and etymon objects.
  - `etymon` OPTIONAL (zero or more) and **UNIQUE**.
- `listingOrder` REQUIRED (exactly one). Number. The position of this etymology among other etymologies of the same entry. ~~Can~~This can be implicit from the serialization.

*Example 155. XML*

```
<etymology>
  <description>...</description>
  <etymon>...</etymon>
  <etymon>...</etymon>
  <etymon>...</etymon>
  ...
</etymology>
```

*Example 156. JSON*

```
{
  "description": ... ,
  "etymons": [...]
}
```

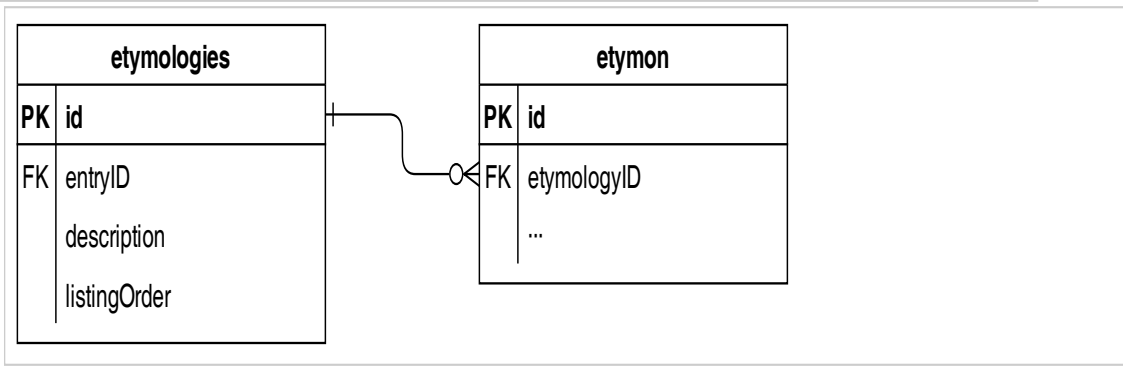
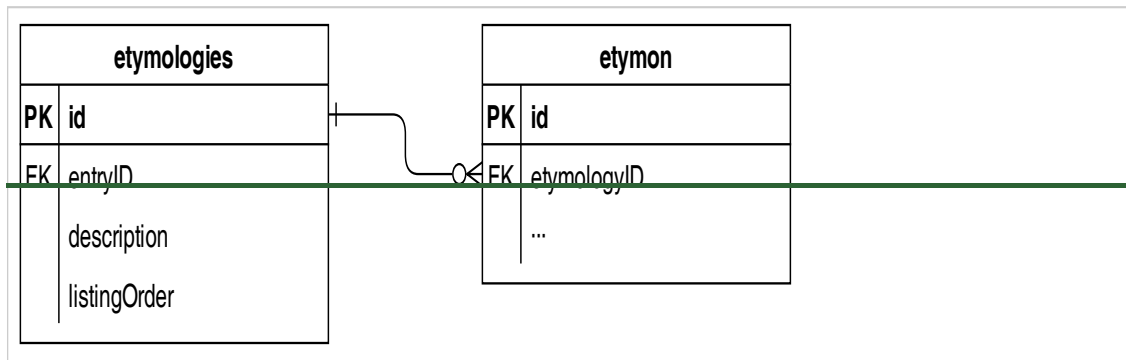
*Example 157. RDF*

```

<#entry> dmlex:etymology [
  a dmlex:Etymology ;
  dmlex:description "...";
  dmlex:etymon ... ;
  dmlex:listingOrder: ...] .

```

Example 158. Relational database



### 4.5.3 etymon

Represents one stage (of possibly several) in the etymological history of the headword.

Property of

- [etymology](#)

Properties

- when **OPTIONAL** (zero or one) and **UNIQUE** (in combination with other unique properties if present). **Normalised string**. Indicates the time period during which this etymological origin is valid. The value is an open-text indication of the time period, in the language of the dictionary.
- type **OPTIONAL** (zero or one). **Normalised string**. The type of the etymological process that occurred at this stage of the headword's etymological history such as derivation, cognate, borrowing. The values can be explained and constrained using the [etymonType](#) object type.
- note **OPTIONAL** (zero or one). **Normalised string**. Any additional information about this stage of the headword's etymological history.

- `etymonUnit` REQUIRED (one or more) and UNIQUE ~~(in combination with other unique properties if present).~~
- `translation` OPTIONAL (zero or one). Normalised string. A translation or gloss of the etymon in the language of the lexicographic resource.
- `listingOrder` REQUIRED (exactly one). Number. The position of this origin among other origins listed in the etymology. ~~Can~~This can be implicit from the serialization.

*Example 159. XML*

```
<etymon when="..." type="...">
  <note>...</note>
  <etymonUnit>...</etymonUnit>
  <etymonUnit>...</etymonUnit>
  <translation>...</translation>
  ...
</etymon>
```

*Example 160. JSON*

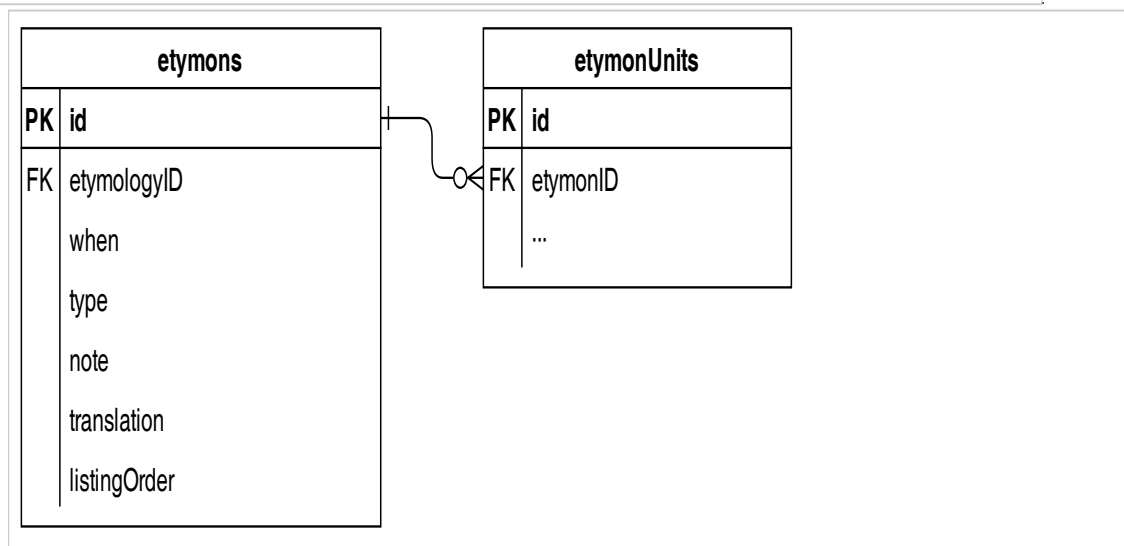
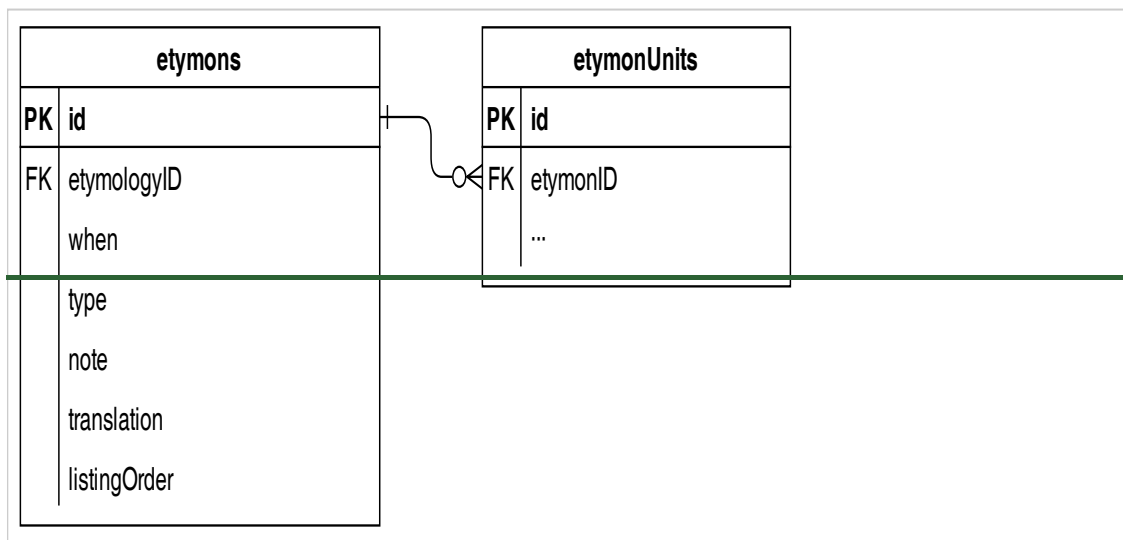
```
{
  "when": "...",
  "type": "...",
  "note": "...",
  "etymonUnits": [...],
  "translation": "..."
}
```

*Example 161. RDF*

```
<#etymon>
  dmlex:when "...";
  dmlex:type "...";
  dmlex:note "...";
  dmlex:etymonUnit ...;
  dmlex:translation ...;
  dmlex:listingOrder 0 .
```

*Example 162. Relational database*





#### 4.5.4 etymonUnit

Represents a form (typically a word) which is the etymological origin of the headword, or another etymologically related form.

*Property of*

- [etymon](#)

*Properties*

- langCode REQUIRED (exactly one) and UNIQUE (in combination with other unique properties if present). An IETF tag. The tags can be explained and constrained using the [etymonLanguage](#) object type.
- text REQUIRED (exactly one) and UNIQUE (in combination with other unique properties if present). Normalised string. The written form of the etymon.
- reconstructed OPTIONAL (zero or one). Boolean. If present and set to true, indicates that the form is reconstructed and not attested in any corpus.

- [partOfSpeech](#) OPTIONAL (zero or more).
- [translation](#) OPTIONAL (zero or one). [Normalised string](#). A translation or gloss of the etymon in the language of the lexicographic resource.
- [listingOrder](#) REQUIRED (exactly one). Number. The position of this etymon among other etymons of the origin. [Can This can](#) be implicit from the serialization.

**Example 163. XML**

```
<etymonUnit langCode="..." reconstructed="true">
  <text>...</text>
  <partOfSpeech tag="..."/>
  <translation>...</translation>
</etymonUnit>
```

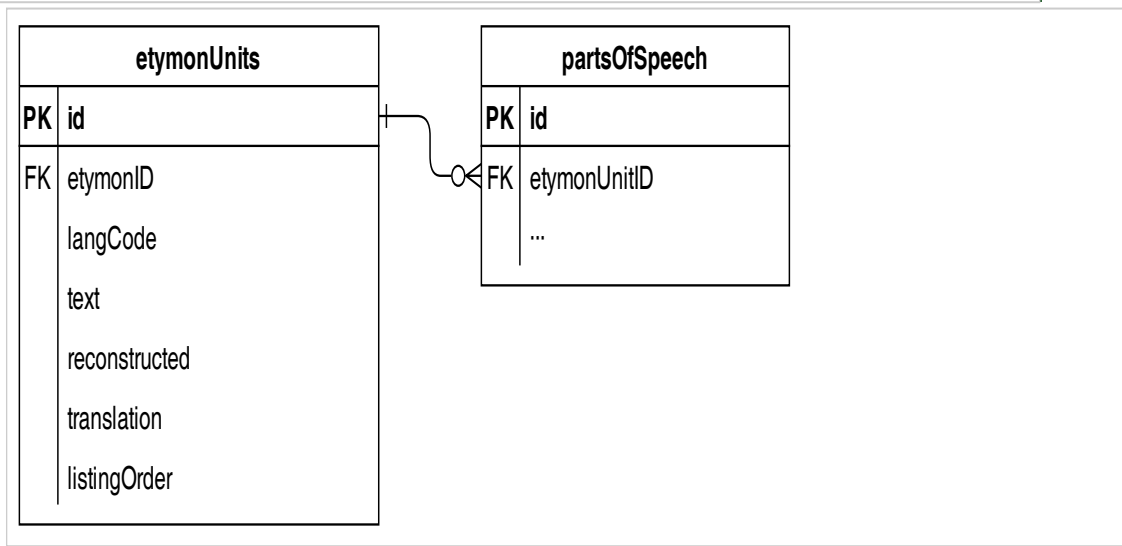
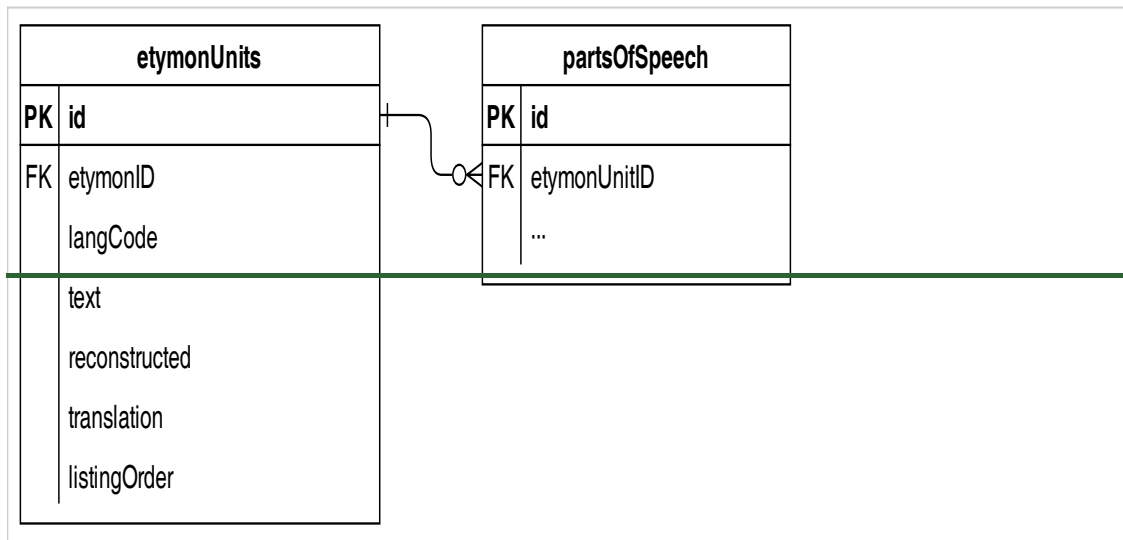
**Example 164. JSON**

```
{
  "langCode": "...",
  "text": "...",
  "reconstructed": true,
  "partsOfSpeech": [...],
  "translation": "..."
}
```

**Example 165. RDF**

```
<#etymonUnit>
  dmlex:langCode "...";
  dmlex:text "...";
  dmlex:reconstructed true;
  dmlex:partOfSpeech "...";
  dmlex:translation ...;
  dmlex:listingOrder 0 .
```

**Example 166. Relational database**



#### 4.5.5 Extensions to `lexicographicResource`

Extends the [lexicographicResource](#) object type from the [Core](#).

##### *Additional properties*

- [etymonLanguage](#) OPTIONAL (zero or more)
- [etymonType](#) OPTIONAL (zero or more)

##### *Example 167. XML*

```
<lexicographicResource>
  <etymonLanguage>...</etymonLanguage/>
  <etymonType>...</etymonType/>
  ...
</lexicographicResource>
```

*Example 168. JSON*

```
{
  "etymonLanguages": [...],
  "etymonTypes": [...],
  ...
}
```

*Example 169. RDF*

```
<#lexicographicResource> dmlex:etymonLanguage ... .
```

### 4.5.6 etymonType

Represents one of possible values for the type of [etymon](#).

*Property of*

- [lexicographicResource](#)

*Properties*

- type REQUIRED (exactly one) and UNIQUE. ~~Non-empty~~ **Normalised** string.
- description OPTIONAL (zero or one). ~~Non-empty~~ **Normalised** string. A human-readable explanation of this type.

*Example 170. XML*

```
<etymonType type="...">
  <description>...</description>
</etymonType>
```

*Example 171. JSON*

```
{
  "type": "...",
  "description": "..."
}
```

*Example 172. RDF*

```
<#lexicographicResource> dmlex:etymonType [
  dmlex:type "...";
  dmlex:description "..." ] .
```

Example 173. Relational database

etymonTypes	
PK	type
FK	lexicographicResourceID description

etymonTypes	
PK	type
FK	lexicographicResourceID description

#### 4.5.7 etymonLanguage

Represents one of several allowed values for the `language` property of `etymonUnit` objects.

Property of

- [lexicographicResource](#)

Properties

- `langCode` REQUIRED (exactly one) and UNIQUE. The IETF language code of the language.
- `displayName` OPTIONAL (zero or one). Normalised string. The name of the etymon language, in the language of the lexicographic resource.

#### Note

Etymologies frequently refer to languages that are not covered by ISO standards. It may be necessary to avail of private-use subtags when composing IETF language tags for etymology, and to explain their meaning in the `displayName`.

Example 174. XML

```
<etymonLanguage langCode="...">  
  <displayName>...</displayName>  
</etymonLanguage>
```

Example 175. JSON

```
{
  "langCode": "...",
  "displayName": "..."}
}
```

Example 176. RDF

```
<#etymonLanguage> dmlex:langCode "...";
  dmlex:displayName "... " .
```

Example 177. Relational database

etymonLanguages	
PK	langCode
FK	lexicographicResourceID displayName

etymonLanguages	
PK	langCode
FK	lexicographicResourceID displayName

#### 4.5.8 Extensions to `partOfSpeech`

Extends the [partOfSpeech](#) object type from the [Core](#).

Can additionally be a property of

- [etymonUnit](#)

Example 178. Relational database

partsOfSpeech	
PK	id
FK	... etymonUnitID

partsOfSpeech	
PK	id
	...
FK	etymonUnitID

## 5 DMLex REQUIRED Serializations (Normative)

### 5.1 DMLex XML serialization (Normative)

This section specifies a serialization of the DMLex data model in XML (Extensible Markup Language) [XML].

For data exchange purposes, a valid DMLex XML document is one where either a `lexicographicResource` or an `entry` is the document element or one with an arbitrary document element other than `lexicographicResource` or `entry` containing a concatenation of one (and only one) of these two element types.

#### 5.1.1 Design Principles (Informative)

The XML serialization of DMLex shown in this document follows these principles:

- The top-level `lexicographicResource` and `entry` object ~~is~~ types are implemented as an XML ~~element~~ elements.
- All other ~~objects~~ object types and their properties are implemented as XML attributes of their parents, unless:
  - the object has an arity other than (0..1) and (1..1) in at least one of its parent objects
  - or the object can have child objects
  - or the object's value is human-readable text, such as a headword or a definition. (this applies, among other things, to all properties named text and description).

In such cases the object or property is implemented as a child XML element of its parent.

#### 5.1.2 Whitespace handling (Normative)

Some XML elements in all this serialization are defined as allowed to have text nodes content while other elements are not allowed to have text content.

- Elements which are not allowed to have text content: whitespace inside these elements should always be collapsed by rules defined in the XML Schema W3C Recommendation [1].
- Elements which are allowed to have text content (which includes elements implementing object types from the Annotation Module): whitespace inside these elements should be handled as follows when reading the XML:
  1. Completely remove all leading whitespace that immediately follows an opening tag.
  2. Completely remove all trailing whitespace that immediately precedes a closing tag.
  - 4-3. Replace all remaining sequences of whitespace should be reduced to with a single space.

- ~~all leading and trailing spaces should be removed~~

~~unless otherwise stated~~ This ensures that whitespace is handled in accordance with DMLex's concept of *normalised strings* defined in [Section 1.3.2, "Properties"](#)~~the~~.

## **5.1.3 Element order (Normative)**

The child elements of each parent element must be in the order in which they are listed in this specification ~~of individual XML elements.~~

The only exception is inline markup elements which implement object types from the Anotation Module: `<placeholderMarker>`, `<headwordMarker>` and `<collocateMarker>`. These can appear in any order.

## **5.1.24 DMLex XML serialization elements (Normative)**

### **5.1.24.1 XML element: `<lexicographicResource>`**

Implements the [lexicographicResource](#) object type.

#### *Attributes*

- @title OPTIONAL
- @uri ~~REQUIRED~~ **OPTIONAL**
- @langCode REQUIRED

#### *Child elements*

- [<entry>](#) OPTIONAL (zero or more)

#### *Child elements if implementing the Crosslingual Module*

- [<translationLanguage>](#) REQUIRED (one or more)

#### *Child elements if implementing the Controlled Values Module*

- [<definitionTypeTag>](#) OPTIONAL (zero or more)
- [<inflectedFormTag>](#) OPTIONAL (zero or more)
- [<labelTag>](#) OPTIONAL (zero or more)
- [<labelTypeTag>](#) OPTIONAL (zero or more)
- [<partOfSpeechTag>](#) OPTIONAL (zero or more)
- [<sourceIdentityTag>](#) OPTIONAL (zero or more)

#### *Child elements if implementing the Linking Module*

- [<relation>](#) OPTIONAL (zero or more)
- [<relationType>](#) OPTIONAL (zero or more)

#### *Child elements if implementing the Etymology Module*

- [<etymonLanguage>](#) OPTIONAL (zero or more)
- [<etymonType>](#) OPTIONAL (zero or more)



#### 5.1.24.2 XML element: <entry>

Implements the [entry](#) object type.

##### Attributes

- @homographNumber OPTIONAL

##### Attributes if implementing the Linking Module

- @id OPTIONAL. Entries that have an ID [may use this ID as object reference instead of the IRI addressing mechanism described in Section 3.2.1, "DMLex fragment identification strings"](#) ~~can be members of relations.~~

##### Child elements

- <headword> REQUIRED (exactly one). If implementing the Annotation Module <headword> can contain a mixture of plain text and [placeholderMarker](#) elements ~~and whitespace inside it should be preserved.~~
- <partOfSpeech> OPTIONAL (zero or more)
- <label> OPTIONAL (zero or more)
- <pronunciation> OPTIONAL (zero or more)
- <inflectedForm> OPTIONAL (zero or more)
- <sense> OPTIONAL (zero or more)

##### Child elements if implementing the Etymology Module

- <etymology> OPTIONAL (zero or more)

#### 5.1.24.3 XML element: <partOfSpeech>

Implements the [partOfSpeech](#) object type.

##### Attributes

- @tag REQUIRED

#### 5.1.24.4 XML element: <inflectedForm>

Implements the [inflectedForm](#) object type.

##### Attributes

- @tag OPTIONAL

##### Child elements

- <text> REQUIRED (exactly one)
- <label> OPTIONAL (zero or more)
- <pronunciation> OPTIONAL (zero or more)

#### 5.1.24.5 XML element: <sense>

Implements the [sense](#) object type.

##### Child elements

- <indicator> OPTIONAL (zero or one)
- <label> OPTIONAL (zero or more)
- <definition> OPTIONAL (zero or more)
- <example> OPTIONAL (zero or more)

##### Attributes if implementing the Linking Module

- @id OPTIONAL. Senses that have an ID [may use this ID as object reference instead of the IRI addressing mechanism described in Section 3.2.1, "DMLex fragment identification strings" can be members of relations.](#)

##### Child elements if implementing the Crosslingual Module

- <headwordExplanation> OPTIONAL (zero or more)
- <headwordTranslation> OPTIONAL (zero or more)

#### 5.1.24.6 XML element: <definition>

Implements the [definition](#) object type.

##### Attributes

- @definitionType OPTIONAL

##### Text content

##### Child elements

- <text> REQUIRED, implements the text property. (exactly one). If implementing the Annotation Module <definitionText> can contain a mixture of plain text, <headwordMarker> elements and elements and <collocateMarker> whitespace inside it should be preserved. elements.

#### 5.1.24.7 XML element: <label>

Implements the [label](#) object type.

##### Attributes

- @tag REQUIRED

#### 5.1.24.8 XML element: <pronunciation>

Implements the [pronunciation](#) object type.

#### Attributes

- @soundFile OPTIONAL if transcription is present, REQUIRED otherwise

#### Child elements

- [<transcription>](#) OPTIONAL (zero or more) if @soundFile is present, REQUIRED (one or more) otherwise
- [<label>](#) OPTIONAL (zero or more)

#### 5.1.24.9 XML element: [<transcription>](#)

Implements the [transcription](#) object type.

#### Attributes

- @scheme OPTIONAL

#### ~~Text content~~

#### ~~Child elements~~

- ~~[<text>](#) REQUIRED, implements the text property. (exactly one).~~

#### 5.1.24.10 XML element: [<example>](#)

Implements the [example](#) object type.

#### Attributes

- @sourceIdentity OPTIONAL
- @sourceElaboration OPTIONAL
- @soundFile OPTIONAL

#### Child elements

- [<text>](#) REQUIRED (exactly one). If implementing the Annotation Module [<text>](#) can contain a mixture of plain text, [<headwordMarker>](#) elements and [<collocateMarker>](#) elements ~~and whitespace inside it should be preserved.~~
- [<label>](#) OPTIONAL (zero or more)

#### Child elements if implementing the Crosslingual Module

- [<exampleTranslation>](#) OPTIONAL (zero or more)

#### 5.1.24.11 XML element: [<translationLanguage>](#)

Implements the [translationLanguage](#) object type from the Crosslingual Module.

#### Attributes

- @langCode REQUIRED

#### 5.1.24.12 XML element: <headwordTranslation>

Implements the [headwordTranslation](#) object type from the Crosslingual Module.

##### Attributes

- @langCode OPTIONAL if the ancestor <lexicographicResource> contains exactly one <translationLanguage>, REQUIRED otherwise

##### Child elements

- <text> REQUIRED (exactly one). If implementing the Annotation Module <text> can contain a mixture of plain text and <placeholderMarker> elements ~~and whitespace inside it should be preserved.~~
- <partOfSpeech> OPTIONAL (zero or more)
- <label> OPTIONAL (zero or more)
- <pronunciation> OPTIONAL (zero or more)
- <inflectedForm> OPTIONAL (zero or more)

#### 5.1.24.13 XML element: <headwordExplanation>

Implements the [headwordExplanation](#) object type from the Crosslingual Module.

##### Attributes

- @langCode OPTIONAL if the ancestor <lexicographicResource> contains exactly one <translationLanguage>, REQUIRED otherwise

##### ~~Text content~~

##### ~~Child elements~~

- ~~<text> REQUIRED, implements the text property. (exactly one).~~

#### 5.1.24.14 XML element: <exampleTranslation>

Implements the [exampleTranslation](#) object type from the Crosslingual Module.

##### Attributes

- @langCode OPTIONAL if the ancestor <lexicographicResource> contains exactly one <translationLanguage>, REQUIRED otherwise
- @soundFile OPTIONAL

##### Child elements

- <text> REQUIRED (exactly one). If implementing the Annotation Module <text> can contain a mixture of plain text, <headwordMarker> elements and <collocateMarker> elements ~~and whitespace inside it should be preserved.~~

- [<label>](#) OPTIONAL (zero or more)

#### 5.1.24.15 XML element: [<partOfSpeechTag>](#)

Implements the [partOfSpeechTag](#) object type from the Controlled Values Module.

##### Attributes

- @tag REQUIRED
- @for OPTIONAL

##### Child elements

- <description> OPTIONAL (zero or one)

#### 5.1.24.16 XML element: [<inflectedFormTag>](#)

Implements the [inflectedFormTag](#) object type from the Controlled Values Module.

##### Attributes

- @tag REQUIRED
- @for OPTIONAL

##### Child elements

- <description> OPTIONAL (zero or one)
- [<sameAs>](#) OPTIONAL (zero or more)

#### 5.1.24.17 XML element: [<definitionTypeTag>](#)

Implements the [definitionTypeTag](#) object type from the Controlled Values Module.

##### Attributes

- @tag REQUIRED

##### Child elements

- <description> OPTIONAL (zero or one)
- [<sameAs>](#) OPTIONAL (zero or more)

#### 5.1.24.18 XML element: [<labelTag>](#)

Implements the [labelTag](#) object type from the Controlled Values Module.

##### Attributes

- @tag REQUIRED
- @typeTag OPTIONAL

- @for OPTIONAL

*Child elements*

- <description> OPTIONAL (zero or one)
- [<sameAs>](#) OPTIONAL (zero or more)

**5.1.24.19 XML element: <labelTypeTag>**

Implements the [labelTypeTag](#) object type from the Controlled Values Module.

*Attributes*

- @tag REQUIRED

*Child elements*

- <description> OPTIONAL (zero or one)
- [<sameAs>](#) OPTIONAL (zero or more)

**5.1.24.20 XML element: <sourceIdentityTag>**

Implements the [sourceIdentityTag](#) object type from the Controlled Values Module.

*Attributes*

- @tag REQUIRED

*Child elements*

- <description> OPTIONAL (zero or one)
- [<sameAs>](#) OPTIONAL (zero or more)

**5.1.24.21 XML element: <transcriptionSchemeTag>**

Implements the [transcriptionSchemeTag](#) object type from the Controlled Values Module.

*Attributes*

- @tag REQUIRED
- @for OPTIONAL

*Child elements*

- <description> OPTIONAL (zero or one)

**5.1.24.22 XML element: <sameAs>**

Implements the [sameAs](#) object type from the Controlled Values Module.

#### Attributes

- @uri REQUIRED

#### 5.1.24.23 XML element: <relation>

Implements the [relation](#) object type from the Linking Module.

#### Attributes

- @type REQUIRED

#### Child elements

- <description> OPTIONAL (zero or one)
- [<member>](#) REQUIRED (two or more)

#### 5.1.24.24 XML element: <member>

Implements the [member](#) object type from the Linking Module.

#### Attributes

- @ref REQUIRED: the id of an entry, a sense or a collocateMarker
- @role OPTIONAL
- @obverseListingOrder OPTIONAL

#### 5.1.24.25 XML element: <relationType>

Implements the [relationType](#) object type from the Linking Module.

#### Attributes

- @type REQUIRED
- @scopeRestriction REQUIRED

#### Child elements

- <description> OPTIONAL (zero or one)
- [<memberType>](#) OPTIONAL (zero or more)
- [<sameAs>](#) OPTIONAL (zero or more)

#### 5.1.24.26 XML element: <memberType>

Implements the [memberType](#) object type from the Linking Module.

#### Attributes

- @role OPTIONAL
- @type REQUIRED

- @min OPTIONAL
- @max OPTIONAL
- @hint OPTIONAL

#### Child elements

- <description> OPTIONAL (zero or one)
- [<sameAs>](#) OPTIONAL (zero or more)

#### 5.1.24.27 XML element: [<placeholderMarker>](#)

Implements the [placeholderMarker](#) object type from the Annotation module. Contains the text which it marks up.

#### 5.1.24.28 XML element: [<headwordMarker>](#)

Implements the [headwordMarker](#) object type from the Annotation module. Contains the text which it marks up.

#### 5.1.24.29 XML element: [<collocateMarker>](#)

Implements the [collocateMarker](#) object type from the Annotation module. Contains the text which it marks up, optionally followed by one or more [<label>](#) elements.

#### Attributes

- @lemma OPTIONAL

#### Attributes if implementing the Linking Module

- @id OPTIONAL. Collocate markers that have an ID [may use this ID as object reference instead of the IRI addressing mechanism described in Section 3.2.1, "DMLex fragment identification strings"](#)~~can appear as members of relations.~~

#### Child elements

- [<label>](#) OPTIONAL (zero or more)

#### 5.1.24.30 XML element: [<etymology>](#)

Implements the [etymology](#) object type from the Etymology Module.

#### Child elements

- <description> OPTIONAL (zero or one)
- [<etymon>](#) OPTIONAL (zero or more)

#### 5.1.24.31 XML element: [<etymon>](#)

Implements the [etymon](#) object type from the Etymology Module.



#### Attributes

- @when OPTIONAL
- @type OPTIONAL

#### Child elements

- <note> OPTIONAL (zero or one)
- [<etymonUnit>](#) REQUIRED (one or more)
- <translation> OPTIONAL (zero or one)

#### 5.1.24.32 XML element: [<etymonUnit>](#)

Implements the [etymonUnit](#) object type from the Etymology Module.

#### Attributes

- @langCode REQUIRED
- @reconstructed OPTIONAL

#### Child elements

- <text> REQUIRED (exactly one)
- [<partOfSpeech>](#) OPTIONAL (zero or more)
- <translation> OPTIONAL (zero or one)

#### 5.1.24.33 XML element: [<etymonType>](#)

Implements the [etymonType](#) object type from the Etymology Module.

#### Attributes

- @type REQUIRED

#### Child elements

- <description> OPTIONAL (zero or one)

#### 5.1.24.34 XML element: [<etymonLanguage>](#)

Implements the [etymonLanguage](#) object type from the Etymology Module.

#### Attributes

- @langCode REQUIRED

#### Child elements

- <displayName> OPTIONAL (zero or one)

## 5.2 DMLex JSON serialization (Normative)

This section specifies a serialization of the DMLex data model in JSON (JavaScript Object Notation) [JSON].

For data exchange purposes, a valid DMLex JSON file contains either a `lexicographicResource` object or an `entry` object or a concatenation of one (and only one) of these two object types in the JSON Lines format (i.e. separated by a system-specific newline delimiter).

### 5.2.1 Design principles

The JSON serialization of DMLex shown in this document follows these principles:

- The top-level `lexicographicResource` object is implemented as a JSON object: `{...}`.
- All other objects are implemented as JSON name-value pairs inside their parent JSON object: `{"name": ...}`.
- The values of objects are implemented:
  - If the object has an arity of (0..1) or (1..1):
    - If the object cannot have any child objects: as a string or number.
    - If the object can have child objects: as a JSON object.
  - If the object has any other arity:
    - If the object cannot have any child objects: as an array of strings or numbers.
    - If the object can have child objects: as an array of JSON objects.

The names of objects are:

- plural if the value is an array, for example `labels`, `partsOfSpeech`
- singular if the value is anything else, for example `headword`, `indicator`

### 5.2.2 DMLex JSON serialization objects (Normative)

#### 5.2.2.1 JSON object: `lexicographicResource`

Implements the `lexicographicResource` object type.

##### Members

- "title" OPTIONAL, string
- "uri" ~~REQUIRED~~OPTIONAL, string
- "langCode" REQUIRED, string
- "entries" OPTIONAL, array of zero or more `entry` instances

##### Members if implementing the Crosslingual Module

- "translationLanguages" REQUIRED, array of one or more strings implementing the `translationLanguage` object type

##### Members if implementing the Controlled Values Module

- "definitionTypeTags" OPTIONAL, array of zero or more `definitionTypeTag` instances
- "inflectedFormTags" OPTIONAL, array of zero or more `inflectedFormTag` instances
- "labelTags" OPTIONAL, array of zero or more `labelTag` instances

- "labelTypeTags" OPTIONAL, array of zero or more [labelTypeTag](#) instances
- "partOfSpeechTags" OPTIONAL, array of zero or more [partOfSpeechTag](#) instances
- "sourceIdentityTags" OPTIONAL, array of zero or more [sourceIdentityTag](#) instances

*Members if implementing the Linking Module*

- "relations" OPTIONAL, array of zero or more [relation](#) instances
- "relationTypes" OPTIONAL, array of zero or more [relationType](#) instances

*Members if implementing the Etymology Module*

- "etymonLanguages" OPTIONAL, array of zero or more [etymonLanguage](#) instances
- "etymonTypes" OPTIONAL, array of zero or more [etymonType](#) instances

### 5.2.2.2 JSON object: *entry*

Implements the [entry](#) object type.

*Members*

- "headword" REQUIRED, string
- "homographNumber" OPTIONAL, string
- "partsOfSpeech" OPTIONAL, array of zero or more strings implementing the [partOfSpeech](#) object type
- "labels" OPTIONAL, array of zero or more strings implementing the [label](#) object type
- "pronunciations" OPTIONAL, array of zero or more [pronunciation](#) instances
- "inflectedForms" OPTIONAL, array of zero or more [inflectedForm](#) instances
- "senses" OPTIONAL, array of zero or more [sense](#) instances

*Members if implementing the Linking Module*

- "id" OPTIONAL, string. Entries that have an ID ~~can be members~~ [may use this ID as object reference instead of the IRI addressing mechanism described in Section 3.2.1, "DMLex fragment identification strings"](#) ~~relations.~~

*Members if implementing the Annotation Module*

- "placeholderMarkers" OPTIONAL, array of zero or more [placeholderMarker](#) instances

*Members if implementing the Etymology Module*

- "etymologies" OPTIONAL, array of zero or more [etymology](#) instances

### 5.2.2.3 JSON object: *inflectedForm*

Implements the [inflectedForm](#) object type.

*Members*

- "tag" OPTIONAL, string
- "text" REQUIRED, string
- "labels" OPTIONAL, array of zero or more strings implementing the [label](#) object type
- "pronunciations" OPTIONAL, array of zero or more [pronunciation](#) instances

#### 5.2.2.4 JSON object: *sense*

Implements the [sense](#) object type.

##### *Members*

- "indicator" OPTIONAL, string
- "labels" OPTIONAL, array of zero or more strings implementing the [label](#) object type
- "definitions" OPTIONAL, array of zero or more [definition](#) instances
- "examples" OPTIONAL, array of zero or more [example](#) instances

##### *Members if implementing the Linking Module*

- "id" OPTIONAL, string. Senses that have an ID ~~can be members~~ [may use this ID as object reference instead of the IRI addressing mechanism described in Section 3.2.1, "DMLex fragment identification strings"](#) ~~relations.~~

##### *Members if implementing the Crosslingual Module*

- "headwordExplanations" OPTIONAL, array of zero or more [headwordExplanation](#) instances
- "headwordTranslations" OPTIONAL, array of zero or more [headwordTranslation](#) instances

#### 5.2.2.5 JSON object: *definition*

Implements the [definition](#) object type.

##### *Members*

- "definitionType" OPTIONAL, string
- "text" REQUIRED, string

##### *Members if implementing the Annotation Module*

- "headwordMarkers" OPTIONAL, array of zero or more [headwordMarker](#) instances
- "collocateMarkers" OPTIONAL, array of zero or more [collocateMarker](#) instances

#### 5.2.2.6 JSON object: *pronunciation*

Implements the [pronunciation](#) object type.

##### *Members*

- "soundFile" OPTIONAL if "transcriptions" is present and non-empty, REQUIRED otherwise; string

- "transcriptions" OPTIONAL if "soundFile" is present, REQUIRED otherwise; array of zero or more [transcription](#) instances
- "labels" OPTIONAL, array of zero or more strings implementing the [label](#) object type

#### 5.2.2.7 JSON object: *transcription*

Implements the [transcription](#) object type.

##### *Members*

- "scheme" OPTIONAL, string
- "text" REQUIRED, string

#### 5.2.2.8 JSON object: *example*

Implements the [example](#) object type.

##### *Members*

- "sourceIdentity" OPTIONAL, string
- "sourceElaboration" OPTIONAL, string
- "soundFile" OPTIONAL, string
- "text" REQUIRED, string
- "labels" OPTIONAL, array of zero or more strings implementing the [label](#) object type

##### *Members if implementing the Annotation Module*

- "headwordMarkers" OPTIONAL, array of zero or more [headwordMarker](#) instances
- "collocateMarkers" OPTIONAL, array of zero or more [collocateMarker](#) instances

##### *Members if implementing the Crosslingual Module*

- "exampleTranslations" OPTIONAL, array of zero or more [exampleTranslation](#) instances

#### 5.2.2.9 JSON object: *headwordTranslation*

Implements the [headwordTranslation](#) object type from the Crosslingual Module.

##### *Members*

- "langCode" OPTIONAL if the `lexicographicResource` contains exactly one `translationLanguage`, REQUIRED otherwise; string
- "text" REQUIRED, string
- "partsOfSpeech" OPTIONAL, array of zero or more strings implementing the [partOfSpeech](#) object type
- "labels" OPTIONAL, array of zero or more strings implementing the [label](#) object type
- "pronunciations" OPTIONAL, array of zero or more [pronunciation](#) instances
- "inflectedForms" OPTIONAL, array of zero or more [inflectedForm](#) instances

##### *Members if implementing the Annotation Module*

- "placeholderMarkers" OPTIONAL, array of zero or more [placeholderMarker](#) instances

#### 5.2.2.10 JSON object: *headwordExplanation*

Implements the [headwordExplanation](#) object type from the Crosslingual Module.

##### Members

- "langCode" OPTIONAL if the `lexicographicResource` contains exactly one `translationLanguage`, REQUIRED otherwise; string
- "text" REQUIRED, string

#### 5.2.2.11 JSON object: *exampleTranslation*

Implements the [exampleTranslation](#) object type from the Crosslingual Module.

##### Members

- "langCode" OPTIONAL if the `lexicographicResource` contains exactly one `translationLanguage`, REQUIRED otherwise; string
- "text" REQUIRED, string
- "labels" OPTIONAL, array of zero or more strings implementing the [label](#) object type

##### Members if implementing the Annotation Module

- "headwordMarkers" OPTIONAL, array of zero or more [headwordMarker](#) instances
- "collocateMarkers" OPTIONAL, array of zero or more [collocateMarker](#) instances

#### 5.2.2.12 JSON object: *partOfSpeechTag*

Implements the [partOfSpeechTag](#) object type from the Controlled Values Module.

##### Members

- "tag" REQUIRED, string
- "description" OPTIONAL, string
- "for" OPTIONAL, string
- "sameAs" OPTIONAL, array of zero or more strings implementing the [sameAs](#) object type

#### 5.2.2.13 JSON object: *inflectedFormTag*

Implements the [inflectedFormTag](#) object type from the Controlled Values Module.

##### Members

- "tag" REQUIRED, string
- "description" OPTIONAL, string
- "for" OPTIONAL, string
- "sameAs" OPTIONAL, array of zero or more strings implementing the [sameAs](#) object type

#### 5.2.2.14 JSON object: *definitionTypeTag*

Implements the [definitionTypeTag](#) object type from the Controlled Values Module.

##### Members

- "tag" REQUIRED, string
- "description" OPTIONAL, string
- "sameAs" OPTIONAL, array of zero or more strings implementing the [sameAs](#) object type

#### 5.2.2.15 JSON object: *labelTag*

Implements the [labelTag](#) object type from the Controlled Values Module.

##### Members

- "tag" REQUIRED, string
- "typeTag" OPTIONAL, string
- "description" OPTIONAL, string
- "for" OPTIONAL, string
- "sameAs" OPTIONAL, array of zero or more strings implementing the [sameAs](#) object type

#### 5.2.2.16 JSON object: *labelTypeTag*

Implements the [labelTypeTag](#) object type from the Controlled Values Module.

##### Members

- "tag" REQUIRED, string
- "description" OPTIONAL, string
- "sameAs" OPTIONAL, array of zero or more strings implementing the [sameAs](#) object type

#### 5.2.2.17 JSON object: *sourceIdentityTag*

Implements the [sourceIdentityTag](#) object type from the Controlled Values Module.

##### Members

- "tag" REQUIRED, string
- "description" OPTIONAL, string
- "sameAs" OPTIONAL, array of zero or more strings implementing the [sameAs](#) object type

#### 5.2.2.18 JSON object: *transcriptionSchemeTag*

Implements the [transcriptionSchemeTag](#) object type from the Controlled Values Module.

##### Members

- "tag" REQUIRED, string
- "description" OPTIONAL, string

- "for" OPTIONAL, string

#### 5.2.2.19 JSON object: *relation*

Implements the [relation](#) object type from the Linking Module.

##### Members

- "type" REQUIRED, string
- "description" OPTIONAL, string
- "members" OPTIONAL, array of two or more [member](#) instances

#### 5.2.2.20 JSON object: *member*

Implements the [member](#) object type from the Linking Module.

##### Members

- "ref" REQUIRED, string: the id of an entry, a sense or a collocateMarker.
- "role" OPTIONAL, string
- "obverseListingOrder" OPTIONAL, number

#### 5.2.2.21 JSON object: *relationType*

Implements the [relationType](#) object type from the Linking Module.

##### Members

- "type" REQUIRED, string
- "scopeRestriction" OPTIONAL, string
- "description" OPTIONAL, string
- "memberTypes" OPTIONAL, array of zero or more [memberType](#) instances
- "sameAs" OPTIONAL, array of zero or more strings implementing the [sameAs](#) object type

#### 5.2.2.22 JSON object: *memberType*

Implements the [memberType](#) object type from the Linking Module.

##### Members

- "role" OPTIONAL, string
- "type" REQUIRED, string
- "min" OPTIONAL, number
- "max" OPTIONAL, number
- "hint" OPTIONAL, string
- "description" OPTIONAL, string
- "sameAs" OPTIONAL, array of zero or more strings implementing the [sameAs](#) object type

#### 5.2.2.23 JSON object: *placeholderMarker*



Implements the [placeholderMarker](#) object type from the Annotation Module.

#### Members

- "startIndex" REQUIRED, number
- "endIndex" REQUIRED, number

#### 5.2.2.24 JSON object: *headwordMarker*

Implements the [headwordMarker](#) object type from the Annotation Module.

#### Members

- "startIndex" REQUIRED, number
- "endIndex" REQUIRED, number

#### 5.2.2.25 JSON object: *collocateMarker*

Implements the [collocateMarker](#) object type from the Annotation Module.

#### Members

- "startIndex" REQUIRED, number
- "endIndex" REQUIRED, number
- "lemma" OPTIONAL, string
- "labels" OPTIONAL, array of zero or more strings implementing the [label](#) object type

#### Members if implementing the Linking Module

- "id" OPTIONAL, string. Collocate markers that have an ID [may use this ID as object reference instead of the IRI addressing mechanism described in Section 3.2.1, "DMLex fragment identification strings"](#)~~can appear as members of relations.~~

#### 5.2.2.26 JSON object: *etymology*

Implements the [etymology](#) object type from the Etymology Module.

#### Members

- "description" OPTIONAL, string
- "etymons" OPTIONAL, array of zero or more [etymon](#) instances

#### 5.2.2.27 JSON object: *etymon*

Implements the [etymon](#) object type from the Etymology Module.

#### Members

- "when" OPTIONAL, string
- "type" OPTIONAL, string
- "note" OPTIONAL, string

- "etymonUnits" REQUIRED, array of one or more [etymonUnit](#) instances
- "translation" OPTIONAL, string

#### 5.2.2.28 JSON object: *etymonUnit*

Implements the [etymonUnit](#) object type from the Etymology Module.

##### Members

- "langCode" REQUIRED, string
- "reconstructed" OPTIONAL, boolean
- "text" REQUIRED, string
- "partsOfSpeech" OPTIONAL, array of zero or more strings implementing the [partOfSpeech](#) object type
- "translation" OPTIONAL, string

#### 5.2.2.29 JSON object: *etymonType*

Implements the [etymonType](#) object type from the Etymology Module.

##### Members

- "type" REQUIRED, string
- "description" OPTIONAL, string

#### 5.2.2.30 JSON object: *etymonLanguage*

Implements the [etymonLanguage](#) object type from the Etymology Module.

##### Members

- "langCode" REQUIRED, string
- "displayName" OPTIONAL, string

### 5.3 DMLex RDF serialization (Normative)

#### 5.3.1 Design principles

The RDF serialization used in this document follows the following principles

- The RDF serialization uses an vocabulary defined at the namespace <https://www.docs.oasis-open.org/etymon/confirmed/lexidma/dmlex/v1.0/schemas/RDF/dmlex.ttl#>, all elements of the vocabulary are in this namespace
- The vocabulary file provides RDF Schema and OWL axioms which implement the restrictions described by the data model. It also provides a linking to the [OntoLex-Lemon](#) model. The ontology provides rules to infer an OntoLex-lemon model from DMLex data. This is achieved by means of subclass and subproperty axioms and so conversion can be performed by an OWL reasoner.
- All elements of the modules are modelled as individuals with an appropriate class definitions
- Each other object type is associated with a property (with a lowercase initial letter) and a class (with an uppercase initial letter). In all case the range of this property is the class

- Arity of properties is implemented by means of OWL class restrictions.
- Listing order can be specified with the property `dmlex:listingOrder` with values starting from 1 and ascending
- There is not a specific `id` property, instead the URI of the entity acts as the ID
- There is no property for [sameAs](#) in the controlled values module. Instead you can use OWL's `sameAs` property
- The implementation is not aware of which modules are in use, to enable publishing on the web. As such cardinality constraints that are only required in a module are not implemented.
- The domain of some properties is given as a `HasProperty` class when it can be used with multiple classes. As such, classes may be subclasses of several different `HasProperty` classes

## 5.3.2 DMLex RDF serialization elements

### 5.3.2.1 RDF element: `dmlex:LexicographicResource`

Represents a dictionary

#### Superclasses

- <http://www.w3.org/ns/lemon/lime#Lexicon>

#### Properties

- `dmlex:title` OPTIONAL (at most 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- `dmlex:uri` OPTIONAL (at most 1) of type <http://www.w3.org/2001/XMLSchema#anyURI>
- `dmlex:entry` OPTIONAL reference to [Entry](#) (subproperty of <http://www.w3.org/ns/lemon/lime#entry>)
- `dmlex:translationLanguage` OPTIONAL reference to [TranslationLanguage](#)
- `dmlex:definitionTypeTag` OPTIONAL reference to [DefinitionTypeTag](#)
- `dmlex:inflectedFormTag` OPTIONAL reference to [InflectedFormTag](#)
- `dmlex:labelTypeTag` OPTIONAL reference to [LabelTypeTag](#)
- `dmlex:partOfSpeechTag` OPTIONAL reference to [PartOfSpeechTag](#)
- `dmlex:sourceIdentityTag` OPTIONAL reference to [SourceIdentityTag](#)
- `dmlex:transcriptionSchemeTag` OPTIONAL reference to [TranscriptionSchemeTag](#)
- `dmlex:relation` OPTIONAL reference to [Relation](#)
- `dmlex:relationType` OPTIONAL reference to [RelationType](#)
- `dmlex:etymonType` OPTIONAL reference to [EtymonType](#)
- `dmlex:etymonLanguage` OPTIONAL reference to [EtymonLanguage](#)
- `dmlex:languageLangCode` REQUIRED (exactly 1) of type <http://www.w3.org/2001/XMLSchema#language> (subproperty of <http://www.w3.org/ns/lemon/lime#language>)

### 5.3.2.2 RDF element: `dmlex:Entry`

Represents a dictionary entry

#### Superclasses

- <http://www.w3.org/ns/lemon/ontolex#LexicalEntry>

## Properties

- `dmlex:headword` REQUIRED (exactly 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- `dmlex:homographNumber` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:sense` OPTIONAL reference to [Sense](#) (subproperty of `http://www.w3.org/ns/lemon/ontolex#evokes`)
- `dmlex:etymology` OPTIONAL reference to [Etymology](#)
- `dmlex:partOfSpeech` OPTIONAL reference to [PartOfSpeech](#)
- `dmlex:label` OPTIONAL reference to [Label](#)
- `dmlex:pronunciation` OPTIONAL reference to [Pronunciation](#)
- `dmlex:inflectedForm` OPTIONAL reference to [InflectedForm](#) (subproperty of `http://www.w3.org/ns/lemon/ontolex#otherForm`)
- `dmlex:placeholderMarker` OPTIONAL reference to [PlaceholderMarker](#)

### 5.3.2.3 RDF element: `dmlex:PartOfSpeech`

Represents a part of speech label

## Properties

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:tag` REQUIRED (exactly 1)

### 5.3.2.4 RDF element: `dmlex:InflectedForm`

Represents one (of possibly many) inflected forms of the headword

## Superclasses

- `http://www.w3.org/ns/lemon/ontolex#Form`

## Properties

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:label` OPTIONAL reference to [Label](#)
- `dmlex:pronunciation` OPTIONAL reference to [Pronunciation](#)
- `dmlex:tag` OPTIONAL (at most 1)
- `dmlex:text` REQUIRED (exactly 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`

### 5.3.2.5 RDF element: `dmlex:Sense`

Represents one of possibly many meanings (or meaning potentials) of the headword.

## Superclasses

- `http://www.w3.org/ns/lemon/ontolex#LexicalConcept`

### Properties

- `dmlex:listingOrder` REQUIRED (exactly 1) of type <http://www.w3.org/2001/XMLSchema#nonNegativeInteger>
- `dmlex:label` OPTIONAL reference to [Label](#)
- `dmlex:indicator` OPTIONAL (at most 1) ~~reference to~~ [of type http://www.w3.org/2000/01/rdf-schema#Literal](http://www.w3.org/2000/01/rdf-schema#Literal)
- `dmlex:definition` OPTIONAL reference to [Definition](#)
- `dmlex:example` OPTIONAL reference to [Example](#)
- `dmlex:headwordExplanation` OPTIONAL reference to [HeadwordExplanation](#)
- `dmlex:headwordTranslation` OPTIONAL reference to [HeadwordTranslation](#)

#### 5.3.2.6 RDF element: `dmlex:Definition`

Represents one of possibly several definitions of a sense.

### Properties

- `dmlex:listingOrder` REQUIRED (exactly 1) of type <http://www.w3.org/2001/XMLSchema#nonNegativeInteger>
- `dmlex:text` REQUIRED (exactly 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- `dmlex:definitionType` OPTIONAL (at most 1) reference to [DefinitionType](#)
- `dmlex:headwordMarker` OPTIONAL reference to [HeadwordMarker](#)
- `dmlex:collocateMarker` OPTIONAL reference to [CollocateMarker](#)

#### 5.3.2.7 RDF element: `dmlex:Label`

Represents a restriction on its parent such as temporal (old-fashioned, neologism), regional (dialect), register (formal, colloquial), domain (medicine, politics) or grammar (singular-only).

### Properties

- `dmlex:listingOrder` REQUIRED (exactly 1) of type <http://www.w3.org/2001/XMLSchema#nonNegativeInteger>
- `dmlex:tag` REQUIRED (exactly 1)

#### 5.3.2.8 RDF element: `dmlex:Pronunciation`

Represents the pronunciation of its parent. Pronunciations must have either a sound file or a transcription.

### Properties

- `dmlex:listingOrder` REQUIRED (exactly 1) of type <http://www.w3.org/2001/XMLSchema#nonNegativeInteger>
- `dmlex:label` OPTIONAL reference to [Label](#)
- `dmlex:soundFile` OPTIONAL (at most 1) of type <http://www.w3.org/2001/XMLSchema#anyURI>
- `dmlex:transcription` OPTIONAL reference to [Transcription](#)

#### 5.3.2.9 RDF element: `dmlex:Transcription`

Represents the transcription of a pronunciation in some notation such as IPA.

#### *Properties*

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:text` REQUIRED (exactly 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- `dmlex:scheme` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#language`

#### **5.3.2.10 RDF element: `dmlex:Example`**

Represents a sentence or other text fragment which illustrates the headword being used.

#### *Properties*

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:label` OPTIONAL reference to [Label](#)
- `dmlex:text` REQUIRED (exactly 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- `dmlex:soundFile` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#anyURI`
- `dmlex:sourceIdentity` OPTIONAL (at most 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- `dmlex:sourceElaboration` OPTIONAL (at most 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- `dmlex:exampleTranslation` OPTIONAL reference to [ExampleTranslation](#)
- `dmlex:headwordMarker` OPTIONAL reference to [HeadwordMarker](#)
- `dmlex:collocateMarker` OPTIONAL reference to [CollocateMarker](#)

#### **5.3.2.11 RDF element: `dmlex:TranslationLanguage`**

Represents one of the languages in which translations are given in this lexicographic resource

#### *Properties*

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:langCode` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#language` ([subproperty of `http://www.w3.org/ns/lemon/lime#language`](#))

#### **5.3.2.12 RDF element: `dmlex:HeadwordTranslation`**

Represents one of possibly multiple translations of a headword

#### *Properties*

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`

- `dmlex:languageLangCode` OPTIONAL (at most 1) of type <http://www.w3.org/2001/XMLSchema#language> (subproperty of <http://www.w3.org/ns/lemon/lime#language>)
- `dmlex:partOfSpeech` OPTIONAL reference to [PartOfSpeech](#)
- `dmlex:label` OPTIONAL reference to [Label](#)
- `dmlex:pronunciation` OPTIONAL reference to [Pronunciation](#)
- `dmlex:inflectedForm` OPTIONAL reference to [InflectedForm](#) (subproperty of <http://www.w3.org/ns/lemon/ontolex#otherForm>)
- `dmlex:text` REQUIRED (exactly 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- `dmlex:placeholderMarker` OPTIONAL reference to [PlaceholderMarker](#)

#### 5.3.2.13 RDF element: `dmlex:HeadwordExplanation`

Represents a statement in the translation language which explains (but does not translate) the meaning of the headword.

##### Properties

- `dmlex:languageLangCode` OPTIONAL (at most 1) of type <http://www.w3.org/2001/XMLSchema#language> (subproperty of <http://www.w3.org/ns/lemon/lime#language>)
- `dmlex:text` REQUIRED (exactly 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>

#### 5.3.2.14 RDF element: `dmlex:ExampleTranslation`

Represents the translation of an example.

##### Properties

- `dmlex:listingOrder` OPTIONAL of type <http://www.w3.org/2001/XMLSchema#nonNegativeInteger>
- `dmlex:languageLangCode` OPTIONAL (at most 1) of type <http://www.w3.org/2001/XMLSchema#language> (subproperty of <http://www.w3.org/ns/lemon/lime#language>)
- `dmlex:label` OPTIONAL reference to [Label](#)
- `dmlex:text` REQUIRED (exactly 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- `dmlex:soundFile` OPTIONAL of type <http://www.w3.org/2001/XMLSchema#anyURI>
- `dmlex:headwordMarker` OPTIONAL reference to [HeadwordMarker](#)
- `dmlex:collocateMarker` OPTIONAL reference to [CollocateMarker](#)

#### 5.3.2.15 RDF element: `dmlex:DefinitionTypeTag`

Represents one (of many) possible values for definitionType of definition.

##### Properties

- `dmlex:tag` REQUIRED (exactly 1)

- dmlex:description OPTIONAL (at most 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- dmlex:sameAs OPTIONAL

#### 5.3.2.16 RDF element: *dmlex:InflectedFormTag*

Represents one (of many) possible values for tag of inflectedForm

##### Properties

- dmlex:inflectedForm OPTIONAL reference to [InflectedForm](http://www.w3.org/ns/lemon/ontolex#otherForm) (subproperty of <http://www.w3.org/ns/lemon/ontolex#otherForm>)
- dmlex:tag REQUIRED (exactly 1)
- dmlex:description OPTIONAL (at most 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- ~~dmlex:for~~dmlex:forPartOfSpeech OPTIONAL of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- dmlex:sameAs OPTIONAL

#### 5.3.2.17 RDF element: *dmlex:LabelTag*

Represents one (of many) possible values for tag of label.

##### Properties

- dmlex:inflectedForm OPTIONAL reference to [InflectedForm](http://www.w3.org/ns/lemon/ontolex#otherForm) (subproperty of <http://www.w3.org/ns/lemon/ontolex#otherForm>)
- dmlex:tag REQUIRED (exactly 1)
- dmlex:description OPTIONAL (at most 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- ~~• dmlex:tag REQUIRED (exactly 1)~~
- ~~• dmlex:description OPTIONAL (at most 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>~~
- ~~dmlex:for~~dmlex:forPartOfSpeech OPTIONAL of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- dmlex:forCollocates OPTIONAL (at most 1) of type <http://www.w3.org/2001/XMLSchema#boolean>
- dmlex:sameAs OPTIONAL

#### 5.3.2.18 RDF element: *dmlex:LabelTypeTag*

Represents one (of many) possible values for typeTag of labelTag.

##### Properties

- dmlex:tag REQUIRED (exactly 1)
- dmlex:description REQUIRED (exactly 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>



### 5.3.2.19 RDF element: *dmlex:PartOfSpeechTag*

Represents one (of many) possible values for tag of partOfSpeech.

#### Properties

- *dmlex:inflectedForm* OPTIONAL reference to [InflectedForm](http://www.w3.org/ns/lemon/ontolex#otherForm) (subproperty of <http://www.w3.org/ns/lemon/ontolex#otherForm>)
- ~~*dmlex:tag* REQUIRED (exactly 1)~~
- ~~*dmlex:description* OPTIONAL (at most 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>~~
- [\*dmlex:tag\* REQUIRED \(exactly 1\)](#)
- [\*dmlex:description\* OPTIONAL \(at most 1\) of type <http://www.w3.org/2000/01/rdf-schema#Literal>](#)
- *dmlex:forEtymology* OPTIONAL (at most 1) of type <http://www.w3.org/2001/XMLSchema#boolean>
- [\*dmlex:sameAs\* OPTIONAL](#)

### 5.3.2.20 RDF element: *dmlex:SourceIdentityTag*

Represents one (of many) possible values for sourceIdentity of example.

#### Properties

- *dmlex:tag* REQUIRED (exactly 1)
- *dmlex:description* OPTIONAL (at most 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- [\*dmlex:sameAs\* OPTIONAL](#)

### 5.3.2.21 RDF element: *dmlex:TranscriptionSchemeTag*

Represents one (of many) possible values for scheme of transcription.

#### Properties

- *dmlex:inflectedForm* OPTIONAL reference to [InflectedForm](http://www.w3.org/ns/lemon/ontolex#otherForm) (subproperty of <http://www.w3.org/ns/lemon/ontolex#otherForm>)
- *dmlex:tag* REQUIRED (exactly 1)
- *dmlex:description* OPTIONAL (at most 1) of type <http://www.w3.org/2000/01/rdf-schema#Literal>
- ~~*dmlex:for* OPTIONAL (at most 1) of type <http://www.w3.org/2001/XMLSchema#boolean>~~

### 5.3.2.22 RDF element: *dmlex:Relation*

Represents the fact that a relation exists between two or more objects.

#### Properties

- `dmlex:description` OPTIONAL (at most 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- `dmlex:type` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:member` REQUIRED (at least 2) reference to [Member](#)

#### 5.3.2.23 RDF element: `dmlex:Member`

Represents the fact that an object, such as an entry or a sense, is a member of a relation.

##### Properties

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:ref` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:obverseListingOrder` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#integer`
- `dmlex:role` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`

#### 5.3.2.24 RDF element: `dmlex:RelationType`

Represents one of possible values for the type of relation.

##### Properties

- `dmlex:description` OPTIONAL (at most 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- [`dmlex:sameAs`](#) OPTIONAL
- `dmlex:type` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:scope` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:memberType` OPTIONAL reference to [MemberType](#)

#### 5.3.2.25 RDF element: `dmlex:MemberType`

Represents one of possible values for the role of member, as well as various restrictions on members having this role.

##### Properties

- `dmlex:description` OPTIONAL (at most 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- [`dmlex:sameAs`](#) OPTIONAL
- `dmlex:type` ~~OPTIONAL~~REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:role` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:min` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`

- `dmlex:max` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:action` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`

#### 5.3.2.26 RDF element: *dmlex:PlaceholderMarker*

Marks up a substring inside a headword or inside a headword translation which is not part of the expression itself but stands for things that can take its place.

##### *Properties*

- `dmlex:startIndex` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:endIndex` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#string`

#### 5.3.2.27 RDF element: *dmlex:HeadwordMarker*

Marks up a substring inside an example, inside an example translation or inside a definition which corresponds to the headword (or to a translation of the headword)

##### *Properties*

- `dmlex:startIndex` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:endIndex` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#string`

#### 5.3.2.28 RDF element: *dmlex:CollocateMarker*

Marks up a substring other than the headword inside an example, inside an example translation or inside a definition.

##### *Properties*

- `dmlex:label` OPTIONAL reference to [Label](#)
- `dmlex:lemma` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`

#### 5.3.2.29 RDF element: *dmlex:Etymology*

Represents a chain of historical derivations of a word.

##### *Properties*

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:description` OPTIONAL (at most 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- `dmlex:etymon` OPTIONAL reference to [Etymon](#)

### 5.3.2.30 RDF element: *dmlex:Etymon*

Represents one stage (of possibly several) in the etymological history of the headword.

#### Properties

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:type` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:when` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:note` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:etymonUnit` REQUIRED (at least 1) reference to [EtymonUnit](#)
- `dmlex:translation` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`

### 5.3.2.31 RDF element: *dmlex:EtymonUnit*

Represents a form (typically a word) which is the etymological origin of the headword, or another etymologically related form

#### Properties

- `dmlex:listingOrder` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#nonNegativeInteger`
- `dmlex:language` ~~REQUIRED (exactly 1)~~ `langCode` OPTIONAL of type `http://www.w3.org/2001/XMLSchema#language` (subproperty of `http://www.w3.org/ns/lemon/lime#language`)
- `dmlex:partOfSpeech` OPTIONAL reference to [PartOfSpeech](#)
- `dmlex:text` REQUIRED (exactly 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- `dmlex:translation` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#string`
- `dmlex:reconstructed` OPTIONAL (at most 1) of type `http://www.w3.org/2001/XMLSchema#boolean`

### 5.3.2.32 RDF element: *dmlex:EtymonType*

Represents one of possible values for the type of etymon.

#### Properties

- `dmlex:description` OPTIONAL (at most 1) of type `http://www.w3.org/2000/01/rdf-schema#Literal`
- `dmlex:type` REQUIRED (exactly 1) of type `http://www.w3.org/2001/XMLSchema#string`

### 5.3.2.33 RDF element: *dmlex:EtymonLanguage*

Represents one of several allowed values for the language property of etymonUnit objects.

## Properties

- `dmlex:langCode` REQUIRED (exactly 1) of type <http://www.w3.org/2001/XMLSchema#language> ([subproperty of http://www.w3.org/ns/lemon/lime#language](http://www.w3.org/ns/lemon/lime#language))
- `dmlex:displayName` OPTIONAL (at most 1) of type <http://www.w3.org/2001/XMLSchema#string>

## 5.4 DMLex relational database serialization (Normative)

This section specifies a serialization of the DMLex data model as a ~~feational~~[relational](#) database, using conventional database modelling concepts (tables with columns and rows, primary keys, ~~relationships~~[relationships](#), foreign keys) and data types from the ISO SQL (Structured Query Language) Standard [[SQL](#)].

### 5.4.1 Design principles (Informative)

The relational database serialization of DMLex shown in this document follows these principles:

- The `lexicographicResource` object is implemented as table. (Alternatively, it can left unimplemented if the database is going to contain only one lexicographic resource.)
- Other objects with an arity other than (0..1) and (1..1) are implemented as tables.
- The values of objects, and objects with an arity of (0..1) or (1..1) are implemented as columns in those tables.
- The parent-child relation is implemented as a one-to-many relation between tables.

### 5.4.2 Database diagrams (Informative)

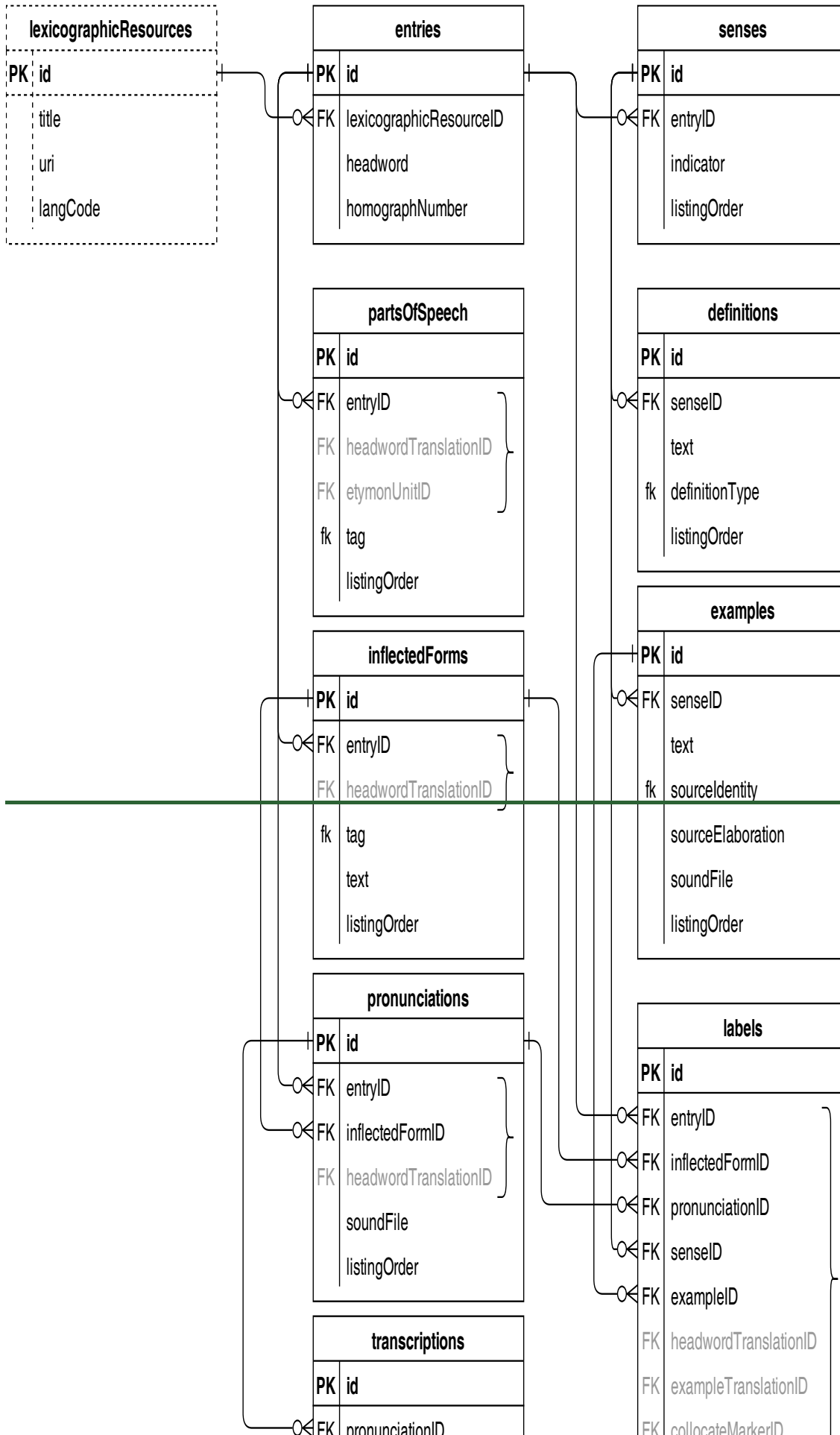
The following diagrams provide a high-level overview of the relational database serialization of DMLex. Each diagram presents a "view" of the database from the perspective of one of the modules or from the perspective of the DMLex Core. The diagrams shown here follow the usual conventions used in relational database diagrams:

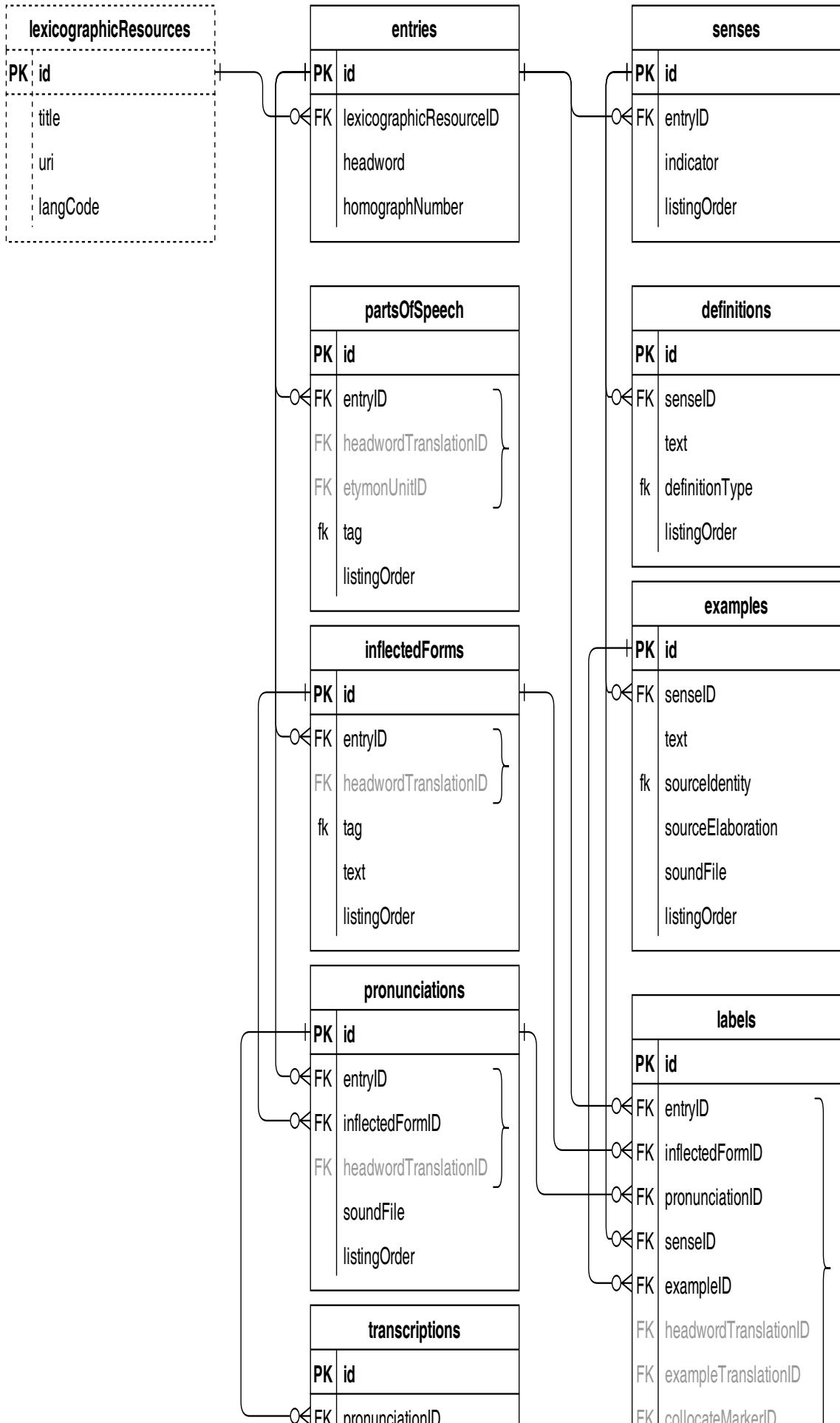
- Each box represents a database table and contains a list of its column headings. The abbreviation PK next to a column name indicates that the column is a primary key. The abbreviation FK next to a column name indicates that the column is a foreign key.
- Lines between the boxes represent relationships between tables, with the conventional "crow's feet" symbols at the ends of the lines to indicate the relation's arity (one-to-many etc.).

In addition to that, the following non-standard conventions are used in the diagrams:

- A curly bracket surrounding a group of foreign keys in a table indicates the constraint that exactly one of the foreign keys in this group must be non-empty while all the other ones must be null.
- Tables and columns shown in grey are tables are defined in another module (or in the core) and are shown only for information.
- The dashed outline on the table `lexicographicResources` indicates that the table is optional. If a database contains only one lexicographic resource, then this table, as well as any foreign-key columns in other tables that reference it, can be left unimplemented.
- Any relationships indicated by a dashed line and/or labelled with the lower-case abbreviation "fk" (as opposed to upper case "FK") are optional: they can be implemented as constraints (if the implementor wishes to have a high degree of referential integrity) or left unimplemented (for example if referential integrity cannot be assumed when importing data from an untrusted tree-structured serialization of DMLex).

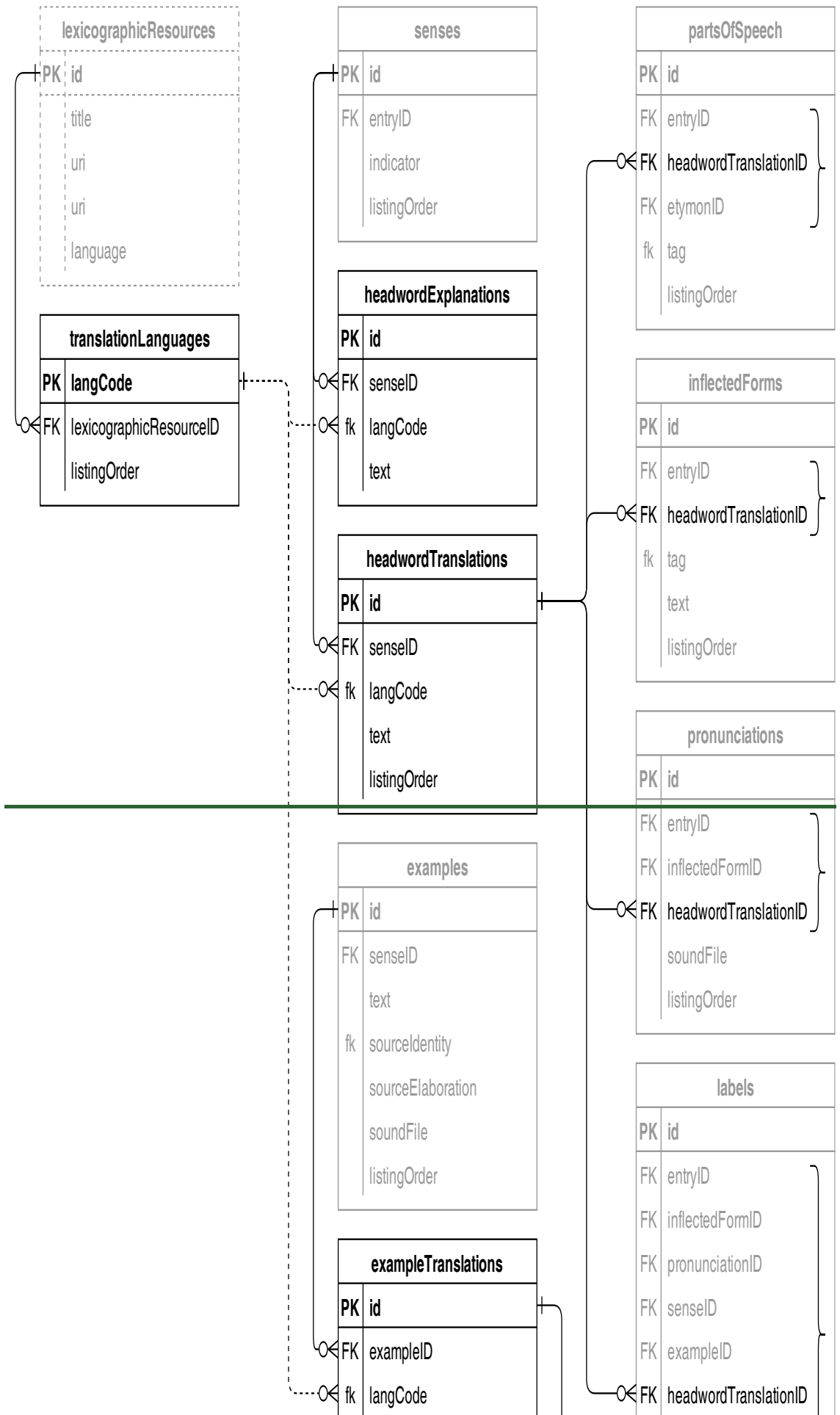
#### 5.4.2.1 Database diagram: DMLex Core

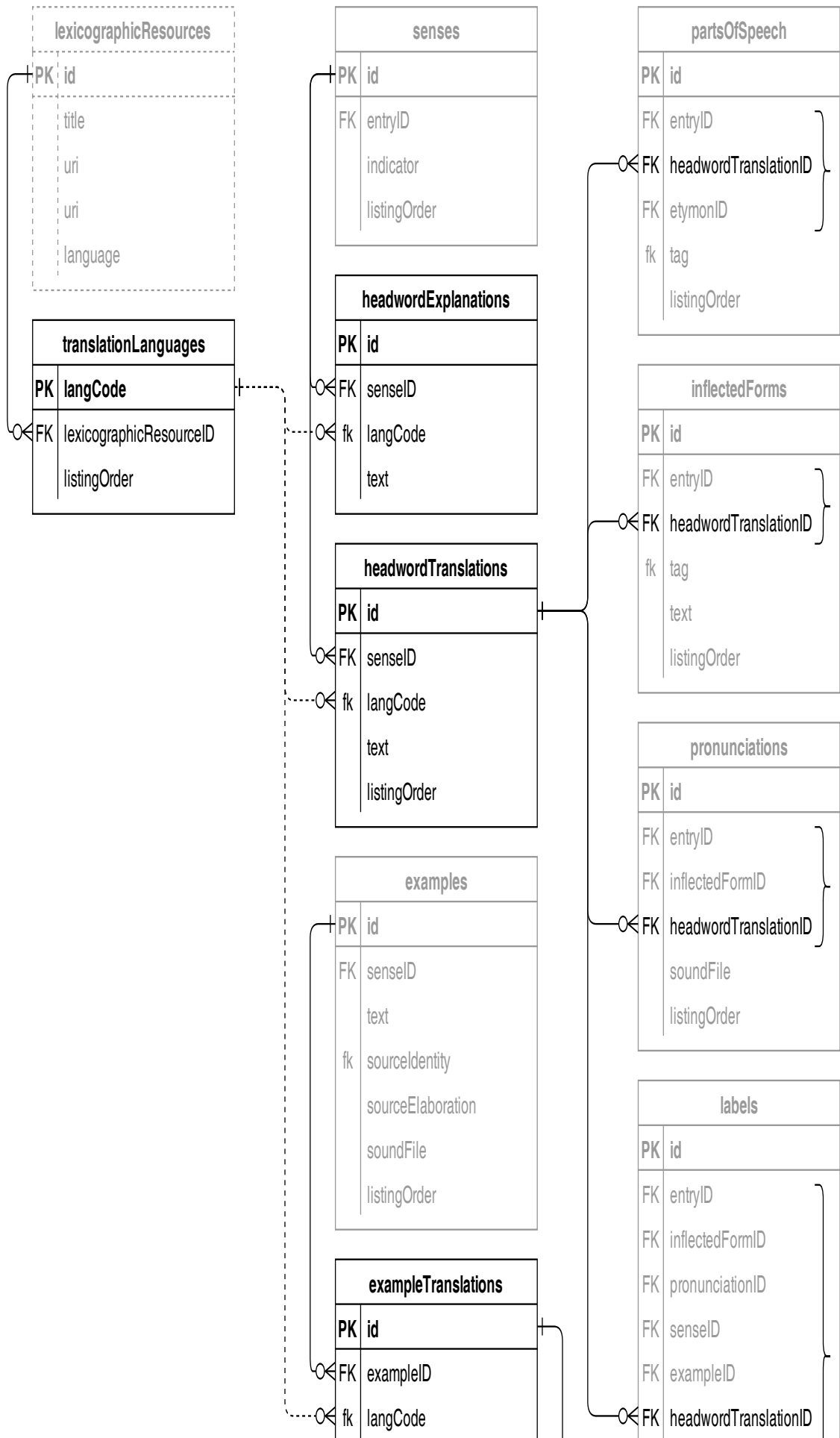






#### **5.4.2.2 Database diagram: DMLex Crosslingual Module**





### **5.4.2.3 Database diagram: DMLex Controlled Values Module**

partsOfSpeech	
<b>PK</b>	id
FK	entryID
FK	headwordTranslationID
FK	etymonID
fk	tag
	listingOrder

inflectedForms	
<b>PK</b>	id
FK	entryID
FK	headwordTranslationID
fk	tag
	text
	listingOrder

labels	
<b>PK</b>	id
FK	entryID
FK	inflectedFormID
FK	pronunciationID
FK	senseID
FK	exampleID
FK	headwordTranslationID
FK	exampleTranslationID

FK	collocateMarkerID
fk	tag
	listingOrder

transcriptions	
<b>PK</b>	id
FK	pronunciationID

lexicographicResources	
<b>PK</b>	id
	title
	uri
	language

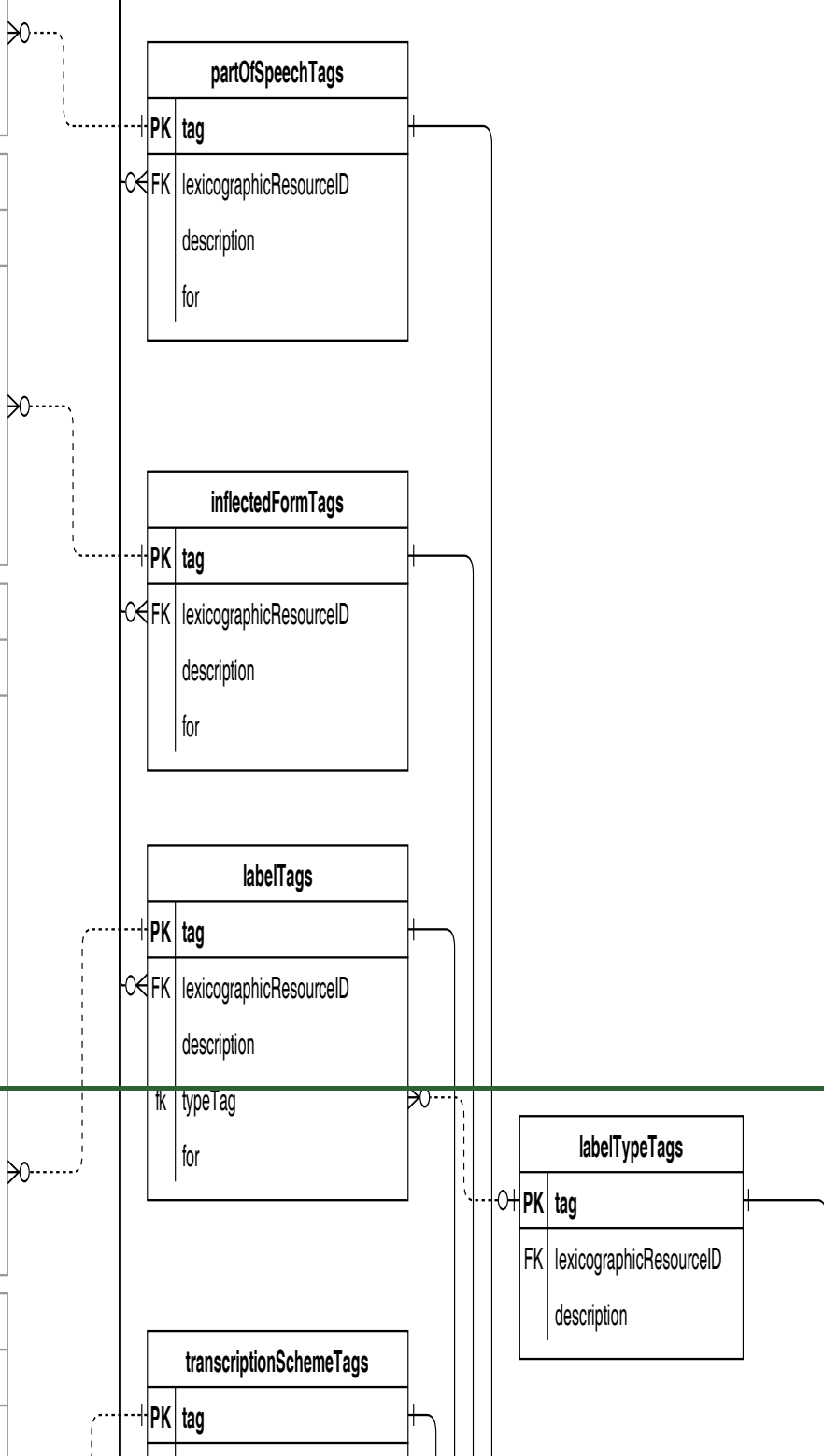
partOfSpeechTags	
<b>PK</b>	tag
FK	lexicographicResourceID
	description
	for

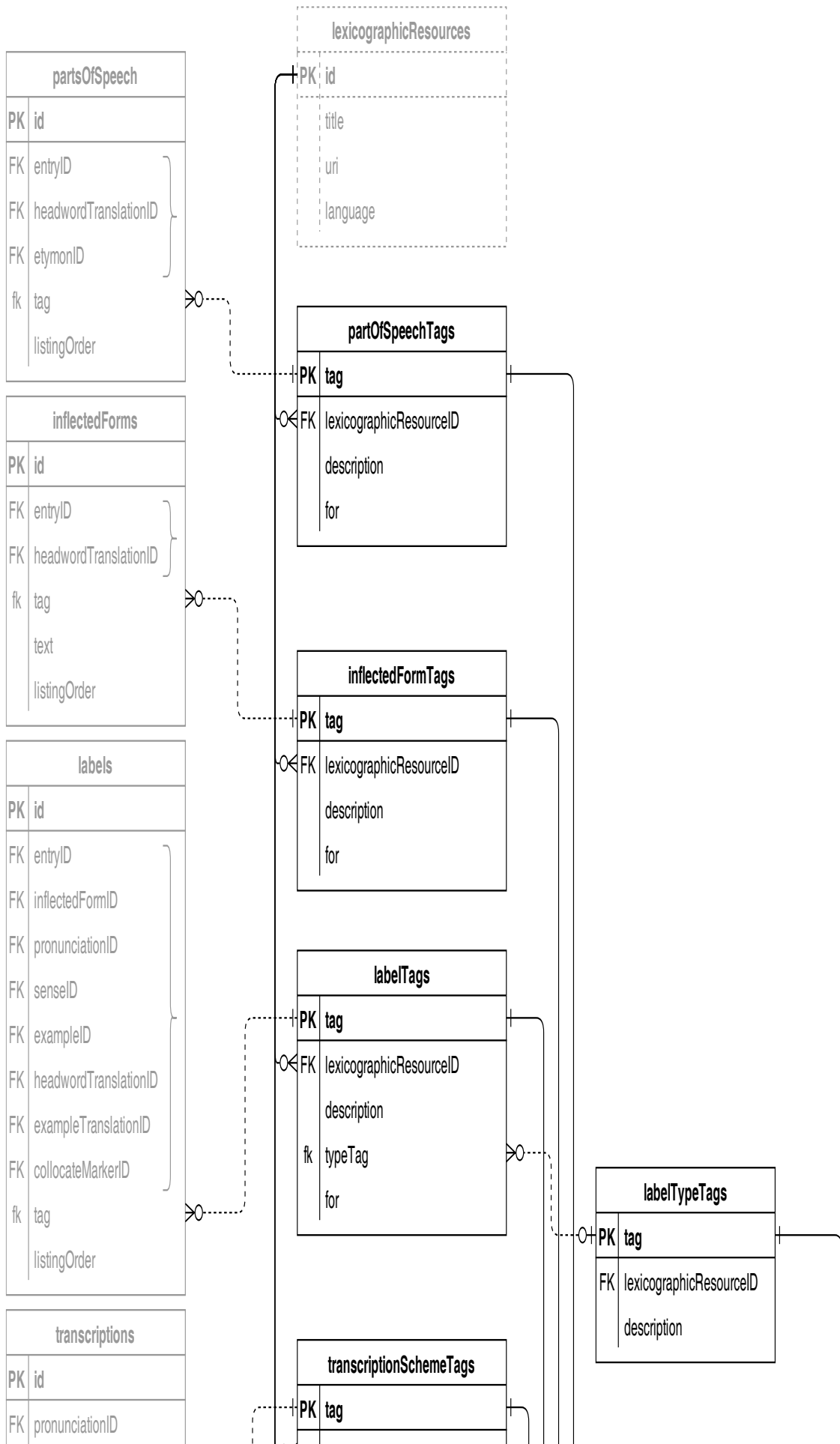
inflectedFormTags	
<b>PK</b>	tag
FK	lexicographicResourceID
	description
	for

labelTags	
<b>PK</b>	tag
FK	lexicographicResourceID
	description
fk	type tag
	for

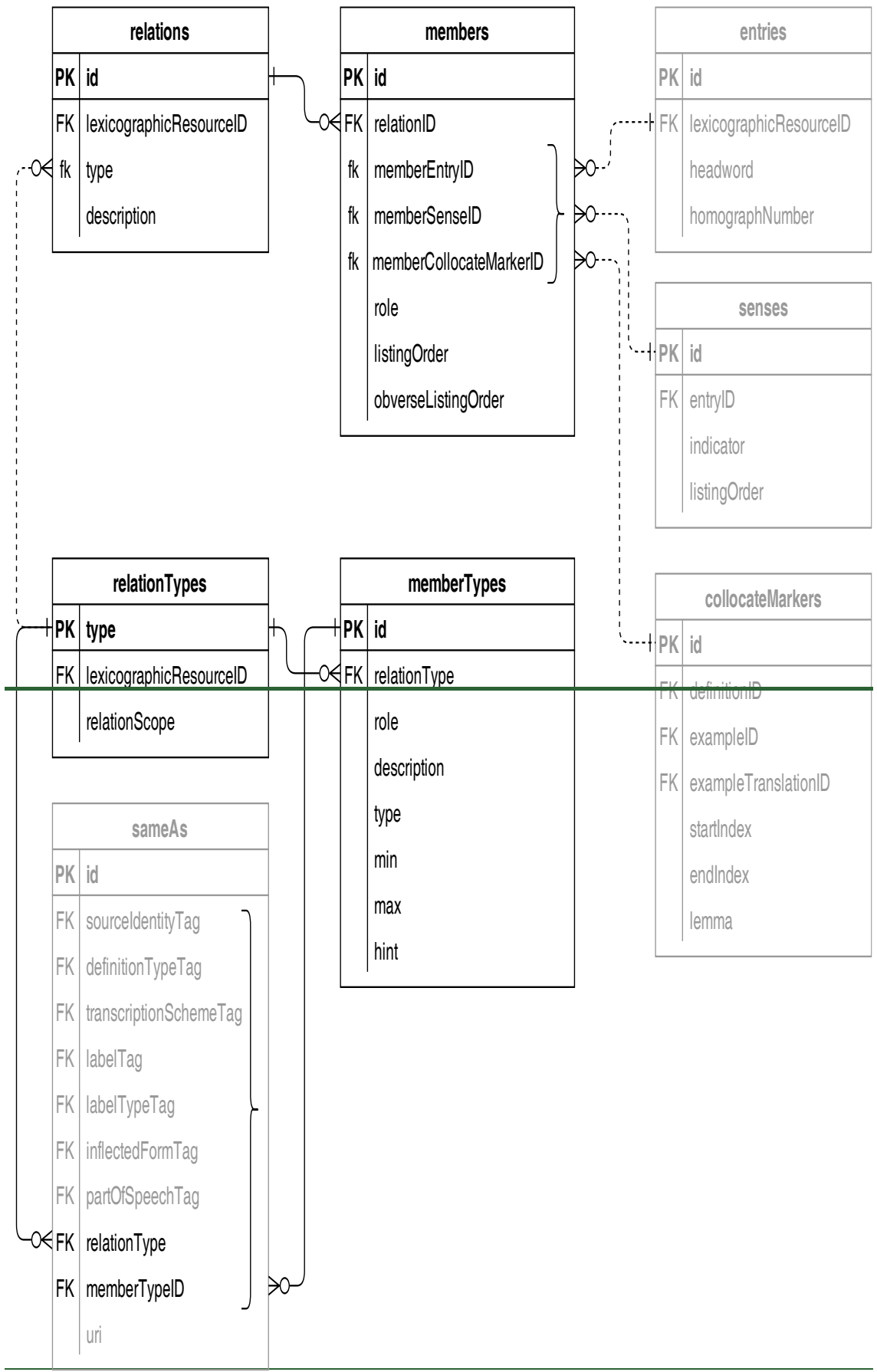
transcriptionSchemeTags	
<b>PK</b>	tag

labelTypeTags	
<b>PK</b>	tag
FK	lexicographicResourceID
	description

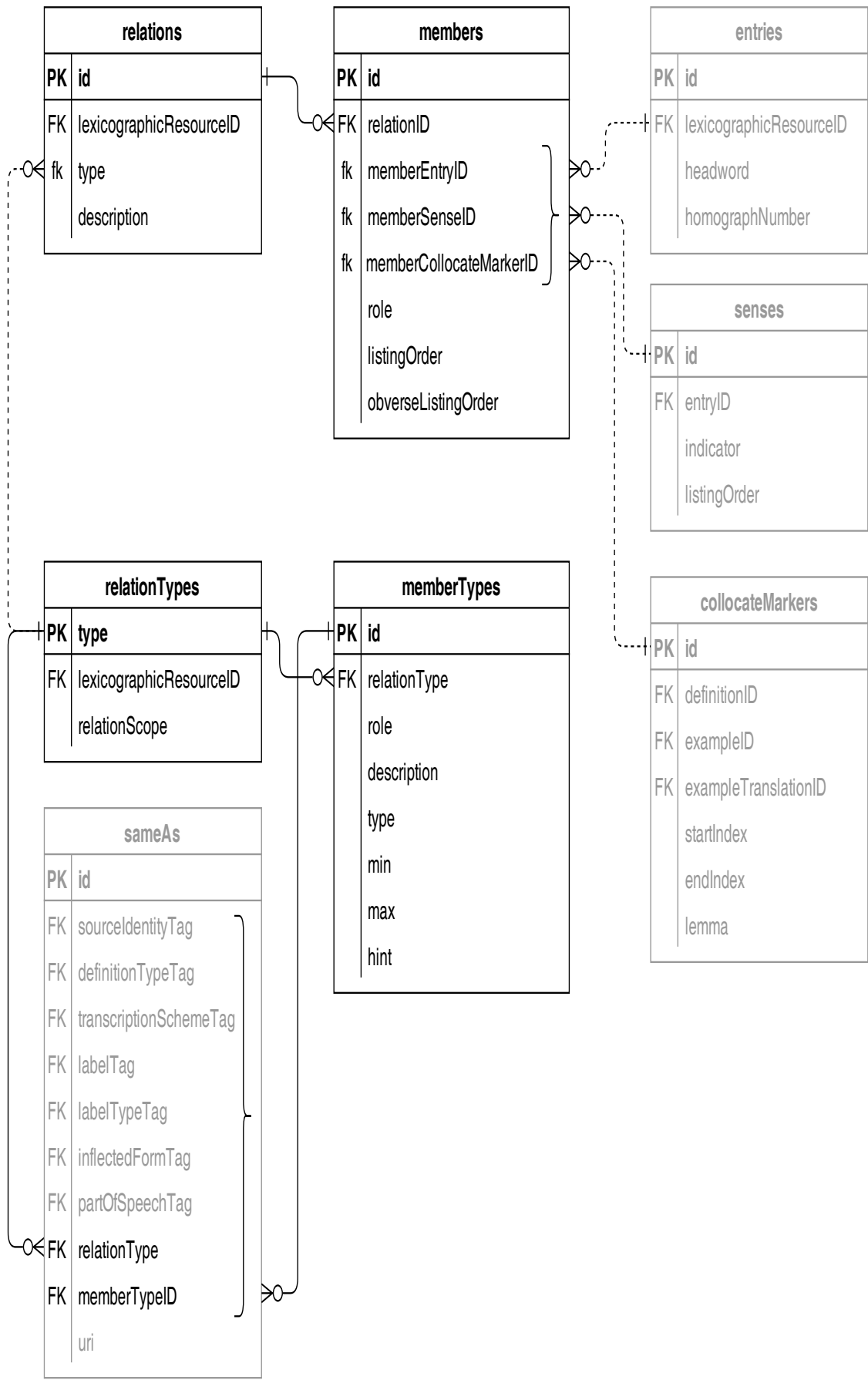




#### **5.4.2.4 Database diagram: DMLex Linking Module**







#### **5.4.2.5 Database diagram: DMLex Annotation Module**

entries	
<b>PK</b>	id
FK	lexicographicResourceID
	headword
	homographNumber

headwordTranslations	
<b>PK</b>	id
FK	senseID
FK	langCode
	text
	listingOrder

definitions	
<b>PK</b>	id
FK	senseID
	text
	definitionType
	listingOrder

examples	
<b>PK</b>	id
FK	senseID
	text
	sourceIdentity
	sourceElaboration
	soundFile
	listingOrder

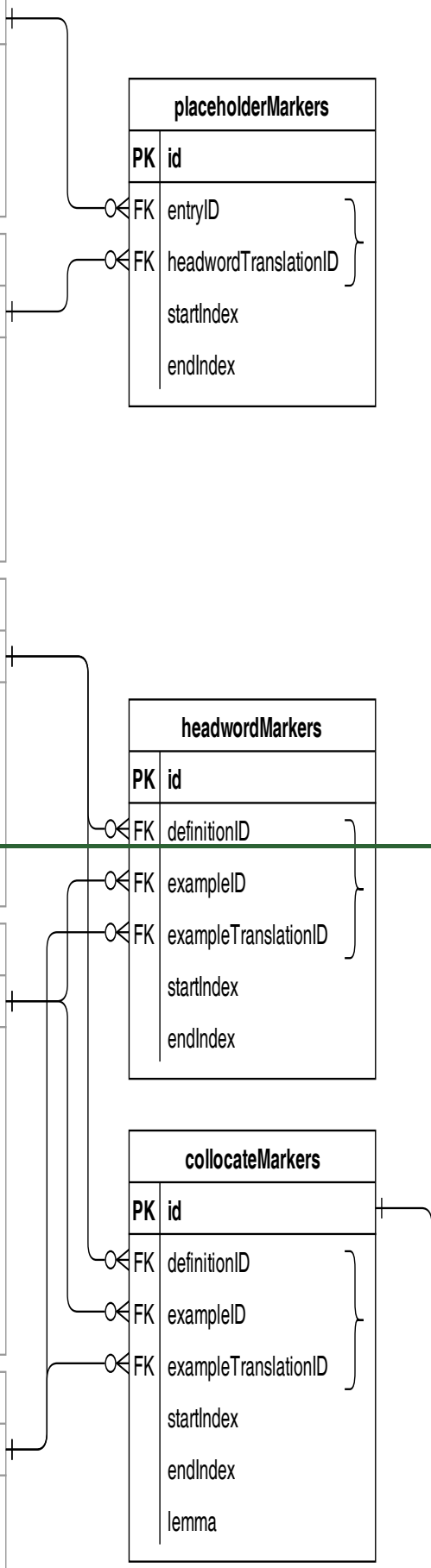
exampleTranslations	
<b>PK</b>	id
FK	exampleID
FK	langCode

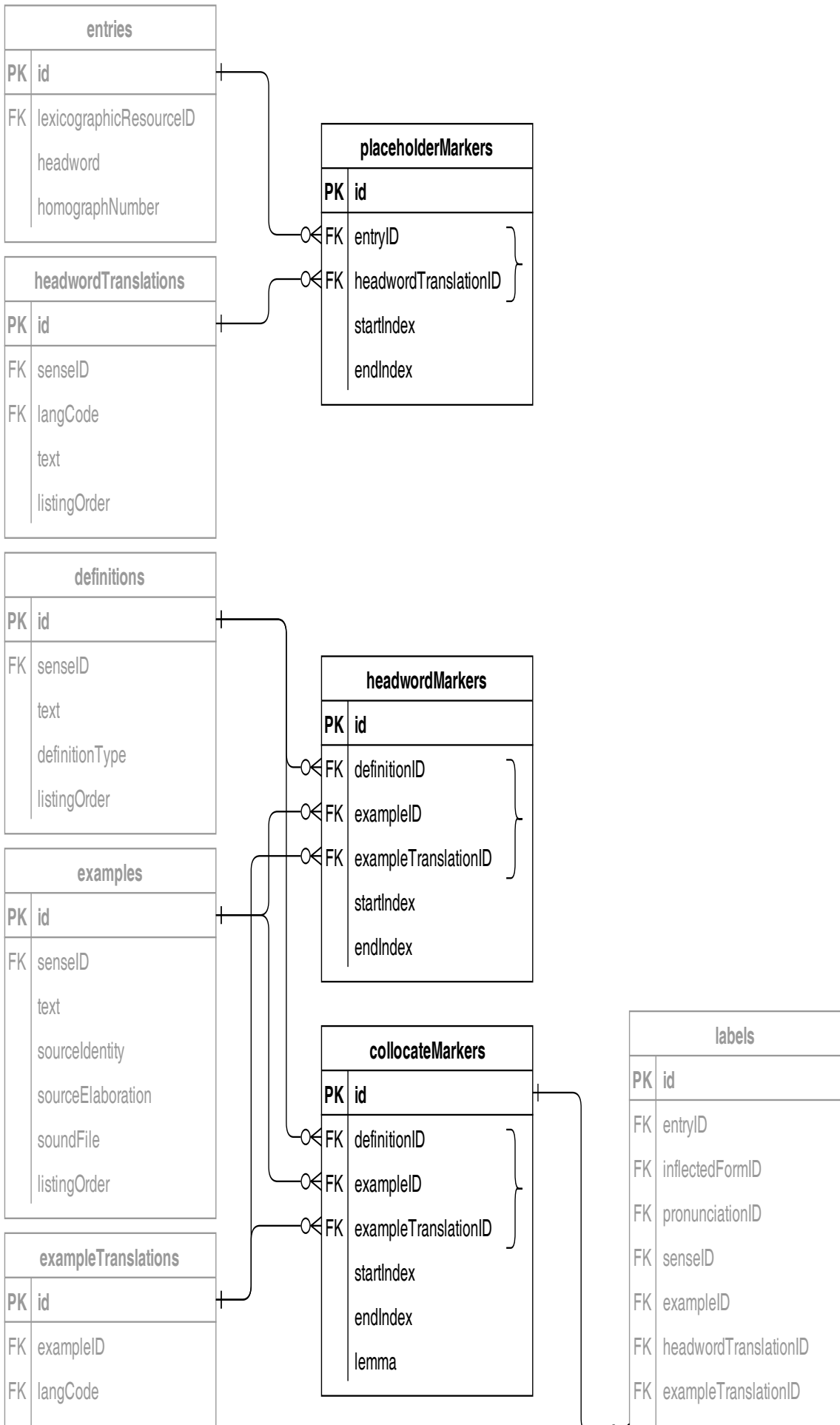
placeholderMarkers	
<b>PK</b>	id
FK	entryID
FK	headwordTranslationID
	startIndex
	endIndex

headwordMarkers	
<b>PK</b>	id
FK	definitionID
FK	exampleID
FK	exampleTranslationID
	startIndex
	endIndex

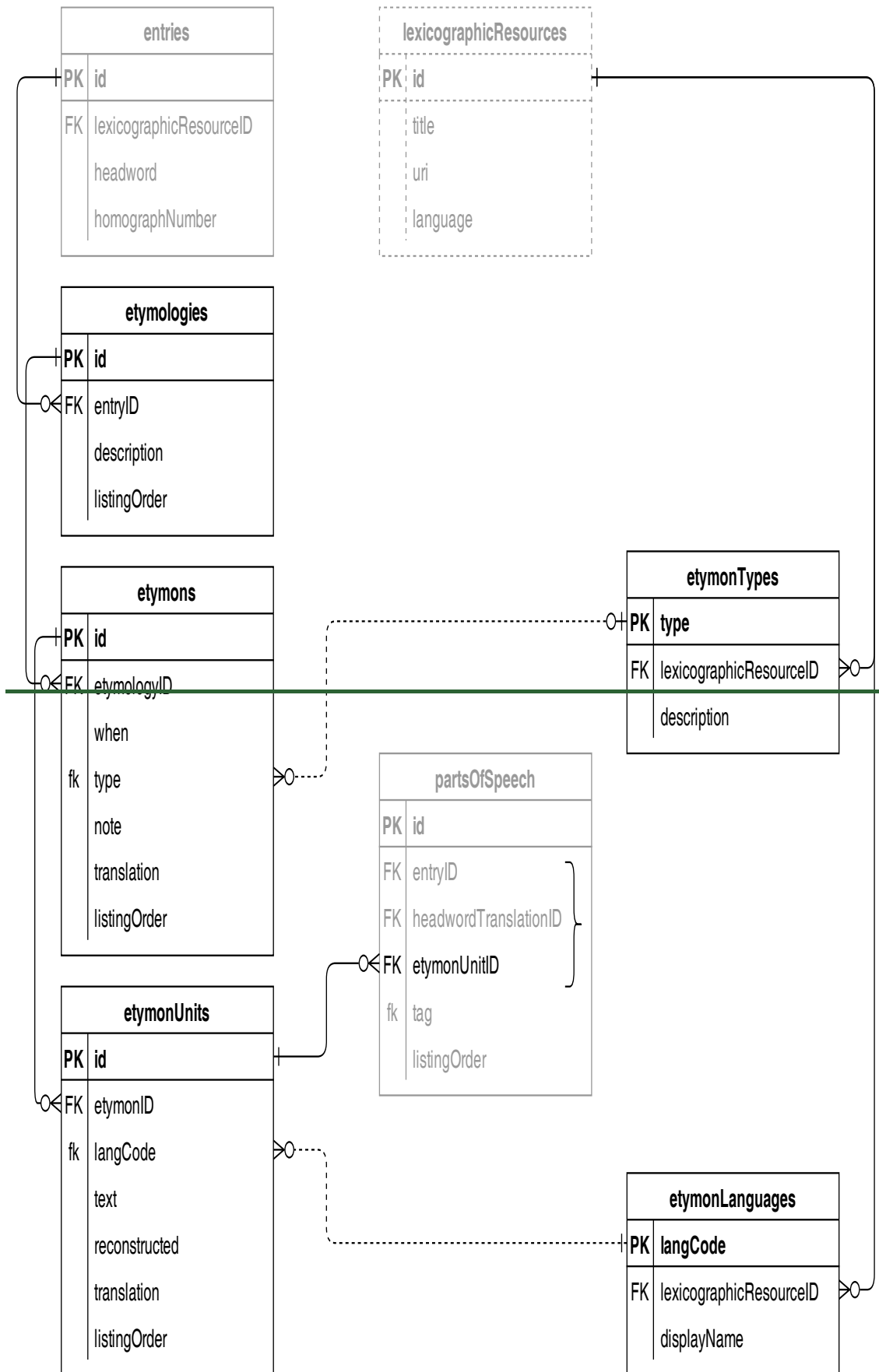
collocateMarkers	
<b>PK</b>	id
FK	definitionID
FK	exampleID
FK	exampleTranslationID
	startIndex
	endIndex
	lemma

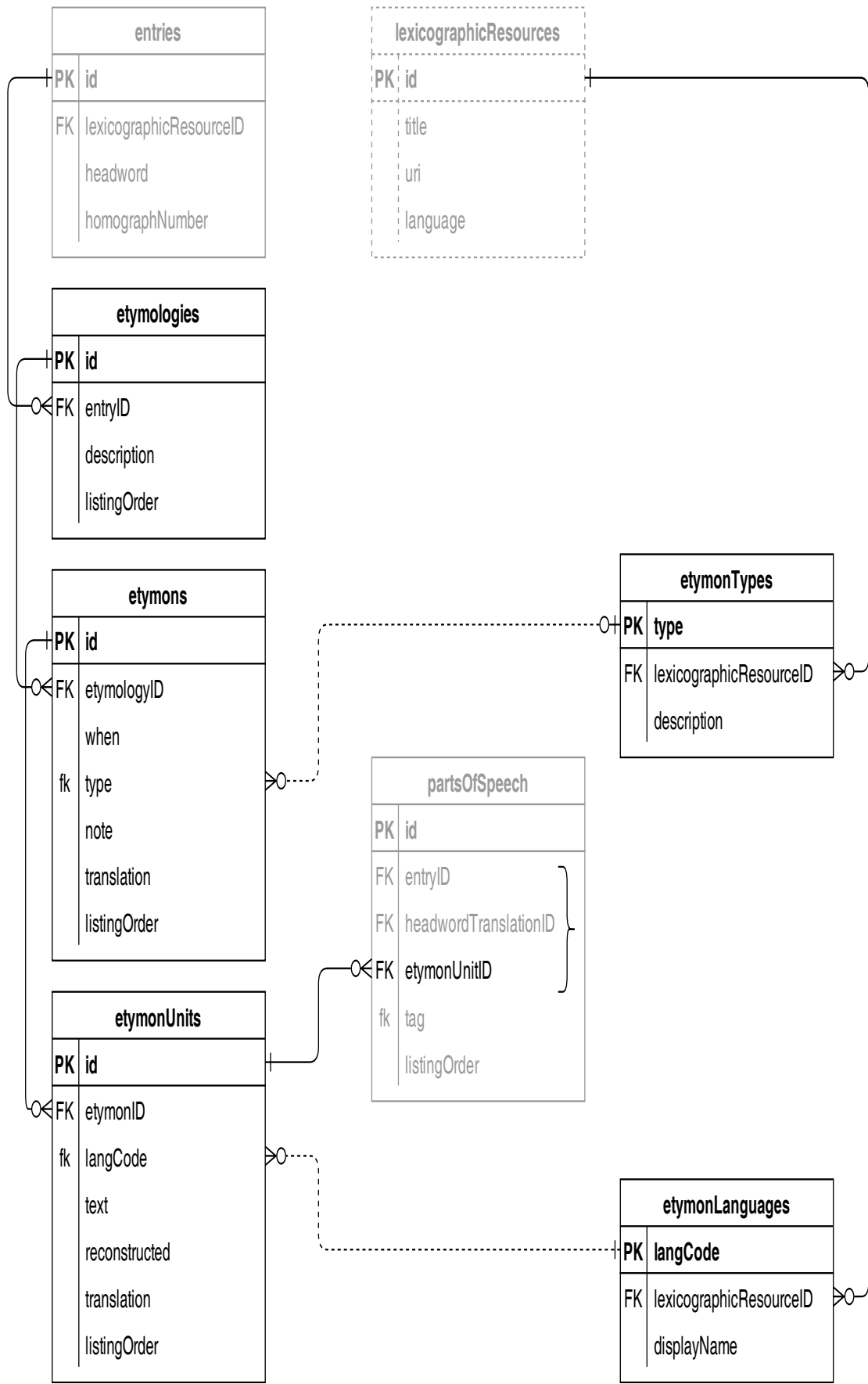
labels	
<b>PK</b>	id
FK	entryID
FK	inflectedFormID
FK	pronunciationID
FK	senseID
FK	exampleID
FK	headwordTranslationID
FK	exampleTranslationID





#### **5.4.2.6 Database diagram: DMLex Etymology Module**





## 5.4.3 DMLex relational database serialization tables (Normative)

### 5.4.3.1 Database table: *lexicographicResources*

Implements the [lexicographicResource](#) object type.

#### Columns

- id: int, primary key
- title: nvarchar
- uri: nvarchar
- langCode: varchar

#### Comment

- If the database is designed to contain only one lexicographic resource then this table, ~~including~~ **including** all foreign keys in other tables that reference it, is not REQUIRED to be implemented.

### 5.4.3.2 Database table: *entries*

Implements the [entry](#) object type.

#### Columns

- id: int, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)
- headword: nvarchar
- homographNumber: int

### 5.4.3.3 Database table: *partsOfSpeech*

Implements the [partOfSpeech](#) object type.

#### Columns

- id: int, primary key
- entryID: int, foreign key, references column id of table [entries](#)
- tag: nvarchar, optionally a foreign key referencing column tag of table [partOfSpeechTags](#)
- listingOrder: int

#### Columns if implementing the Crosslingual Module

- headwordTranslationID: int, foreign key, references column id of table [headwordTranslations](#)

#### Columns if implementing the Etymology Module

- etymonUnitID: int, foreign key, references column id of table [etymonUnits](#)



### *Constraint*

- Exactly one of the foreign keys `entryID`, `headwordTranslationID` and `etymonID` is REQUIRED to be non-empty.

### **5.4.3.4 Database table: *inflectedForms***

Implements the [inflectedForm](#) object type.

#### *Columns*

- `id`: int, primary key
- `entryID`: int, foreign key, references column `id` of table [entries](#)
- `tag`: nvarchar, optionally a foreign key referencing column `tag` of table [inflectedFormTags](#)
- `text`: nvarchar
- `listingOrder`: int

#### *Columns if implementing the Crosslingual Module*

- `headwordTranslationID`: int, foreign key, references column `id` of table [headwordTranslations](#)

### *Constraint*

- Exactly one of the foreign keys `entryID` and `headwordTranslationID` is REQUIRED to be non-empty.

### **5.4.3.5 Database table: *senses***

Implements the [sense](#) object type.

#### *Columns*

- `id`: int, primary key
- `entryID`: int, foreign key, references column `id` of table [entries](#)
- `indicator`: nvarchar
- `listingOrder`: int

### **5.4.3.6 Database table: *definitions***

Implements the [definition](#) object type.

#### *Columns*

- `id`: int, primary key
- `senseID`: int, foreign key, references column `id` of table [senses](#)
- `text`: nvarchar
- `definitionType`: nvarchar, optionally a foreign key referencing column `tag` of table [definitionTypeTags](#)
- `listingOrder`: int

#### 5.4.3.7 Database table: *labels*

Implements the [label](#) object type.

##### Columns

- id: int, primary key
- entryID: int, foreign key, references column id of table [entries](#)
- inflectedFormID: int, foreign key, references column id of table [inflectedForms](#)
- pronunciationID: int, foreign key, references column id of table [pronunciations](#)
- senseID: int, foreign key, references column id of table [senses](#)
- exampleID: int, foreign key, references column id of table [examples](#)
- tag: nvarchar, optionally a foreign key referencing column tag of table [labelTags](#)
- listingOrder: int

##### Columns if implementing the Crosslingual Module

- headwordTranslationID: int, foreign key, references column id of table [headwordTranslations](#)
- exampleTranslationID: int, foreign key, references column id of table [exampleTranslations](#)

##### Columns if implementing the Annotation Module

- collocateMarkerID: int, foreign key, references column id of table [collocateMarkers](#)

##### Constraint

- Exactly one of the foreign keys entryID, inflectedFormID, pronunciationID, senseID, exampleID, headwordTranslationID, exampleTranslationID and collocateMarkerID is REQUIRED to be non-empty.

#### 5.4.3.8 Database table: *pronunciations*

Implements the [pronunciation](#) object type.

##### Columns

- id: int, primary key
- entryID: int, foreign key, references column id of table [entries](#)
- inflectedFormID: int, foreign key, references column id of table [inflectedForms](#)
- soundFile: nvarchar
- listingOrder: int

##### Columns if implementing the Crosslingual Module

- headwordTranslationID: int, foreign key, references column id of table [headwordTranslations](#)

##### Constraint

- Exactly one of the foreign keys `entryID`, `inflectedFormID` and `headwordTranslationID` is REQUIRED to be non-empty.

#### **5.4.3.9 Database table: *transcriptions***

Implements the [transcription](#) object type.

##### *Columns*

- `id`: int, primary key
- `pronunciationID`: int, foreign key, references column `id` of table [pronunciations](#)
- `text`: nvarchar
- `scheme`: nvarchar, optionally a foreign key referencing column `tag` of table [transcriptionSchemeTags](#)
- `listingOrder`: int

#### **5.4.3.10 Database table: *examples***

Implements the [example](#) object type.

##### *Columns*

- `id`: int, primary key
- `senseID`: int, foreign key, references column `id` of table [senses](#)
- `text`: nvarchar
- `sourceIdentity`: nvarchar, optionally a foreign key referencing column `tag` of table [sourceIdentityTags](#)
- `sourceElaboration`: nvarchar
- `soundFile`: nvarchar
- `listingOrder`: int

#### **5.4.3.11 Database table: *translationLanguages***

Implements the [translationLanguage](#) object type from the Crosslingual Module.

##### *Columns*

- `langCode`: varchar, primary key
- `lexicographicResourceID`: int, foreign key, references column `id` of table [lexicographicResources](#)
- `listingOrder`: int

#### **5.4.3.12 Database table: *headwordTranslations***

Implements the [headwordTranslation](#) object type from the Crosslingual Module.

##### *Columns*

- `id`: int, primary key
- `senseID`: int, foreign key, references column `id` of table [senses](#)

- langCode: varchar, optionally a foreign key referencing column langCode of table [translationLanguages](#)
- text: nvarchar
- listingOrder: int

#### **5.4.3.13 Database table: *headwordExplanations***

Implements the [headwordExplanation](#) object type from the Crosslingual Module.

##### **Columns**

- id: int, primary key
- senseID: int, foreign key, references column id of table [senses](#)
- langCode: varchar, optionally a foreign key referencing column langCode of table [translationLanguages](#)
- text: nvarchar

#### **5.4.3.14 Database table: *exampleTranslations***

Implements the [exampleTranslation](#) object type from the Crosslingual Module.

##### **Columns**

- id: int, primary key
- exampleID: int, foreign key, references column id of table [examples](#)
- langCode: varchar, foreign key, references column langCode of table [translationLanguages](#)
- text: nvarchar
- soundFile: nvarchar
- listingOrder: int

#### **5.4.3.15 Database table: *definitionTypeTags***

Implements the [definitionTypeTag](#) object type from the Controlled Values Module.

##### **Columns**

- tag: nvarchar, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)
- description: nvarchar

#### **5.4.3.16 Database table: *inflectedFormTags***

Implements the [inflectedFormTags](#) object type from the Controlled Values Module.

##### **Columns**

- tag: nvarchar, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)

- description: nvarchar
- for: nvarchar

#### **5.4.3.17 Database table: *labelTags***

Implements the [labelTag](#) object type from the Controlled Values Module.

##### *Columns*

- tag: nvarchar, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)
- description: nvarchar
- typeTag: nvarchar, optionally a foreign key referencing column tag of table [labelTypeTags](#)
- for: nvarchar

#### **5.4.3.18 Database table: *labelTypeTags***

Implements the [labelTypeTag](#) object type from the Controlled Values Module.

##### *Columns*

- tag: nvarchar, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)
- description: nvarchar

#### **5.4.3.19 Database table: *partOfSpeechTags***

Implements the [partOfSpeechTag](#) object type from the Controlled Values Module.

##### *Columns*

- tag: nvarchar, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)
- description: nvarchar
- for: nvarchar

#### **5.4.3.20 Database table: *sourceIdentityTags***

Implements the [sourceIdentityTag](#) object type from the Controlled Values Module.

##### *Columns*

- tag: nvarchar, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)
- description: nvarchar

#### 5.4.3.21 Database table: *transcriptionSchemeTags*

Implements the [transcriptionSchemeTag](#) object type from the Controlled Values Module.

##### Columns

- tag: nvarchar, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)
- description: nvarchar
- for: nvarchar

#### 5.4.3.22 Database table: *sameAs*

Implements the [sameAs](#) object type from the Controlled Values Module.

##### Columns

- id: int, primary key
- sourceIdentityTag: nvarchar, foreign key, references column tag of table [sourceIdentityTags](#)
- definitionTypeTag: nvarchar, foreign key, references column tag of table [definitionTypeTags](#)
- transcriptionSchemeTag: nvarchar, foreign key, references column tag of table [transcriptionSchemeTags](#)
- labelTag: nvarchar, foreign key, references column tag of table [labelTags](#)
- labelTypeTag: nvarchar, foreign key, references column tag of table [labelTypeTags](#)
- inflectedFormTag: nvarchar, foreign key, references column tag of table [inflectedFormTags](#)
- partOfSpeechTag: nvarchar, foreign key, references column tag of table [partOfSpeechTags](#)
- uri: nvarchar

##### Columns if implementing the Linking Module

- relationType: nvarchar, foreign key, references column type of table [relationTypes](#)
- memberTypeID: int, foreign key, references column id of table [memberTypes](#)

##### Constraint

- Exactly one of the foreign keys sourceIdentityTag, definitionTypeTag, transcriptionSchemeTag, labelTag, labelTypeTag, inflectedFormTag, partOfSpeechTag, relationType and memberTypeID is REQUIRED to be non-empty.

#### 5.4.3.23 Database table: *relations*

Implements the [relation](#) object type from the Linking Module.

##### Columns

- id: int, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)
- type: nvarchar, optionally a foreign key referencing column type of table [relationTypes](#)
- description: nvarchar

#### 5.4.3.24 Database table: *members*

Implements the [member](#) object type from the Linking Module.

##### Columns

- id: int, primary key
- relationID: int, foreign key, references column id of table [relations](#)
- memberEntryID: int, foreign key, references column id of table [entries](#)
- memberSenseID: int, foreign key, references column id of table [senses](#)
- role: nvarchar
- listingOrder: int
- obverseListingOrder: int

##### Columns if implementing the Annotation Module

- memberCollocateMarkerID: int, foreign key, references column id of table [entries](#)

##### Constraint

- Exactly one of the foreign keys memberEntryID, memberSenseID and memberCollocateMakerID is REQUIRED to be non-empty.

#### 5.4.3.25 Database table: *relationTypes*

Implements the [relationType](#) object type from the Linking Module.

##### Columns

- type: nvarchar, primary key
- lexicographicResourceID: int, foreign key, references column id of table [lexicographicResources](#)
- relationScope: nvarchar

#### 5.4.3.26 Database table: *memberTypes*

Implements the [memberType](#) object type from the Linking Module.

##### Columns

- id: int, primary key
- relationType: nvarchar, foreign key, references column type of table [relationType](#)
- role: nvarchar

- description: nvarchar
- type: nvarchar
- min: int
- max: int
- hint: varchar

#### 5.4.3.27 Database table: *placeholderMarkers*

Implements the [placeholderMarker](#) object type from the Annotationm Module.

##### Columns

- id: int, primary key
- entryID: int, foreign key, references column id of table [entries](#)
- startIndex: int
- endIndex: int

##### Columns if implementing the Crosslingual Module

- headwordTranslationID: int, foreign key, references column id of table [headwordTranslations](#)

##### Constraint

- Exactly one of the foreign keys entryID and headwordTranslationID, is REQUIRED to be non-empty.

#### 5.4.3.28 Database table: *headwordMarkers*

Implements the [headwordMarker](#) object type from the Annotationm Module.

##### Columns

- id: int, primary key
- definitionID: int, foreign key, references column id of table [definitions](#)
- exampleID: int, foreign key, references column id of table [examples](#)
- startIndex: int
- endIndex: int

##### Columns if implementing the Crosslingual Module

- exampleTranslationID: int, foreign key, references column id of table [exampleTranslations](#)

##### Constraint

- Exactly one of the foreign keys definitionID, exampleID and exampleTranslationID, is REQUIRED to be non-empty.

#### 5.4.3.29 Database table: *collocateMarkers*



Implements the [collocateMarker](#) object type from the Annotationm Module.

#### Columns

- id: int, primary key
- definitionID: int, foreign key, references column id of table [definitions](#)
- exampleID: int, foreign key, references column id of table [examples](#)
- startIndex: int
- endIndex: int
- lemma: nvarchar

#### Columns if implementing the Crosslingual Module

- exampleTranslationID: int, foreign key, references column id of table [exampleTranslations](#)

#### Constraint

- Exactly one of the foreign keys definitionID, exampleID and exampleTranslationID, is REQUIRED to be non-empty.

#### 5.4.3.30 Database table: *etymologies*

Implements the [etymology](#) object type from the Etymology Module.

#### Columns

- id: int, primary key
- entryID: int, foreign key, references column id of table [entries](#)
- description: nvarchar
- listingOrder: int

#### 5.4.3.31 Database table: *etymons*

Implements the [etymon](#) object type from the Etymology Module.

#### Columns

- id: int, primary key
- etymologyID: int, foreign key, references column id of table [etymologies](#)
- when: nvarchar
- type: nvarchar, optionally a foreign key referencing column type of table [etymonTypes](#)
- note: nvarchar
- translation: nvarchar
- listingOrder: int

#### 5.4.3.32 Database table: *etymonUnits*

Implements the [etymonUnit](#) object type from the Etymology Module.

## Columns

- `id`: int, primary key
- `etymonID`: int, foreign key, references column `id` of table [etymons](#)
- `langCode`: nvarchar, optionally a foreign key referencing column `langCode` of table [etymonLanguages](#)
- `text`: nvarchar
- `reconstructed`: bit
- `translation`: nvarchar
- `listingOrder`: int

### 5.4.3.33 Database table: *etymonTypes*

Implements the [etymonType](#) object type from the Etymology Module.

## Columns

- `type`: nvarchar, primary key
- `lexicographicResourceID`: int, foreign key, references column `id` of table [lexicographicResources](#)
- `description`: nvarchar

### 5.4.3.34 Database table: *etymonLanguages*

Implements the [etymonLanguage](#) object type from the Etymology Module.

## Columns

- `langCode`: nvarchar, primary key
- `lexicographicResourceID`: int, foreign key, references column `id` of table [lexicographicResources](#)
- `displayName`: nvarchar

## Appendix A Informative material on serializations (Informative)

### A.1 Examples (Informative)

This section gives examples which show how to use DMLex to model lexicographic resources. Each example is shown in pseudocode first to demonstrate the object model at an abstract level. After that, each example is shown in XML and JSON. The XML and JSON encoding shown here follows DMLex's own implementation guidance for XML and JSON.

In the pseudocode, relation objects use alphanumeric IDs to indicate which entries or senses they point to, for example `microscope-1` and `der-ozean-1`. These IDs are used here instead of [fragment identification strings](#) for easier readability.

#### A.1.1 A basic entry

This is a basic, beginner-level example of how to use DMLex to represent a simple monolingual lexicographic resource consisting of one entry with two senses. It demonstrates some of the basic features

of DMLex Core: how to subdivide a entry into senses, how attach various data such as definition, part-of-speech labels to entries and senses, and how to add labels to various objects such as senses and examples.

#### Example A.1. Pseudocode

```
- lexicographicResource (uri: http://example.com, langCode: en)
  - entry (id: abandon-verb, headword: abandon)
    - partOfSpeech (tag: verb)
    - sense (id: abandon-verb-1)
      - definition (text: to suddenly leave a place or a person)
      - example (text: I'm sorry I abandoned you like that.)
      - example (text: Abandon ship!)
        - label (tag: idiom)
    - sense (id: abandon-verb-2)
      - label (tag: mostly-passive)
      - definition (text: to stop supporting an idea)
      - example (text: That theory has been abandoned.)
```

#### Example A.2. XML

```
<lexicographicResource title="Example Dictionary"
  uri="http://example.com" langCode="en">
  <title>Example Dictionary</title>
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <entry id="abandon-verb">
    <headword>abandon</headword>
    <partOfSpeech tag="verb"/>
    <sense id="abandon-verb-1">
      <definition>
        <text>to suddenly leave a place or a person</text>
      </definition>
      <example>
        <text>I'm sorry I abandoned you like that.</text>
      </example>
      <example>
        <text>Abandon ship!</text>
        <label tag="idiom"/>
      </example>
    </sense>
    <sense id="abandon-verb-2">
      <label tag="mostly-passive"/>
      <definition>
        <text>to stop supporting an idea</text>
      </definition>
      <example>
        <text>That theory has been abandoned.</text>
      </example>
    </sense>
  </entry>
</lexicographicResource>
```

#### Example A.3. JSON

```

{
  "uri": "http://example.com",
  "langCode": "en",
  "title": "Example Dictionary",
  "entries": [{
    "id": "abandon-verb",
    "headword": "abandon",
    "partsOfSpeech": ["verb"],
    "senses": [{
      "id": "abandon-verb-1",
      "definitions": [{
        "text": "to suddenly leave a place or a person"
      }],
      "examples": [{
        "text": "I'm sorry I abandoned you like that."
      }, {
        "text": "Abandon ship!",
        "labels": ["idiom"]
      }]
    }, {
      "id": "abandon-verb-2",
      "labels": ["mostly-passive"],
      "definitions": [{"text": "to stop supporting an idea"}],
      "examples": [{
        "text": "That theory has been abandoned."
      }]
    }
  ]
}

```

**Example A.4. RDF**

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:title "Example Dictionary";
  dmlex:uri "http://example.com";
  dmlex:langCode "en";
  dmlex:entry ex:abandon-verb.
ex:abandon-verb a dmlex:Entry;
  dmlex:sense ex:abandon-verb-1,
    ex:abandon-verb-2;
  dmlex:partOfSpeech [
    dmlex:tag "verb";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:headword "abandon".
ex:abandon-verb-1 a dmlex:Sense;
  dmlex:definition [ a dmlex:Definition;
    dmlex:text "to suddenly leave a place or a person";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:example [ a dmlex:Example;
    dmlex:text "I'm sorry I abandoned you like that.";
    dmlex:listingOrder "1"^^xsd:unsignedInt ],
  [ a dmlex:Example;
    dmlex:text "Abandon ship!";
    dmlex:label [

```

```

        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:tag "idiom" ];
    dmlex:listingOrder "2"^^xsd:unsignedInt ];
    dmlex:listingOrder "1"^^xsd:unsignedInt.
ex:abandon-verb-2 a dmlex:Sense;
    dmlex:listingOrder "2"^^xsd:unsignedInt;
    dmlex:label [
        dmlex:tag "mostly-passive";
        dmlex:listingOrder "1"^^xsd:unsignedInt ];
    dmlex:example [ a dmlex:Example;
        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "That theory has been abandoned." ];
    dmlex:definition [ a dmlex:Definition;
        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "to stop supporting an idea" ].

```

## A.1.2 How to use `inflectedForm`

This is an entry from a hypothetical Irish dictionary for the headword "folúsghlantóir" ("vacuum cleaner") which gives its two inflected forms, the singular genitive and the plural.

### Example A.5. Pseudocode

```

- entry (id: folúsghlantóir-n, headword: folúsghlantóir)
  - partOfSpeech (tag: n-masc)
  - inflectedForm (tag: sg-gen, text: folúsghlantóra)
  - inflectedForm (tag: pl, text: folúsghlantóirí)
  - sense...

```

### Example A.6. XML

```

<entry id="folúsghlantóir-n">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <headword>folúsghlantóir</headword>
  <partOfSpeech tag="n-masc"/>
  <inflectedForm tag="sg-gen">
    <text>folúsghlantóra</text>
  </inflectedForm>
  <inflectedForm tag="pl">
    <text>folúsghlantóirí</text>
  </inflectedForm>
</entry>

```

### Example A.7. JSON

```

{
  "id": "folúsghlantóir-n",
  "headword": "folúsghlantóir",
  "partsOfSpeech": ["n-masc"],
  "inflectedForms": [{
    "tag": "sg-gen",
    "text": "folúsghlantóra"
  }

```

```

    }, {
      "tag": "pl",
      "text": "folúsghlantóirí"
    }],
    "senses": []
  }
}

```

**Example A.8. RDF**

```

ex:folúsghlantóir-n a dmlex:Entry;
  dmlex:headword "folúsghlantóir";
  dmlex:partOfSpeech [
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:tag "n-masc" ].

```

### A.1.3 Pronunciation given as transcription

**Example A.9. Pseudocode**

```

- entry (id: aardvark noun, headword: aardvark)
  - pronunciation
    - transcription (text: a:rdva:rk)
  - sense...

```

**Example A.10. XML**

```

<entry id="aardvark-noun">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <headword>aardvark</headword>
  <pronunciation>
    <transcription>
      <text>a:rdva:rk</text>
    </transcription>
  </pronunciation>
</entry>

```

**Example A.11. JSON**

```

{
  "id": "aardvark-noun",
  "headword": "aardvark",
  "pronunciations": [{
    "transcriptions": [{"text": "a:rdva:rk"}]
  }],
  "senses": []
}

```

**Example A.12. RDF**

---

```

ex:aardvark-noun a dmlex:Entry;
  dmlex:headword "aardvark";
  dmlex:pronunciation [ a dmlex:Pronunciation;
    dmlex:transcription [ a dmlex:Transcription;
      dmlex:text "a:rdva:rk";
      dmlex:listingOrder "1"^^xsd:unsignedInt ];
    dmlex:listingOrder "1"^^xsd:unsignedInt ].

```

## A.1.4 Pronunciation given as a sound file

### Example A.13. Pseudocode

```

- entry (id: aardvark-noun, headword: aardvark)
  - pronunciation (soundFile: aardvark.mp3)
  - sense: ...

```

### Example A.14. XML

```

<entry id="aardvark-noun">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0"
  <headword>aardvark</headword>
  <pronunciation soundFile="aardvark.mp3"/>
</entry>

```

### Example A.15. JSON

```

{
  "id": "aardvark-noun",
  "headword": "aardvark",
  "pronunciations": [{
    "soundFile": "aardvark.mp3"
  }],
  "senses": []
}

```

### Example A.16. RDF

```

ex:aardvark-noun a dmlex:Entry;
  dmlex:pronunciation [ a dmlex:Pronunciation;
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:soundFile "aardvark.mp3" ];
  dmlex:headword "aardvark".

```

## A.1.5 Pronunciation given both ways

### Example A.17. Pseudocode

```

- entry (id: aardvark-noun, headword: aardvark)
  - pronunciation (soundFile: aardvark.mp3)

```

```
- transcription (text: a:rdva:rk)
- sense: ...
```

#### Example A.18. XML

```
<entry id="aardvark-noun">
  <xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <headword>aardvark</headword>
  <pronunciation soundFile="aardvark.mp3">
    <transcription>
      <text>a:rdva:rk</text>
    </transcription>
  </pronunciation>
</entry>
```

#### Example A.19. JSON

```
{
  "id": "aardvark-noun",
  "headword": "aardvark",
  "pronunciations": [{
    "soundFile": "aardvark.mp3",
    "transcriptions": [{"text": "a:rdva:rk"}]
  }],
  "senses": []
}
```

#### Example A.20. RDF

```
ex:aardvark-noun a dmlex:Entry;
  dmlex:pronunciation [ a dmlex:Pronunciation;
    dmlex:soundFile "aardvark.mp3";
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:transcription [ a dmlex:Transcription;
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:text "a:rdva:rk" ] ];
  dmlex:headword "aardvark".
```

### A.1.6 How to use `partOfSpeechTag` and `inflectedFormTag`

This is an entry from a hypothetical Irish dictionary for the headword "folúsghlantóir" ("vacuum cleaner"). The meaning of the various tags used in this entry is explained in the `partOfSpeechTag` and `inflectedFormTag` objects.

#### Example A.21. Pseudocode

```
- lexicographicResource (uri: http://example.com, langCode: ga)
  - entry (id: folúsghlantóir-n, headword: folúsghlantóir)
    - partOfSpeech (tag: n-masc)
    - inflectedForm (tag: sg-gen, text: folúsghlantóra)
```



- inflectedForm (tag: pl, text: folúsghlantóirí)
- sense: ...
- partOfSpeechTag (tag: n-masc, description: masculine noun)
- partOfSpeechTag (tag: n-fem, description: feminine noun)
- inflectedFormTag (tag: sg-gen, description: singular genitive)
- inflectedFormTag (tag: pl, description: plural)

#### Example A.22. XML

```
<lexicographicResource uri="http://example.com" langCode="ga">
  <entry id="folúsghlantóir-n">
    <headword>folúsghlantóir</headword>
    <partOfSpeech tag="n-masc"/>
    <inflectedForm tag="sg-gen">
      <text>folúsghlantóra</text>
    </inflectedForm>
    <inflectedForm tag="pl">
      <text>folúsghlantóirí</text>
    </inflectedForm>
  </entry>
  <inflectedFormTag tag="sg-gen" for="n-masc n-fem">
    <description>singular genitive</description>
  </inflectedFormTag>
  <inflectedFormTag tag="pl" for="n-masc n-fem">
    <description>plural</description>
  </inflectedFormTag>
  <partOfSpeechTag tag="n-masc">
    <description>masculine noun</description>
  </partOfSpeechTag>
  <partOfSpeechTag tag="n-fem">
    <description>feminine noun</description>
  </partOfSpeechTag>
</inflectedFormTag tag="sg-gen">
  <description>singular genitive</description>
</inflectedFormTag>
<inflectedFormTag tag="pl">
  <description>plural</description>
</inflectedFormTag>
</lexicographicResource>
```

#### Example A.23. JSON

```
{
  "uri": "http://example.com",
  "langCode": "ga",
  "entries": [{
    "id": "folúsghlantóir-n",
    "headword": "folúsghlantóir",
    "partsOfSpeech": ["n-masc"],
    "inflectedForms": [{
      "tag": "sg-gen",
      "text": "folúsghlantóra"
    }],
  }],
}
```

```

        "tag": "pl",
        "text": "folúsghlantóirí"
    }],
    "senses": []
  }],
  "partOfSpeechTags": [{
    "tag": "n-masc",
    "description": "masculine noun"
  }, {
    "tag": "n-fem",
    "description": "feminine noun"
  }],
  "inflectedFormTags": [{
    "tag": "sg-gen",
    "description": "singular genitive",
    "for": "n-masc n-fem"
  }, {
    "tag": "pl",
    "description": "plural",
    "for": "n-masc n-fem"
  }
  ]
}

```

#### Example A.24. RDF

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:inflectedFormTag [ a dmlex:InflectedFormTag;
    dmlex:description "singular genitive";
    dmlex:for "n-masc n-fem" ;
    dmlex:tag "sg-gen" ],
  [ a dmlex:InflectedFormTag;
    dmlex:tag "pl";
    dmlex:for "n-masc n-fem" ;
    dmlex:description "plural" ];
  dmlex:partOfSpeechTag [ a dmlex:PartOfSpeechTag;
    dmlex:tag "n-fem";
    dmlex:description "feminine noun" ],
  [ a dmlex:PartOfSpeechTag;
    dmlex:description "masculine noun";
    dmlex:tag "n-masc" ];
  dmlex:uri "http://example.com";
  dmlex:langCode "ga";
  dmlex:entry ex:folúsghlantóir-n.
ex:folúsghlantóir-n a dmlex:Entry;
  dmlex:partOfSpeech [
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:tag "n-masc" ];
  dmlex:headword "folúsghlantóir".

```

### A.1.7 Mapping controlled values to external inventories

This shows how to map the value of a tag such as `n-masc` and `n-fem` to items in an external inventory such as LexInfo.

#### Example A.25. Pseudocode

```

- partOfSpeechTag (tag: n-masc, description: masculine noun)
  - sameAs (uri: http://www.lexinfo.net/ontology/3.0/lexinfo#noun)
  - sameAs (uri:
http://www.lexinfo.net/ontology/3.0/lexinfo#masculine)
- partOfSpeechTag (tag: n-fem, description: feminine noun)
  - sameAs (uri: http://www.lexinfo.net/ontology/3.0/lexinfo#noun)
  - sameAs (uri:
http://www.lexinfo.net/ontology/3.0/lexinfo#feminine)

```

**Example A.26. XML**

```

<lexicographicResource langCode="en">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0"
  <partOfSpeechTag tag="n-masc">
    <description>masculine noun</description>
    <sameAs
      uri="http://www.lexinfo.net/ontology/3.0/lexinfo#noun"/>
    <sameAs
uri="http://www.lexinfo.net/ontology/3.0/lexinfo#masculine"/>
  </partOfSpeechTag>
  <partOfSpeechTag tag="n-fem">
    <description>feminine noun</description>
    <sameAs
      uri="http://www.lexinfo.net/ontology/3.0/lexinfo#noun"/>
    <sameAs
uri="http://www.lexinfo.net/ontology/3.0/lexinfo#feminine"/>
  </partOfSpeechTag>
</lexicographicResource>

```

**Example A.27. JSON**

```

{
  "langCode": "en",
  "partOfSpeechTags": [{
    "tag": "n-masc",
    "description": "masculine noun",
    "sameAs": [
      "http://www.lexinfo.net/ontology/3.0/lexinfo#noun",
      "http://www.lexinfo.net/ontology/3.0/lexinfo#masculine"
    ]
  }, {
    "tag": "n-fem",
    "description": "feminine noun",
    "sameAs": [
      "http://www.lexinfo.net/ontology/3.0/lexinfo#noun",
      "http://www.lexinfo.net/ontology/3.0/lexinfo#feminine"
    ]
  }
]}

```

**Example A.28. RDF**

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:langCode "en";
  dmlex:partOfSpeechTag [ a dmlex:PartOfSpeechTag;
    dmlex:tag "n-masc";
    dmlex:description "masculine noun";
    owl:dmlex:sameAs
      <http://www.lexinfo.net/ontology/3.0/lexinfo#noun>,
      <http://www.lexinfo.net/ontology/3.0/lexinfo#masculine> ],
  [ a dmlex:PartOfSpeechTag;
    owl:dmlex:sameAs
      <http://www.lexinfo.net/ontology/3.0/lexinfo#feminine>,
      <http://www.lexinfo.net/ontology/3.0/lexinfo#noun>;
    dmlex:tag "n-fem";
    dmlex:description "feminine noun" ].

```

### A.1.8 Defining a bilingual lexicographic resource

This defines a lexicographic resource where the source language is German and the translation language is English and the English translations are going to come with pronunciation transcriptions in English IPA.

*Example A.29. Pseudocode*

```

- lexicographicResource (title: My German-English Dictionary,
  uri: http://example.com, langCode: de)
- translationLanguage (langCode: en)

```

*Example A.30. XML*

```

<lexicographicResource title="My German-English Dictionary"
  uri="http://example.com" langCode="de">
  <title>My German-English Dictionary</title>
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <translationLanguage langCode="en"/>
</lexicographicResource>

```

*Example A.31. JSON*

```

{
  "title": "My German-English Dictionary",
  "uri": "http://example.com",
  "langCode": "de",
  "translationLanguages": ["en"]
}

```

*Example A.32. RDF*

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:uri "http://example.com";
  dmlex:translationLanguage "en" [
    dmlex:langCode "en" ;

```

```
dmlex:listingOrder 1 ] ;
dmlex:title "My German-English Dictionary";
dmlex:langCode "de".
```

### A.1.9 Defining a multilingual lexicographic resource

This defines a lexicographic resource where the source language is Irish and the translation languages are English, German and Czech.

#### Example A.33. Pseudocode

```
- lexicographicResource (title: My Irish-Multilingual Dictionary,
    uri: http://example.com, langCode: ga)
  - translationLanguage (langCode: en)
  - translationLanguage (langCode: de)
  - translationLanguage (langCode: cs)
```

#### Example A.34. XML

```
<lexicographicResource title="My Irish-Multilingual Dictionary"
  uri="http://example.com" langCode="ga">
  <title>My Irish-Multilingual Dictionary</title>
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <translationLanguage langCode="en"/>
  <translationLanguage langCode="de"/>
  <translationLanguage langCode="cs"/>
</lexicographicResource>
```

#### Example A.35. JSON

```
{
  "title": "My Irish-Multilingual Dictionary",
  "uri": "http://example.com",
  "langCode": "ga",
  "translationLanguages": ["en", "de", "cs"]
}
```

#### Example A.36. RDF

```
ex:lexicon a dmlex:LexicographicResource;
dmlex:title "My Irish-Multilingual Dictionary";
dmlex:uri "http://example.com";
dmlex:translationLanguage "cs", [
  dmlex:langCode "cs" ;
  dmlex:listingOrder 1
] , [
  dmlex:langCode "en" ;
  dmlex:listingOrder 2
] , [
```

```
dmlex:langCode "de" ] ;  
dmlex:listingOrder 3 ] ;  
dmlex:langCode "ga".
```

### A.1.10 How to use `headwordTranslation` in a bilingual lexicographic resource

This is an entry from a hypothetical English-German dictionary for English-speaking learners of German.

#### Example A.37. Pseudocode

```
- entry (id: doctor-n, headword: doctor)  
  - sense (id: doctor-n-1, indicator: medical doctor)  
    - headwordTranslation (text: Arzt)  
      - partOfSpeech (tag: n-masc)  
    - headwordTranslation (text: Ärztin)  
      - partOfSpeech (tag: n-fem)  
  - sense (id: doctor-n-2, indicator: academic title)  
    - headwordTranslation (text: Doktor)  
      - partOfSpeech (tag: n-masc)  
    - headwordTranslation (text: Doktorin)  
      - partOfSpeech (tag: n-fem)
```

#### Example A.38. XML

```
<entry id="doctor-n">  
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0"  
  <headword>doctor</headword>  
  <sense id="doctor-n-1">  
    <indicator>medical doctor</indicator>  
    <headwordTranslation langCode="de">  
      <text>Arzt</text>  
      <partOfSpeech tag="n-masc"/>  
    </headwordTranslation>  
    <headwordTranslation langCode="de">  
      <text>Ärztin</text>  
      <partOfSpeech tag="n-fem"/>  
    </headwordTranslation>  
  </sense>  
  <sense id="doctor-n-2">  
    <indicator>academic title</indicator>  
    <headwordTranslation langCode="de">  
      <text>Doktor</text>  
      <partOfSpeech tag="n-masc"/>  
    </headwordTranslation>  
    <headwordTranslation langCode="de">  
      <text>Doktorin</text>  
      <partOfSpeech tag="n-fem"/>  
    </headwordTranslation>  
  </sense>  
</entry>
```

#### Example A.39. JSON

```

{
  "id": "doctor-n",
  "headword": "doctor",
  "senses": [{
    "id": "doctor-n-1",
    "indicator": "medical doctor",
    "headwordTranslations": [{
      "text": "Arzt",
      "partsOfSpeech": ["n-masc"],
      "langCode": "de"
    }, {
      "text": "Ärztin",
      "partsOfSpeech": ["n-fem"],
      "langCode": "de"
    }
  ]
}, {
  "id": "doctor-n-2",
  "indicator": "academic title",
  "headwordTranslations": [{
    "text": "Doktor",
    "partsOfSpeech": ["n-masc"],
    "langCode": "de"
  }, {
    "text": "Doktorin",
    "partsOfSpeech": ["n-fem"],
    "langCode": "de"
  }
  ]
}]
}

```

*Example A.40. RDF*

```

ex:doctor-n a dmlex:Entry;
  dmlex:sense ex:doctor-n-1,
  ex:doctor-n-2;
  dmlex:headword "doctor".
ex:doctor-n-1 a dmlex:Sense;
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
    dmlex:text "Ärztin";
    dmlex:listingOrder "2"^^xsd:unsignedInt;
    dmlex:langCode "de"
  ]
  dmlex:partOfSpeech [
    dmlex:tag "n-fem";
    dmlex:listingOrder "1"^^xsd:unsignedInt ] ],
  [ a dmlex:HeadwordTranslation;
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:langCode "de";
    dmlex:text "Arzt";
    dmlex:partOfSpeech [
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:tag "n-masc" ] ];
  dmlex:indicator "medical doctor";
  dmlex:listingOrder "1"^^xsd:unsignedInt.

```

```

ex:doctor-n-2 a dmlex:Sense;
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
    dmlex:partOfSpeech [
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:tag "n-masc" ];
    dmlex:langCode "de";
    dmlex:text "Doktor";
    dmlex:listingOrder "1"^^xsd:unsignedInt ],
  [ a dmlex:HeadwordTranslation;
    dmlex:partOfSpeech [
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:tag "n-fem" ];
    dmlex:langCode "de";
    dmlex:listingOrder "2"^^xsd:unsignedInt;
    dmlex:text "Doktorin" ];
  dmlex:listingOrder "2"^^xsd:unsignedInt;
  dmlex:indicator "academic title".

```

### A.1.11 How to use `headwordTranslation` in a multilingual lexicographic resource

This is an entry from a hypothetical Irish-multilingual dictionary.

#### Example A.41. Pseudocode

```

- entry (id: fómhar-n, headword: fómhar)
  - sense (id: fómhar-n-1)
    - headwordTranslation (langCode: en, text: autumn)
    - headwordTranslation (langCode: en, text: fall)
    - headwordTranslation (langCode: de, text: Herbst)
    - headwordTranslation (langCode: cs, text: podzim)
  - sense (id: fómhar-n-2)
    - headwordTranslation (langCode: en, text: harvest)
    - headwordTranslation (langCode: de, text: Ernte)
    - headwordTranslation (langCode: cs, text: sklizeň)

```

#### Example A.42. XML

```

<entry id="fómhar-n">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <headword>fómhar</headword>
  <sense id="fómhar-n-1">
    <headwordTranslation langCode="en">
      <text>autumn</text>
    </headwordTranslation>
    <headwordTranslation langCode="en">
      <text>fall</text>
    </headwordTranslation>
    <headwordTranslation langCode="de">
      <text>Herbst</text>
    </headwordTranslation>
    <headwordTranslation langCode="cs">
      <text>podzim</text>
    </headwordTranslation>
  </sense>
  <sense id="fómhar-n-2">
    <headwordTranslation langCode="en">
      <text>harvest</text>
    </headwordTranslation>
    <headwordTranslation langCode="de">
      <text>Ernte</text>
    </headwordTranslation>
    <headwordTranslation langCode="cs">
      <text>sklizeň</text>
    </headwordTranslation>
  </sense>
</entry>

```



```

    </headwordTranslation>
  </sense>
  <sense id="fómhar-n-2">
    <headwordTranslation langCode="en">
      <text>harvest</text>
    </headwordTranslation>
    <headwordTranslation langCode="de">
      <text>Ernte</text>
    </headwordTranslation>
    <headwordTranslation langCode="cs">
      <text>sklizeň</text>
    </headwordTranslation>
  </sense>
</entry>

```

**Example A.43. JSON**

```

{
  "id": "fómhar-n",
  "headword": "fómhar",
  "senses": [{
    "id": "fómhar-n-1",
    "headwordTranslations": [{
      "langCode": "en",
      "text": "autumn"
    }, {
      "langCode": "en",
      "text": "fall"
    }, {
      "langCode": "de",
      "text": "Herbst"
    }, {
      "langCode": "cs",
      "text": "podzim"
    }
  ]
}, {
  "id": "fómhar-n-2",
  "headwordTranslations": [{
    "langCode": "en",
    "text": "harvest"
  }, {
    "langCode": "de",
    "text": "Ernte"
  }, {
    "langCode": "cs",
    "text": "sklizeň"
  }
  ]
}]
}

```

**Example A.44. RDF**

```

ex:fómhar-n-1 a dmlex:Sense;

```

```

dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
  dmlex:listingOrder "3"^^xsd:unsignedInt;
  dmlex:text "Herbst";
  dmlex:langCode "de" ],
[ a dmlex:HeadwordTranslation;
  dmlex:langCode "en";
  dmlex:listingOrder "1"^^xsd:unsignedInt;
  dmlex:text "autumn" ],
[ a dmlex:HeadwordTranslation;
  dmlex:listingOrder "2"^^xsd:unsignedInt;
  dmlex:langCode "en";
  dmlex:text "fall" ],
[ a dmlex:HeadwordTranslation;
  dmlex:text "podzim";
  dmlex:listingOrder "4"^^xsd:unsignedInt;
  dmlex:langCode "cs" ];
dmlex:listingOrder "1"^^xsd:unsignedInt.
ex:fómhar-n-2 a dmlex:Sense;
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
    dmlex:langCode "en";
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:text "harvest" ],
  [ a dmlex:HeadwordTranslation;
    dmlex:text "sklizeň";
    dmlex:langCode "cs";
    dmlex:listingOrder "3"^^xsd:unsignedInt ],
  [ a dmlex:HeadwordTranslation;
    dmlex:text "Ernte";
    dmlex:langCode "de";
    dmlex:listingOrder "2"^^xsd:unsignedInt ];
dmlex:listingOrder "2"^^xsd:unsignedInt.
ex:fómhar-n a dmlex:Entry;
  dmlex:headword "fómhar";
  dmlex:sense ex:fómhar-n-1,
  ex:fómhar-n-2.

```

## A.1.12 How to use `headwordExplanation`

### Example A.45. Pseudocode

```

- entry (id: treppenwitz, headword: Treppenwitz)
  - partOfSpeech (tag: n-masc)
  - sense (id: treppenwitz-1)
    - headwordExplanation (text: belated realisation of
      what one could have said)
    - headwordTranslation (text: staircase wit)

```

### Example A.46. XML

```

<entry id="treppenwitz">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <headword>Treppenwitz</headword>
  <partOfSpeech tag="n-masc"/>

```

```

    <sense id="treppenwitz-1">
      <headwordExplanation>belated realisation of what one could have
sa langCode="en">
id      <text>
      belated realisation of what one
      could have said
      </text>
    </headwordExplanation>
    <headwordTranslation langCode="en">
      <text>staircase wit</text>
    </headwordTranslation>
  </sense>
</entry>

```

Example A.47. JSON

```

{
  "id": "treppenwitz",
  "headword": "Treppenwitz",
  "partsOfSpeech": ["n-masc"],
  "senses": [{
    "id": "treppenwitz-1",
    "headwordExplanations": [{
      "text": "belated realisation of what one could have said",
      "langCode": "en"
    }],
    "headwordTranslations": [{
      "text": "staircase wit",
      "langCode": "en"
    }]
  }]
}

```

Example A.48. RDF

```

ex:treppenwitz a dmlex:Entry;
  dmlex:partOfSpeech [
    dmlex:tag "n-masc";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:sense ex:treppenwitz-1;
  dmlex:headword "Treppenwitz".
ex:treppenwitz-1 a dmlex:Sense;
  dmlex:headwordExplanation [ a dmlex:HeadwordExplanation;
    dmlex:text "belated realisation of what one could have said";
    dmlex:langCode "en" ];
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
    dmlex:text "staircase wit";
    dmlex:langCode "en";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:listingOrder "1"^^xsd:unsignedInt.

```

### A.1.13 Modelling parts and wholes

We have three entries with one sense each: "glasses", "microscope" and "lens". We want to represent the fact that "lens" is a meronym of both "glasses" and "microscope", and simultaneously that "glasses" and "microscope" are both holonyms of "lens".

*Example A.49. Pseudocode*

```
- lexicographicResource (uri: http://example.com, langCode: en)
  - entry (id: glasses, headword: glasses)
    - sense (id: glasses-1)
      - definition (text: an optical seeing aid)
  - entry (id: microscope, headword: microscope)
    - sense (id: microscope-1)
      - definition (text: equipment for looking at very small
things)
  - entry (id: lens, headword: lens)
    - sense (id: lens-1)
      - definition (text: curved glass that makes things seem
bigger)
  - relation (type: meronymy)
    - member (ref: glasses-1, role: whole)
    - member (ref: lens-1, role: part)
  - relation (type: meronymy)
    - member (ref: microscope-1, role: whole)
    - member (ref: lens-1, role: part)
  - relationType (type: meronomy, description: part-whole
relationship)
    - memberType (role: whole, type: sense, min: 1, max: 1,
hint: navigate)
    - memberType (role: part, type: sense, min: 1, max: 1,
hint: navigate)
```

*Example A.50. XML*

```
<lexicographicResource uri="http://example.com" langCode="en">
  <entry id="glasses">
    <headword>glasses</headword>
    <sense id="glasses-1">
      <definition>an optical seeing aid</definition>
    </sense>
  </entry>
  <entry id="microscope">
    <headword>microscope</headword>
    <sense id="microscope-1">
      <definition>
        <text>an optical seeing aid</text>
      </definition>
    </sense>
  </entry>
  <entry id="microscope">
    <headword>microscope</headword>
    <sense id="microscope-1">
      <definition>
        <text>equipment for looking at very small things</text>
```

```

        </definition>
    </sense>
</entry>
<entry id="lens">
    <headword>lens</headword>
    <sense id="lens-1">
        <definition>
            <text>curved glass that makes things seem bigger</text>
        </definition>
    </sense>
</entry>
<relation type="meronymy">
    <member ref="glasses-1" role="whole"/>
    <member ref="lens-1" role="part"/>
</relation>
<relation type="meronymy">
    <member ref="microscope-1" role="whole"/>
    <member ref="lens-1" role="part"/>
</relation>
<relationType type="meronymy">
    <description>part-whole relationship</description>
    <memberType role="whole" type="sense" min="1" max="1"
        hint="navigate"/>
    <memberType role="part" type="sense" min="1" max="1"
        hint="navigate"/>
</relationType>
</lexicographicResource>

```

**Example A.51. JSON**

```

{
  "uri": "http://example.com",
  "langCode": "en",
  "entries": [{
    "id": "glasses",
    "headword": "glasses",
    "senses": [{
      "id": "glasses-1",
      "definitions": [{"text": "an optical seeing aid"}]
    }], {
    "id": "microscope",
    "headword": "microscope",
    "senses": [{
      "id": "microscope-1",
      "definitions": [{"text":
        "equipment for looking at very small things"}]
    }], {
    "id": "lens",
    "headword": "lens",
    "senses": [{
      "id": "lens-1",
      "definitions": [{"text":
        "curved glass that makes things seem bigger"}]
    }
  ]
},
}

```

```

"relations": [{
  "type": "meronymy",
  "members": [{
    "ref": "glasses-1",
    "role": "whole"
  }, {
    "ref": "lens-1",
    "role": "part"
  }]
}, {
  "type": "meronymy",
  "members": [{
    "ref": "microscope-1",
    "role": "whole"
  }, {
    "ref": "lens-1",
    "role": "part"
  }]
}],
"relationTypes": [{
  "type": "meronymy",
  "description": "part-whole relationship",
  "memberTypes": [{
    "role": "whole",
    "type": "sense",
    "min": 1,
    "max": 1,
    "hint": "navigate"
  }, {
    "role": "part",
    "type": "sense",
    "min": 1,
    "max": 1,
    "hint": "navigate"
  }]
}]
}

```

**Example A.52. RDF**

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:entry ex:microscope,
  ex:lens,
  ex:glasses;
dmlex:relationType [ a dmlex:RelationType;
  dmlex:memberType [ a dmlex:MemberType;
    dmlex:min "1"^^xsd:unsignedInt;
    dmlex:type dmlex:sense;
    dmlex:hint dmlex:navigate;
    dmlex:max "1"^^xsd:unsignedInt;
    dmlex:role "part" ],
  [ a dmlex:MemberType;
    dmlex:max "1"^^xsd:unsignedInt;
    dmlex:type dmlex:sense;
    dmlex:role "whole";

```

```

        dmlex:min "1"^^xsd:unsignedInt;
        dmlex:hint dmlex:navigate ];
    dmlex:description "part-whole relationship";
    dmlex:type "meronymy" ];
dmlex:relation [ a dmlex:Relation;
    dmlex:member [ a dmlex:Member;
        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:ref "microscope-1";
        dmlex:role "whole" ],
        [ a dmlex:Member;
            dmlex:role "part";
            dmlex:listingOrder "2"^^xsd:unsignedInt;
            dmlex:ref "lens-1" ];
    dmlex:type "meronymy";
    dmlex:listingOrder "2"^^xsd:unsignedInt ],
[ a dmlex:Relation;
    dmlex:type "meronymy";
    dmlex:member [ a dmlex:Member;
        dmlex:ref "glasses-1";
        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:role "whole" ],
        [ a dmlex:Member;
            dmlex:role "part";
            dmlex:listingOrder "2"^^xsd:unsignedInt;
            dmlex:ref "lens-1" ];
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
dmlex:langCode "en";
dmlex:uri "http://example.com".
ex:glasses a dmlex:Entry;
    dmlex:headword "glasses";
    dmlex:sense ex:glasses-1.
ex:glasses-1 a dmlex:Sense;
    dmlex:definition [ a dmlex:Definition;
        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "an optical seeing aid" ];
    dmlex:listingOrder "1"^^xsd:unsignedInt.
ex:lens a dmlex:Entry;
    dmlex:sense ex:lens-1;
    dmlex:headword "lens".
ex:lens-1 a dmlex:Sense;
    dmlex:definition [ a dmlex:Definition;
        dmlex:text "curved glass that makes things seem bigger";
        dmlex:listingOrder "1"^^xsd:unsignedInt ];
    dmlex:listingOrder "1"^^xsd:unsignedInt.
ex:microscope a dmlex:Entry;
    dmlex:sense ex:microscope-1;
    dmlex:headword "microscope".
ex:microscope-1 a dmlex:Sense;
    dmlex:definition [ a dmlex:Definition;
        dmlex:text "equipment for looking at very small things";
        dmlex:listingOrder "1"^^xsd:unsignedInt ];
    dmlex:listingOrder "1"^^xsd:unsignedInt.

```

*Example A.53. Suggested rendering for human users*

glasses

```

- an optical seeing aid
  contains: lens

microscope
- equipment for looking at very small things
  contains: lens

lens
- curved glass that makes things seem bigger
  things that contain lens: glasses, microscope

```

### A.1.14 Modelling antonyms

We have two entries for the verbs "buy" and "sell" with one sense each. We want to express the fact that the senses are antonyms.

#### Example A.54. Pseudocode

```

- lexicographicResource (uri: http://example.com, langCode: en)
  - entry (id: buy, headword: buy)
    - sense (id: buy-1)
      - definition (text: get something by paying money for it)
  - entry (id: sell, headword: sell)
    - sense (id: sell-1)
      - definition (text: exchange something for money)
  - relation (type: antonyms)
    - member (ref: buy-1)
    - member (ref: sell-1)
  - relationType: (type: antonyms)
    - memberType (type: sense, min: 2, max: 2, hint: navigate)

```

#### Example A.55. XML

```

<lexicographicResource uri="http://example.com" langCode="en">
  <xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
    <entry id="buy">
      <headword>buy</headword>
      <sense id="buy-1">
        <definition>
          <text>get something by paying money for it</text>
        </definition>
      </sense>
    </entry>
    <entry id="sell">
      <headword>sell</headword>
      <sense id="sell-1">
        <definition>
          <text>exchange something for money</text>
        </definition>
      </sense>
    </entry>
    <relation type="antonyms">

```



```

    <member ref="buy-1"/>
    <member ref="sell-1"/>
  </relation>
  <relationType type="antonyms">
    <memberType type="sense" min="2" max="2" hint="navigate"
      role="obverse"/>
  </relationType>
</lexicographicResource>

```

**Example A.56. JSON**

```

{
  "uri": "http://example.com",
  "langCode": "en",
  "entries": [{
    "id": "buy",
    "headword": "buy",
    "senses": [{
      "id": "buy-1",
      "definitions": [{"text":
        "get something by paying money for it"}]
    }], {
    "id": "sell",
    "headword": "sell",
    "senses": [{
      "id": "sell-1",
      "definitions": [{"text": "exchange something for money"}]
    }
  ]
}],
  "relations": [{
    "type": "antonyms",
    "members": [
      {"ref": "buy-1"},
      {"ref": "sell-1"}
    ]
  }
],
  "relationTypes": [{
    "type": "antonyms",
    "memberTypes": [{
      "role": "obverse",
      "type": "sense",
      "min": 2,
      "max": 2,
      "hint": "navigate"
    }
  ]
}]
}

```

**Example A.57. RDF**

```

[] a dmlex:LexicographicResource;
  dmlex:uri "http://example.com";
  dmlex:langCode "en";
  dmlex:relationType [ a dmlex:RelationType;

```

```

dmlex:type "antonyms";
dmlex:memberType [ a dmlex:MemberType;
  dmlex:type dmlex:sense;
  dmlex:max "2"^^xsd:unsignedInt;
  dmlex:min "2"^^xsd:unsignedInt;
  dmlex:role "obverse";
  dmlex:hint dmlex:navigate ] ];
dmlex:entry ex:buy,
  ex:sell;
dmlex:relation [ a dmlex:Relation;
  dmlex:type "antonyms";
  dmlex:member [ a dmlex:Member;
    dmlex:ref "sell-1";
    dmlex:listingOrder "2"^^xsd:unsignedInt ],
  [ a dmlex:Member;
    dmlex:ref "buy-1";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:listingOrder "1"^^xsd:unsignedInt ].
ex:sell a dmlex:Entry;
  dmlex:headword "sell";
  dmlex:sense ex:sell-1.
ex:sell-1 a dmlex:Sense;
  dmlex:definition [ a dmlex:Definition;
    dmlex:text "exchange something for money";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:listingOrder "1"^^xsd:unsignedInt.
ex:buy a dmlex:Entry;
  dmlex:sense ex:buy-1;
  dmlex:headword "buy".
ex:buy-1 a dmlex:Sense;
  dmlex:listingOrder "1"^^xsd:unsignedInt;
  dmlex:definition [ a dmlex:Definition;
    dmlex:text "get something by paying money for it";
    dmlex:listingOrder "1"^^xsd:unsignedInt ].

```

*Example A.58. Suggested rendering for human users*

```

buy
- get something by paying money for it
  opposite meaning: sell

sell
- exchange something for money
  opposite meaning: buy

```

### A.1.15 Modelling synonyms

We have three German entries with one sense each, two which mean "sea" and one which means "ocean". We want to set up a relation which brings these three sense together as near-synonyms.

*Example A.59. Pseudocode*

```

- lexicographicResource (uri: http://example.com, langCode: de)
  - translationLanguage (langCode: en)
  - entry (id: die-see, headword: See)
    - partOfSpeech (tag: n-fem)
    - sense (id: die-see-1)
      - headwordTranslation (text: sea)
  - entry (id: das-meer, headword: Meer)
    - partOfSpeech (tag: n-neut)
    - sense (id: das-meer-1)
      - headwordTranslation (text: sea)
  - entry (id: der-ozean, headword: Ozean)
    - partOfSpeech (tag: n-masc)
    - sense (id: der-ozean-1)
      - headwordTranslation (text: ocean)
  - relation (type: synonyms, description: words that mean sea and
ocean)
    - member (ref: die-see-1)
    - member (ref: das-meer-1)
    - member (ref: der-ozean-1)
  - relationType (type: synonyms, description: synonyms and near
synonyms)
    memberType (type: sense, min: 2, hint: navigate)

```

**Example A.60. XML**

```

<lexicographicResource uri="http://example.com" langCode="de">
  <xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
<translationLanguage langCode="en"/>
    <entry id="die-see">
      <headword>See</headword>
      <partOfSpeech tag="n-fem"/>
      <sense id="die-see-1">
        <headwordTranslation><text>sea</text></headwordTranslation>
      </sense>
    </entry>
    <entry id="das-meer">
      <headword>Meer</headword>
      <partOfSpeech tag="n-neut"/>
      <sense id="das-meer-1">
        <headwordTranslation><text>sea</text></headwordTranslation>
      </sense>
    </entry>
    <entry id="der-ozean">
      <headword>Ozean</headword>
      <partOfSpeech tag="n-masc"/>
      <sense id="der-ozean-1">
<headwordTranslation><text>ocean</text></headwordTranslation>
        </sense>
      </entry>
<translationLanguage langCode="en"/>
    <relation type="synonyms">
      <description>words that mean sea and ocean</description>
      <member ref="die-see-1"/>
      <member ref="das-meer-1"/>

```

```

    <member ref="der-ozean-1"/>
  </relation>
  <relationType type="synonyms">
    <description>synonyms and near synonyms</description>
    <memberType type="sense" min="2" hint="navigate"/>
  </relationType>
</lexicographicResource>

```

**Example A.61. JSON**

```

{
  "uri": "http://example.com",
  "langCode": "de",
  "translationLanguages": ["en"],
  "entries": [
    {
      "id": "die-see",
      "headword": "See",
      "partsOfSpeech": ["n-fem"],
      "senses": [
        {
          "id": "die-see-1",
          "headwordTranslations": [{"text": "sea"}]
        }
      ]
    },
    {
      "id": "das-meer",
      "headword": "Meer",
      "partsOfSpeech": ["n-neut"],
      "senses": [
        {
          "id": "das-meer-1",
          "headwordTranslations": [{"text": "sea"}]
        }
      ]
    },
    {
      "id": "der-ozean",
      "headword": "Ozean",
      "partsOfSpeech": ["n-masc"],
      "senses": [
        {
          "id": "der-ozean-1",
          "headwordTranslations": [{"text": "ocean"}]
        }
      ]
    }
  ],
  "relations": [
    {
      "type": "synonyms",
      "description": "words that mean sea and ocean",
      "members": [
        {"ref": "die-see-1"},
        {"ref": "das-meer-1"},
        {"ref": "der-ozean-1"}
      ]
    }
  ],
  "relationTypes": [
    {
      "type": "synonyms",
      "description": "synonyms and near synonyms",
      "memberTypes": [
        {
          "type": "sense",
          "min": 2,
          "hint": "navigate"
        }
      ]
    }
  ]
}

```

```

    }]
  }]
}

```

*Example A.62. RDF*

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:langCode "de";
  dmlex:entry ex:die-see,
    ex:das-meer,
    ex:der-ozean;
  dmlex:uri "http://example.com";
  dmlex:relation [ a dmlex:Relation;
    dmlex:type "synonyms";
    dmlex:member [ a dmlex:Member;
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:ref "die-see-1" ],
      [ a dmlex:Member;
        dmlex:listingOrder "2"^^xsd:unsignedInt;
        dmlex:ref "das-meer-1" ],
      [ a dmlex:Member;
        dmlex:ref "der-ozean-1";
        dmlex:listingOrder "3"^^xsd:unsignedInt ];
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:description "words that mean sea and ocean" ];
  dmlex:translationLanguage "en"; [
    dmlex:langCode "en" ;
    dmlex:listingOrder 1 ] ;
  dmlex:relationType [ a dmlex:RelationType;
    dmlex:description "synonyms and near synonyms";
    dmlex:memberType [ a dmlex:MemberType;
      dmlex:hint dmlex:navigate;
      dmlex:type dmlex:sense;
      dmlex:min "2"^^xsd:unsignedInt ];
    dmlex:type "synonyms" ].
ex:der-ozean a dmlex:Entry;
  dmlex:sense ex:der-ozean-1;
  dmlex:headword "Ozean";
  dmlex:partOfSpeech [
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:tag "n-masc" ].
ex:der-ozean-1 a dmlex:Sense;
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:text "ocean" ];
  dmlex:listingOrder "1"^^xsd:unsignedInt.
ex:das-meer a dmlex:Entry;
  dmlex:headword "Meer";
  dmlex:partOfSpeech [
    dmlex:tag "n-neut";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:sense ex:das-meer-1.
ex:das-meer-1 a dmlex:Sense;
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;

```

```

    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:text "sea" ];
  dmlex:listingOrder "1"^^xsd:unsignedInt.
ex:die-see a dmlex:Entry;
  dmlex:partOfSpeech [
    dmlex:tag "n-fem";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:sense ex:die-see-1;
  dmlex:headword "See".
ex:die-see-1 a dmlex:Sense;
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:text "sea" ];
  dmlex:listingOrder "1"^^xsd:unsignedInt.

```

*Example A.63. Suggested rendering for human users*

```

See, feminine noun
- see
  same or similar meaning: Meer, Ozean

Meer, neuter noun
- see
  same or similar meaning: See, Ozean

Ozean, masculine noun
- ocean
  same or similar meaning: See, Meer

```

## A.1.16 Modelling variants

We have two entries in our lexicographic resource, one for the headword "colour" and one for the headword "color". We want to create a relation to represent the fact that these are spelling variants. One of the entries is fully fleshed-out (has a sense with a definition, an example etc) while the other one is only skeletal: its only purpose is to serve as the origin of a navigable link to the other entry.

*Example A.64. Pseudocode*

```

- lexicographicResource (uri: http://example.com, langCode: en)
  - entry (id: colour, headword: colour)
    - partOfSpeech (tag: n)
    - label (tag: europeanSpelling)
    - sense (id: colour-1)
      - definition (text: red, blue, yellow etc.)
      - example (text: What is your favourite colour?)
  - entry (id: color, headword: color)
    - partOfSpeech (tag: n)
    - label (tag: americanSpelling)
  - relation (type: variants)
    - member (ref: colour)
    - member (ref: color)
  - relationType (type: variants, description: words which differ
    only in spelling)

```

- memberType (type: entry, min: 2, hint: navigate)

#### Example A.65. XML

```
<lexicographicResource uri="http://example.com" langCode="en"
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <entry id="colour">
    <headword>colour</headword>
    <partOfSpeech tag="n"/>
    <label tag="europeanSpelling"/>
    <sense id="colour-1">
      <definition>red, blue, yellow etc.</definition>
      <text>red, blue, yellow etc.</text>
    </definition>
    <example><>
      <text>What is your favourite colour?</text></>
    </example>
  </sense>
</entry>
<entry id="color">
  <headword>color</headword>
  <partOfSpeech tag="n"/>
  <label tag="americanSpelling"/>
</entry>
<relation type="variants">
  <member ref="colour"/>
  <member ref="color"/>
</relation>
<relationType type="variants">
  <description>words which differ only in spelling</description>
  <memberType type="entry" min="2" hint="navigate"/>
</relationType>
</lexicographicResource>
```

#### Example A.66. JSON

```
{
  "uri": "http://example.com",
  "langCode": "en",
  "entries": [{
    "id": "colour",
    "headword": "colour",
    "partsOfSpeech": ["n"],
    "labels": ["europeanSpelling"],
    "senses": [{
      "id": "colour-1",
      "definitions": [{"text": "red, blue, yellow etc."}],
      "examples": [{"text": "What is your favourite colour?"}]
    }]
  }, {
    "id": "color",
    "headword": "color",
    "partsOfSpeech": ["n"],
```

```

    "labels": ["americanSpelling"]
  }],
  "relations": [{
    "type": "variants",
    "members": [
      {"ref": "colour"},
      {"ref": "color"}
    ]
  }],
  "relationTypes": [{
    "type": "variants",
    "description": "words which differ only in spelling",
    "memberTypes": [{
      "type": "entry",
      "min": 2,
      "hint": "navigate"
    }]
  }]
}

```

*Example A.67. RDF*

*Example A.68. Suggested rendering for human users*

```

colour (noun, European spelling), variant: color
- red, blue, yellow etc.
  "What is your favourite colour?"

color (noun, American spelling), see: colour

```

### A.1.17 Modelling subsenses

We have an entry for the noun "colour" with four senses. We want to express the fact that senses number two and three are subsenses of sense number one, and should be displayed as such to human users.

*Example A.69. Pseudocode*

```

- lexicographicResource (uri: http://example.com, langCode: en)
  - entry (id: colour, headword: colour)
    - sense (id: colour-1)
      - definition (text: red, blue, yellow etc.)
      - example (text: What is your favourite colour?)
    - sense (id: colour-2)
      - definition (text: not being black and white)
      - example (text: Owning a colour TV meant you were rich.)
    - sense (id: colour-3)
      - definition (text: a sign of a person's race)
      - example (text: People of all creeds and colours.)
    - sense (id: colour-4)
      - definition (text: interest or excitement)
      - example (text: Examples add colour to your writing.)

```



- relation (type: subsensing)
  - member (rer: colour-1, role: super)
  - member (ref: colour-2, role: sub)
- relation (type: subsensing)
  - member (ref: colour-1, role: super)
  - member (ref: colour-3, role: sub)
- relationType (type: subsensing, scopeRestriction: sameEntry)
  - memberType (role: super, type: sense, min: 1, max: 1, hint: none)
  - memberRole (role: sub, type: sense, min: 1, max: 1, hint: embed)

**Example A.70. XML**

```

<lexicographicResource uri="http://example.com" langCode="en"
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <entry id="colour">
    <headword>colour</headword>
    <sense id="colour-1">
      <definition>
        <text>red, blue, yellow etc.</text>
      </definition>
      <example>
        <text>What is your favourite colour?</text>
      </example>
    </sense>
    <sense id="colour-2">
      <definition>
        <text>not being black and white</text>
      </definition>
      <example>
        <text>Owning a colour TV meant you were rich.</text>
      </example>
    </sense>
    <sense id="colour-3">
      <definition>
        <text>a sign of a person's race</text>
      </definition>
      <example>
        <text>People of all creeds and colours.</text>
      </example>
    </sense>
    <sense id="colour-4">
      <definition>
        <text>interest or excitement</text>
      </definition>
      <example>
        <text>Examples add colour to your writing.</text>
      </example>
    </sense>
  </entry>
  <relation type="subsensing">
    <member ref="colour-1" role="super"/>
    <member ref="colour-2" role="sub"/>
  </relation>

```

```

<relation type="subsensing">
  <member ref="colour-1" role="super"/>
  <member ref="colour-3" role="sub"/>
</relation>
<relationType type="subsensing" scopeRestriction="sameEntry">
  <memberType role="super" type="sense" min="1"
    max="1" hint="none"/>
  <memberType role="sub" type="sense" min="1"
    max="1" hint="embed"/>
</relationType>
</lexicographicResource>

```

**Example A.71. JSON**

```

{
  "uri": "http://example.com",
  "langCode": "en",
  "entries": [{
    "id": "colour",
    "headword": "colour",
    "senses": [{
      "id": "colour-1",
      "definitions": [{"text": "red, blue, yellow etc."}],
      "examples": [{"text":
        "What is your favourite colour?"}]
    }, {
      "id": "colour-2",
      "definitions": [{"text": "not being black and white"}],
      "examples": [{"text":
        "Owning a colour TV meant you were rich."}]
    }, {
      "id": "colour-3",
      "definitions": [{"text": "a sign of a person's race"}],
      "examples": [{"text":
        "People of all creeds and colours."}]
    }, {
      "id": "colour-4",
      "definitions": [{"text": "interest or excitement"}],
      "examples": [{"text":
        "Examples add colour to your writing."}]
    }
  ]
}, {
  "relations": [{
    "type": "subsensing",
    "members": [
      {"role": "super", "ref": "colour-1"},
      {"role": "sub", "ref": "colour-2"}
    ]
  }, {
    "type": "subsensing",
    "members": [
      {"role": "super", "ref": "colour-1"},
      {"role": "sub", "ref": "colour-3"}
    ]
  }
],
}

```

```

"relationTypes": [{
  "type": "subsensing",
  "scopeRestriction": "sameEntry",
  "memberTypes": [{
    "role": "super",
    "type": "sense",
    "min": 1,
    "max": 1,
    "hint": "none"
  }, {
    "role": "sub",
    "type": "sense",
    "min": 1,
    "max": 1,
    "hint": "embed"
  }
  ]
}]
}

```

**Example A.72. RDF**

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:entry ex:colour;
  dmlex:relation [ a dmlex:Relation;
    dmlex:member [ dmlex:ref "colour-3";
      dmlex:listingOrder "2"^^xsd:unsignedInt;
      dmlex:role "sub" ], [
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:role "super";
      dmlex:ref "colour-1" ];
    dmlex:type "subsensing";
    dmlex:listingOrder "2"^^xsd:unsignedInt ], [
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:type "subsensing";
    dmlex:member [ dmlex:ref "colour-2";
      dmlex:role "sub";
      dmlex:listingOrder "2"^^xsd:unsignedInt ], [
      dmlex:role "super";
      dmlex:ref "colour-1";
      dmlex:listingOrder "1"^^xsd:unsignedInt ] ];
  dmlex:langCode "en";
  dmlex:uri "http://example.com";
  dmlex:relationType [ dmlex:scopeRestriction dmlex:sameEntry;
    dmlex:type "subsensing";
    dmlex:memberType [ dmlex:min 1; dmlex:max 1;
      dmlex:type dmlex:sense;
      dmlex:role "sub";
      dmlex:hint dmlex:embed ], [
      dmlex:min 1; dmlex:max: 1;
      dmlex:hint dmlex:none;
      dmlex:type dmlex:sense;
      dmlex:role "super" ] ].
ex:colour a dmlex:Entry;
  dmlex:sense ex:colour-4, ex:colour-3, ex:colour-2, ex:colour-1;

```

```

    dmlex:headword "colour".
ex:colour-1 a dmlex:Sense;
    dmlex:definition [ dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "red, blue, yellow etc." ];
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:example [ dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "What is your favourite colour?" ].
ex:colour-2 a dmlex:Sense;
    dmlex:example [ dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "Owning a colour TV meant you were rich." ];
    dmlex:listingOrder "2"^^xsd:unsignedInt;
    dmlex:definition [ dmlex:text "not being black and white";
        dmlex:listingOrder "1"^^xsd:unsignedInt ].
ex:colour-3 a dmlex:Sense;
    dmlex:listingOrder "3"^^xsd:unsignedInt;
    dmlex:definition [ dmlex:text "a sign of a person's race";
        dmlex:listingOrder "1"^^xsd:unsignedInt ];
    dmlex:example [ dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "People of all creeds and colours." ].
ex:colour-4 a dmlex:Sense;
    dmlex:listingOrder "4"^^xsd:unsignedInt;
    dmlex:definition [ dmlex:text "interest or excitement";
        dmlex:listingOrder "1"^^xsd:unsignedInt ];
    dmlex:example [ dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "Examples add colour to your writing." ].

```

*Example A.73. Suggested rendering for human users*

```

colour
  1. red, blue, yellow etc. "What is your favourite colour?"
     (a) not being black and white "Owning a colour TV meant you were
rich."
     (b) a sign of a person's race "People of all creeds and colours.."
  2. interest or excitement "Examples add colour to your writing."

```

### A.1.18 Modelling subentries (at subsense level)

We have an entry for the adjective "safe" with two senses, and an entry for the multi-word expression "better safe than sorry" with one sense. We want to express the fact that the multi-word entry should appear under the first sense of "safe" as a subentry.

*Example A.74. Pseudocode*

```

- lexicographicResource (uri: http://example.com, langCode: en)
  - entry (id: safe, headword: safe)
    - sense (id: safe-1, indicator: protected from harm)
      - example (text: It isn't safe to park here.)
    - sense (id: safe-2, indicator: not likely to cause harm)
      - example (text: Is the ride safe for a small child?)
  - entry (id: better-safe, headword: better safe than sorry)
    - sense (id: better-safe-1)
      - definition (text: you should be careful even if it seems
unnecessary)

```

- relation (type: subentrying)
  - member (ref: safe-1, role: container)
  - member (ref: better-safe, role: subentry)
- relationType (type: subentrying, scopeRestriction: sameResource)
  - memberType (role: container, type: sense, min: 1, max: 1, hint: navigate)
  - memberType (role: subentry, type: entry, min: 1, max: 1, hint: embed)

**Example A.75. XML**

```

<lexicographicResource uri="http://example.com" langCode="en">
  <xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
    <entry id="safe">
      <headword>safe</headword>
      <sense id="safe-1">
        <indicator>protected from harm</indicator>
        <example>
          <text>It isn't safe to park here.</text>
        </example>
      </sense>
      <sense id="safe-2">
        <indicator>not likely to cause harm</indicator>
        <example>
          <text>Is the ride safe for a small child?</text>
        </example>
      </sense>
    </entry>
    <entry id="better-safe">
      <headword>better safe than sorry</headword>
      <sense id="better-safe-1">
        <definition>
          <text>you should be careful even if it seems
unnecessary</
text>
        </definition>
      </sense>
    </entry>
    <relation type="subentrying">
      <member ref="safe-1" role="container"/>
      <member ref="better-safe" role="subentry"/>
    </relation>
    <relationType type="subentrying" scopeRestriction="sameResource">
      <memberType role="container" type="sense" min="1" max="1"
hint="navigate"/>
      <memberType role="subentry" type="entry" min="1" max="1"
hint="embed"/>
    </relationType>
  </lexicographicResource>

```

**Example A.76. JSON**

```
{
```

```

"uri": "http://example.com",
"langCode": "en",
"entries": [{
  "id": "safe",
  "headword": "safe",
  "senses": [{
    "id": "safe-1",
    "indicator": "protected from harm",
    "examples": [{"text": "It isn't safe to park here."}]
  }, {
    "id": "safe-2",
    "indicator": "not likely to cause harm",
    "examples": [{"text": "Is the ride safe for a small
child?"}]
  }
], {
  "id": "better-safe",
  "headword": "better safe than sorry",
  "senses": [{
    "id": "better-safe-1",
    "definitions": [{
      "text": "you should be careful even if it seems
unnecessar
y"
    }
  ]
}],
"relations": [{
  "type": "subentrying",
  "members": [
    {"role": "container", "ref": "safe-1"},
    {"role": "subentry", "ref": "better-safe"}
  ]
}],
"relationTypes": [{
  "type": "subentrying",
  "scopeRestriction": "sameResource",
  "memberTypes": [{
    "role": "container",
    "type": "sense",
    "min": 1,
    "max": 1,
    "hint": "navigate"
  }, {
    "role": "subentry",
    "type": "entry",
    "min": 1,
    "max": 1,
    "hint": "embed"
  }
]
}]
}

```

Example A.77. RDF

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:entry ex:safe,
    ex:better-safe;
  dmlex:relation [ a dmlex:Relation;
    dmlex:member [ dmlex:ref "safe-1";
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:role "container" ], [
      dmlex:listingOrder "2"^^xsd:unsignedInt;
      dmlex:ref "better-safe";
      dmlex:role "subentry" ];
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:type "subentrying" ];
  dmlex:uri "http://example.com";
  dmlex:relationType [ dmlex:type "subentrying";
    dmlex:memberType [ a dmlex:MemberType;
      dmlex:type dmlex:entry;
      dmlex:hint dmlex:embed;
      dmlex:min 1 ; dmlex:max 1 ;
      dmlex:role "subentry" ], [
      dmlex:hint dmlex:navigate;
      dmlex:role "container";
      dmlex:min 1 ; dmlex:max 1;
      dmlex:type dmlex:sense ];
    dmlex:scopeRestriction dmlex:sameResource ];
  dmlex:langCode "en".
ex:safe a dmlex:Entry;
  dmlex:headword "safe";
  dmlex:sense ex:safe-1, ex:safe-2.
ex:safe-1 a dmlex:Sense;
  dmlex:listingOrder "1"^^xsd:unsignedInt;
  dmlex:indicator "protected from harm";
  dmlex:example [ dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:text "It isn't safe to park here." ].
ex:safe-2 a dmlex:Sense;
  dmlex:example [ dmlex:text "Is the ride safe for a small child?";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:listingOrder "2"^^xsd:unsignedInt;
  dmlex:indicator "not likely to cause harm".
ex:better-safe a dmlex:Entry;
  dmlex:sense ex:better-safe-1;
  dmlex:headword "better safe than sorry".
ex:better-safe-1 a dmlex:Sense;
  dmlex:listingOrder "1"^^xsd:unsignedInt;
  dmlex:definition [ dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:text "you should be careful even if it seems unnecessary"
  ].

```

*Example A.78. Suggested rendering for human users*

```

safe
- protected from harm "It isn't safe to park here."
  better safe than sorry
    - you should be careful even if it seems unnecessary
- not likely to cause harm "Is the ride safe for a small child?"

```

better safe than sorry

- you should be careful even if it seems unnecessary
- see also: safe

### A.1.19 Modelling subentries (at sense level)

We have an entry for the word "bible" and another entry for the expression "the Bible". We want to make sure that, when a human user is viewing the entry for "bible", the entry for "the Bible" is shown as a subentry of it, as if it were its first sense.

*Example A.79. Pseudocode*

```
- lexicographicResource (uri: http://example.com, langCode: en)
  - entry (id: the-bible, headword: the Bible)
    - sense (id: the-bible-1)
      - definition (text: the book considered holy by Christians)
  - entry (id: bible, headword: bible)
    - sense (id: bible-1)
    - sense (id: bible-2)
      - definition (text: a book considered important for a
subject)
  - relation (type: subentrying)
    - member (ref: bible-1, role: container)
    - member (ref: the-bible, role: subentry)
  - relationType (type: subentrying, scopeRestriction: sameResource)
    - memberType (role: container, type: sense, min: 1, max: 1,
hint: navigate)
    - memberRole (role: subentry, type: entry, min: 1, max: 1,
hint: embed)
```

*Example A.80. XML*

```
<lexicographicResource uri="http://example.com" langCode="en"
xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <entry id="the-bible">
    <headword>the Bible</headword>
    <sense id="the-bible-1">
      <definition>
        <text>the book considered holy by Christians</text>
      </definition>
    </sense>
  </entry>
  <entry id="bible">
    <headword>bible</headword>
    <sense id="bible-1"/>
    <sense id="bible-2">
      <definition>
        <text>a book considered important for a subject</text>
      </definition>
    </sense>
  </entry>
```



```

<relation type="subentrying">
  <member ref="bible-1" role="container"/>
  <member ref="the-bible" role="subentry"/>
</relation>
<relationType type="subentrying" scopeRestriction="sameResource">
  <memberType role="container" type="sense" min="1" max="1"
    hint="navigate"/>
  <memberType role="subentry" type="entry" min="1" max="1"
    hint="embed"/>
</relationType>
</lexicographicResource>

```

**Example A.81. JSON**

```

{
  "uri": "http://example.com",
  "langCode": "en",
  "entries": [{
    "id": "the-bible",
    "headword": "the Bible",
    "senses": [{
      "id": "the-bible-1",
      "definitions": [{"text":
        "the book considered holy by Christians"}]
    }]
  }, {
    "id": "bible",
    "headword": "bible",
    "senses": [{
      "id": "bible-1"
    }, {
      "id": "bible-2",
      "definitions": [{"text":
        "a book considered important for a subject"}]
    }]
  }],
  "relations": [{
    "type": "subentrying",
    "members": [
      {"role": "container", "ref": "bible-1"},
      {"role": "subentry", "ref": "the-bible"}
    ]
  }],
  "relationTypes": [{
    "type": "subentrying",
    "scopeRestriction": "sameResource",
    "memberTypes": [{
      "role": "container",
      "type": "sense",
      "min": 1,
      "max": 1,
      "hint": "navigate"
    }, {
      "role": "subentry",

```

```

        "type": "entry",
        "min": 1,
        "max": 1,
        "hint": "embed"
    }
  }
}

```

**Example A.82. RDF**

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:langCode "en";
  dmlex:entry ex:the-bible,
    ex:bible;
  dmlex:relationType [ a dmlex:RelationType;
    dmlex:type "subentrying";
    dmlex:memberType [ a dmlex:MemberType;
      dmlex:hint dmlex:embed;
      dmlex:role "subentry";
      dmlex:max "1"^^xsd:unsignedInt;
      dmlex:type dmlex:entry;
      dmlex:min "1"^^xsd:unsignedInt ],
    [ a dmlex:MemberType;
      dmlex:hint dmlex:navigate;
      dmlex:max "1"^^xsd:unsignedInt;
      dmlex:type dmlex:sense;
      dmlex:min "1"^^xsd:unsignedInt;
      dmlex:role "container" ];
    dmlex:scopeRestriction dmlex:sameResource ];
  dmlex:uri "http://example.com";
  dmlex:relation [ a dmlex:Relation;
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:type "subentrying";
    dmlex:member [ a dmlex:Member;
      dmlex:listingOrder "2"^^xsd:unsignedInt;
      dmlex:ref "the-bible";
      dmlex:role "subentry" ],
    [ a dmlex:Member;
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:ref "bible-1";
      dmlex:role "container" ] ].
ex:bible a dmlex:Entry;
  dmlex:sense ex:bible-1,
    ex:bible-2;
  dmlex:headword "bible".
ex:bible-1 a dmlex:Sense;
  dmlex:listingOrder "1"^^xsd:unsignedInt.
ex:bible-2 a dmlex:Sense;
  dmlex:definition [ a dmlex:Definition;
    dmlex:text "a book considered important for a subject";
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:listingOrder "2"^^xsd:unsignedInt.
ex:the-bible a dmlex:Entry;
  dmlex:sense ex:the-bible-1;

```

```

    dmlex:headword "the Bible".
ex:the-bible-1 a dmlex:Sense;
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:definition [ a dmlex:Definition;
        dmlex:text "the book considered holy by Christians";
        dmlex:listingOrder "1"^^xsd:unsignedInt ].

```

**Example A.83. Suggested rendering for human users**

```

bible
- the Bible
  - the book considered holy by Christians
- a book considered important for a subject

the Bible
- the book considered holy by Christians
  see also: bible

```

### A.1.20 Using placeholderMarker

**Example A.84. Pseudocode**

```

- entry (id: continue-studies, headword: continue your studies,
        placeholderMarker: "your")
  - sense ...

```

**Example A.85. XML**

```

<entry id="continue-studies">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0"
  <headword>continue <placeholderMarker>your</placeholderMarker>
  studies
</headword>
</entry>

```

**Example A.86. JSON**

```

{
  "id": "continue-studies",
  "headword": "continue your studies",
  "placeholderMarkers": [
    {"startIndex": 9, "endIndex": 13}
  ],
  "senses": []
}

```

**Example A.87. RDF**

```

ex:continue-studies a dmlex:Entry;
  dmlex:headword "continue your studies";
  dmlex:placeholderMarker [
    dmlex:startIndex "9"^^xsd:unsignedInt;
    dmlex:endIndex "13"^^xsd:unsignedInt ].

```

## A.1.21 Using `placeholderMarker` in a bilingual lexicographic resource

### Example A.88. Pseudocode

```

- entry (id: beat-up, headword: beat sb. up, placeholderMarker: "sb.")
  - sense (id: beat-up-1)
    headwordTranslation (text: jemanden verprügeln,
      placeholderMarker: "jemanden")

```

### Example A.89. XML

```

<entry id="beat-up">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0"
  <headword>beat <placeholderMarker>sb.</placeholderMarker>
up</headword>
  >
  <sense id="beat-up-1">
    <headwordTranslation>langCode="de">
      <text><placeholderMarker>jemanden</placeholderMarker>
verprügeln</
text>
    </headwordTranslation>
  </sense>
</entry>

```

### Example A.90. JSON

```

{
  "id": "beat-up",
  "headword": "beat sb. up",
  "placeholderMarkers": [
    {"startIndex": 5, "endIndex": 8}
  ],
  "senses": [{
    "id": "beat-up-1",
    "headwordTranslations": [{
      "text": "jemanden verprügeln",
      "langCode": "de",
      "placeholderMarkers": [
        {"startIndex": 0, "endIndex": 8}
      ]
    }
  ]
}]
}

```

### Example A.91. RDF

```
ex:beat-up-1 a dmlex:Sense;
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
    dmlex:placeholderMarker [
      dmlex:startIndex "0"^^xsd:unsignedInt;
      dmlex:endIndex "8"^^xsd:unsignedInt ];
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:langCode "de";
    dmlex:text "jemanden verprügeln" ];
  dmlex:listingOrder "1"^^xsd:unsignedInt.
ex:beat-up a dmlex:Entry;
  dmlex:sense ex:beat-up-1;
  dmlex:headword "beat sb. up";
  dmlex:placeholderMarker [
    dmlex:startIndex "5"^^xsd:unsignedInt;
    dmlex:endIndex "8"^^xsd:unsignedInt ].
```

### A.1.22 Using `headwordMarker`

#### Example A.92. Pseudocode

```
- entry (id: autopsy, headword: autopsy)
  - sense (id: autopsy-1)
    - headwordTranslation (text: pitva)
    - example (text: The coroner performed an autopsy.,
      headwordMarker: "autopsy")
      - exampleTranslation (text: Koroner provedl pitvu.,
        headwordMarker: "pitvu")
```

#### Example A.93. XML

```
<entry id="autopsy">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <headword>autopsy</headword>
  <sense id="autopsy-1">
<headwordTranslation><text>pitva</text></headwordTranslation>
    <example>
      <text>The coroner performed an
<headwordMarker>autopsy</headwo
rdMarker>.</text>
      <exampleTranslation> langCode="cs">
        <text>Koroner provedl
<headwordMarker>pitvu</headwordMarke
r>.</text>
      </exampleTranslation>
    </example>
    <headwordTranslation
langCode="cs"><text>pitva</text></headwordTra
nslation>
```

```
</sense>
</entry>
```

#### Example A.94. JSON

```
{
  "id": "autopsy",
  "headword": "autopsy",
  "senses": [{
    "id": "autopsy-1",
    "headwordTranslations": [{"text": "pitva"}], [{"
      "text": "pitva",
      "langCode": "cs"
    }],
    "examples": [{
      "text": "The coroner performed an autopsy.",
      "headwordMarkers": [
        {"startIndex": 25, "endIndex": 32}
      ],
      "exampleTranslations": [{
        "text": "Koroner provedl pitvu.",
        "langCode": "cs",
        "headwordMarkers": [
          {"startIndex": 16, "endIndex": 21}
        ]
      }
    ]
  ]
}]
}
```

#### Example A.95. RDF

```
ex:autopsy-1 a dmlex:Sense;
  dmlex:listingOrder "1"^^xsd:unsignedInt;
  dmlex:example [ a dmlex:Example;
    dmlex:exampleTranslation [ a dmlex:ExampleTranslation;
      dmlex:text "Koroner provedl pitvu.";
      dmlex:langCode "cs";
      dmlex:headwordMarker [
        dmlex:endIndex "21"^^xsd:unsignedInt;
        dmlex:startIndex "16"^^xsd:unsignedInt ];
      dmlex:listingOrder "1"^^xsd:unsignedInt ];
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:text "The coroner performed an autopsy.";
    dmlex:headwordMarker [
      dmlex:endIndex "32"^^xsd:unsignedInt;
      dmlex:startIndex "25"^^xsd:unsignedInt ] ];
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:langCode "cs";
    dmlex:text "pitva" ].
ex:autopsy a dmlex:Entry;
```

```
dmlex:sense ex:autopsy-1;
dmlex:headword "autopsy".
```

### A.1.23 Using `collocateMarker`

#### Example A.96. Pseudocode

```
- entry (id: autopsy, headword: autopsy)
  - sense id: autopsy-1)
  headwordTranslation (text: pitva)
    - example (text: The coroner performed an autopsy.)
      - headwordMarker: "autopsy"
      - collocateMarker: "performed" (lemma: perform)
      - exampleTranslation (text: Koroner provedl pitvu.)
        - headwordMarker: "pitvu"
        - collocateMarker: "provedl" (lemma: provést)
    headwordTranslation (text: pitva)
```

#### Example A.97. XML

```
<entry id="autopsy">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <headword>autopsy</headword>
  <sense id="autopsy-1">
    <headwordTranslation><text>pitva</text></headwordTranslation>
    <example>
      <text>The coroner <collocateMarker
lemma="perform">performed</
collocateMarker> an <headwordMarker>autopsy</headwordMarker>.</text>
      <exampleTranslation <langCode="cs">
        <text>Koroner <collocateMarker
lemma="provést">provedl</co
llocateMarker> <headwordMarker>pitvu</headwordMarker>.</text>
      </exampleTranslation>
    </example>
    <headwordTranslation
langCode="cs"><text>pitva</text></headwordTra
nslation>
  </sense>
</entry>
```

#### Example A.98. JSON

```
{
  "id": "autopsy",
  "headword": "autopsy",
  "senses": [{
    "id": "autopsy-1",
    "headwordTranslations": [{"langCode": "cs", "text": "pitva"}],
    "examples": [{
      "text": "The coroner performed an autopsy.",
```

```

"headwordMarkers": [
  {"startIndex": 25, "endIndex": 32}
],
"collocateMarkers": [
  {"startIndex": 12, "endIndex": 21, "lemma": "perform"}
],
"exampleTranslations": [{
  "langCode": "cs",
  "text": "Koroner provedl pitvu.",
  "headwordMarkers": [
    {"startIndex": 16, "endIndex": 21}
  ],
  "collocateMarkers": [
    {"startIndex": 8, "endIndex": 15, "lemma": "provést"}
  ]
}]
}]
}

```

*Example A.99. RDF*

```

ex:autopsy a dmlex:Entry;
  dmlex:sense ex:autopsy-1;
  dmlex:headword "autopsy".
ex:autopsy-1 a dmlex:Sense;
  dmlex:headwordTranslation [ a dmlex:HeadwordTranslation;
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:text "pitva" ];
  dmlex:listingOrder "1"^^xsd:unsignedInt;
  dmlex:example [ a dmlex:Example;
    dmlex:headwordMarker [
      dmlex:startIndex "25"^^xsd:unsignedInt;
      dmlex:endIndex "32"^^xsd:unsignedInt ];
    dmlex:text "The coroner performed an autopsy.";
    dmlex:listingOrder "1"^^xsd:unsignedInt;
    dmlex:exampleTranslation [ a dmlex:ExampleTranslation;
      dmlex:langCode "cs" ;
      dmlex:collocateMarker [
        dmlex:endIndex "15"^^xsd:unsignedInt;
        dmlex:startIndex "8"^^xsd:unsignedInt;
        dmlex:lemma "provést" ];
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:headwordMarker [
        dmlex:endIndex "21"^^xsd:unsignedInt;
        dmlex:startIndex "16"^^xsd:unsignedInt ];
      dmlex:text "Koroner provedl pitvu." ];
    dmlex:collocateMarker [
      dmlex:endIndex "21"^^xsd:unsignedInt;
      dmlex:lemma "perform";
      dmlex:startIndex "12"^^xsd:unsignedInt ] ].

```

## A.1.24 Modelling etymology



#### Example A.100. Pseudocode

```
- entry (id: cat-n, headword: cat)
  - sense ...
  - etymology
    - etymon
      - etymonUnit (langCode: enm, text: catte)
    - etymon
      - etymonUnit (langCode: ang, text: catt,
translation: male cat)
      - etymonUnit (langCode: ang, text: catte,
translation: female cat)
    - etymon
      - etymonUnit (langCode: gem-x-proto, text: kattuz,
reconstructed: true)
```

#### Example A.101. XML

```
<entry id="cat-n">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0">
  <headword>cat</headword>
  <etymology>
    <etymon>
      <etymonUnit langCode="enm">
        <text>catte</text>
      </etymonUnit>
    </etymon>
    <etymon>
      <etymonUnit langCode="ang">
        <text>catt</text>
        <translation>male cat</translation>
      </etymonUnit>
      <etymonUnit langCode="ang">
        <text>catte</text>
        <translation>female cat</translation>
      </etymonUnit>
    </etymon>
    <etymon>
      <etymonUnit langCode="gem-x-proto" reconstructed="true">
        <text>kattuz</text>
      </etymonUnit>
    </etymon>
  </etymology>
</entry>
```

#### Example A.102. JSON

```
{
  "id": "cat-n",
  "headword": "cat",
  "senses": [],
  "etymologies": [{
```

```

"etymons": [{
  "etymonUnits": [
    {"langCode": "enm", "text": "catte"}
  ]
}, {
  "etymonUnits": [
    {"langCode": "ang", "text": "catt", "translation": "male cat"},
    {"langCode": "ang", "text": "catte", "translation": "female
cat"}
  ]
}, {
  "etymonUnits": [
    {"langCode": "gem-x-proto", "text": "kattuz",
"reconstructed": true}
  ]
}]
}]
}

```

*Example A.103. RDF*

```

ex:cat-n a dmlex:Entry;
  dmlex:etymology [ a dmlex:Etymology;
    dmlex:etymon [ a dmlex:Etymon;
      dmlex:listingOrder "1"^^xsd:unsignedInt;
      dmlex:etymonUnit [ a dmlex:EtymonUnit;
        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "catte";
        dmlex:langCode "enm" ] ],
    [ a dmlex:Etymon;
      dmlex:listingOrder "3"^^xsd:unsignedInt;
      dmlex:etymonUnit [ a dmlex:EtymonUnit;
        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:text "kattuz";
        dmlex:reconstructed true;
        dmlex:langCode "gem-x-proto" ] ],
    [ a dmlex:Etymon;
      dmlex:listingOrder "2"^^xsd:unsignedInt;
      dmlex:etymonUnit [ a dmlex:EtymonUnit;
        dmlex:listingOrder "2"^^xsd:unsignedInt;
        dmlex:text "catte";
        dmlex:translation "female cat";
        dmlex:langCode "ang" ],
      [ a dmlex:EtymonUnit;
        dmlex:text "catt";
        dmlex:langCode "ang";
        dmlex:translation "male cat";
        dmlex:listingOrder "1"^^xsd:unsignedInt ] ];
    dmlex:listingOrder "1"^^xsd:unsignedInt ];
  dmlex:headword "cat".

```

## A.1.25 Modelling etymology types and language

*Example A.104. Pseudocode*

```

- lexicographicResource (langCode: en)
  - entry (id: school-noun-2, headword: school)
    - etymology (description: Middle Dutch scole ("multitude,
      troop, crowd, school") from Proto-Germanic
      *skulō; doublet of shoal)
    - etymon (type: derivation)
      - etymonUnit (langCode: dut, text: scole,
        translation: multitude, troop, crowd, school)
    - etymon (type: derivation)
      - etymonUnit (langCode: gem-pro, text: skulō,
        reconstructed: true)
    - etymon (type: doublet)
      - etymonUnit (langCode: en, text: shoal)
  - etymonLanguage (langCode: dum, displayName: Middle Dutch)
  - etymonLanguage (langCode: gem-pro, displayName: Proto-Germanic)
  - etymonType (name: derivation,
    description: A historical form of the word)
  - etymonType (name: doublet,
    description: Two words derived from the same etymological root)

```

**Example A.105. XML**

```

<lexicographicResource langCode="en">
  xmlns="http://docs.oasis-open.org/lexidma/ns/dmlex-1.0"
  <entry id="school-noun-2">
    <headword>school</headword>
    <etymology>
      <description>Middle Dutch scole ("multitude, troop, crowd,
sch
ool") from Proto-Germanic *skulō; doublet of shoal</description>
      <etymon type="derivation">
        <etymonUnit langCode="dum">
          <text>scole</text>
          <translation>multitude, troop, crowd,
school</translat
ion>
        </etymonUnit>
      </etymon>
      <etymon type="derivation">
        <etymonUnit langCode="gem-pro" reconstructed="true">
          <text>skulō</text>
        </etymonUnit>
      </etymon>
      <etymon type="doublet">
        <etymonUnit langCode="en">
          <text>shoal</text>
        </etymonUnit>
      </etymon>
    </etymology>
  </entry>
  <etymonLanguage langCode="dum">
    <displayName>Middle Dutch</displayName>
  </etymonLanguage>
  <etymonLanguage langCode="gem-pro">

```

```

      <displayName>Proto-Germanic</displayName>
    </etymonLanguage>
    <etymonType type="derivation">
      <description>A historical form of the word</description>
    </etymonType>
    <etymonType type="doublet">
      <description>Two words derived from the same etymological
root</de
scription>
    </etymonType>
<etymonLanguage langCode="dum">
<displayName>Middle Dutch</displayName>
</etymonLanguage>
<etymonLanguage langCode="gem-pro">
<displayName>Proto-Germanic</displayName>
</etymonLanguage>
</lexicographicResource>

```

**Example A.106. JSON**

```

{
  "langCode": "en",
  "entries": [ {
    "id": "school-noun-2",
    "headword": "school",
    "etymologies": [ {
      "description": "Middle Dutch scole (\"multitude, troop,
crowd, s
chool\") from Proto-Germanic *skulō; doublet of shoal",
      "etymons": [ {
        "type": "derivation",
        "etymonUnits": [ {
          "langCode": "dum",
          "text": "scole",
          "translation": "multitude, troop, crowd, school"
        }
      ]
    },
    {
      "type": "derivation",
      "etymonUnits": [ {
        "langCode": "gem-pro",
        "text": "skulō",
        "reconstructed": true
      }
    ]
  }
],
  "etymonLanguages": [ {
    "langCode": "dum",

```

```

    "displayName": "Middle Dutch"
  }, {
    "langCode": "gem-pro",
    "displayName": "Proto-Germanic"
  }
],
"etymonTypes": [ {
  "type": "derivation",
  "description": "A historical form of the word"
}, {
  "type": "doublet",
  "description": "Two words derived from the same etymological
root"
}
]
}

```

*Example A.107. RDF*

```

ex:lexicon a dmlex:LexicographicResource;
  dmlex:etymonLanguage [ a dmlex:EtymonLanguage;
    dmlex:langCode "dum";
    dmlex:displayName "Middle Dutch" ],
  [ a dmlex:EtymonLanguage;
    dmlex:langCode "gem-pro";
    dmlex:displayName "Proto-Germanic" ];
  dmlex:langCode "en";
  dmlex:entry ex:school-noun-2;
  dmlex:etymonType [ a dmlex:EtymonType;
    dmlex:description "Two words derived from the same etymological
root
";
    dmlex:type "doublet" ],
  [ a dmlex:EtymonType;
    dmlex:description "A historical form of the word";
    dmlex:type "derivation" ].
ex:school-noun-2 a dmlex:Entry;
  dmlex:headword "school";
  dmlex:etymology [ a dmlex:Etymology;
    dmlex:description "Middle Dutch scole (\"multitude, troop, crowd,
sc
hool\") from Proto-Germanic *skulō; doublet of shoal";
    dmlex:etymon [ a dmlex:Etymon;
      dmlex:type "derivation";
      dmlex:etymonUnit [ a dmlex:EtymonUnit;
        dmlex:text "skulō";
        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:reconstructed true;
        dmlex:langCode "gem-pro" ];
      dmlex:listingOrder "2"^^xsd:unsignedInt ],
      [ a dmlex:Etymon;
        dmlex:listingOrder "1"^^xsd:unsignedInt;
        dmlex:etymonUnit [ a dmlex:EtymonUnit;
          dmlex:langCode "dum";
          dmlex:text "scole";
          dmlex:translation "multitude, troop, crowd, school";
          dmlex:listingOrder "1"^^xsd:unsignedInt ];

```

```

    dmlex:type "derivation" ],
  [ a dmlex:Etymon;
    dmlex:etymonUnit [ a dmlex:EtymonUnit;
      dmlex:langCode "en";
      dmlex:text "shoal";
      dmlex:listingOrder "1"^^xsd:unsignedInt ];
    dmlex:type "doublet";
    dmlex:listingOrder "3"^^xsd:unsignedInt ];
  dmlex:listingOrder "1"^^xsd:unsignedInt ].

```

## A.2 DMLex NVH serialization (Informative)

This section specifies a serialization of the DMLex data model in NVH (Name-Value Hierarchy) [\[NVH\]](#).

For data exchange purposes, a valid DMLex NVH file is one where either a `lexicographicResource` or an `entry` is the top-level node or the file contains a concatenation of one (and only one) of these two nodes separated by a system-specific newline delimiter.

### A.2.1 Design Principles (Informative)

The NVH serialization of DMLex shown in this document follows these principles:

- The value of the node is the text property of the object, if defined, otherwise it is one of the required object properties of maximum arity 1.
- All other objects properties are implemented as NVH nodes.

See [namevaluehierarchy.org](http://namevaluehierarchy.org) for details about the NVH file format and processing libraries.

### A.2.2 DMLex NVH serialization nodes (Normative)

#### A.2.2.1 NVH node: *lexicographicResource*

Implements the [lexicographicResource](#) object type.

##### Value

- `title` OPTIONAL

##### Child nodes

- `entry` OPTIONAL (zero or ~~one~~ **more**)
- `uri` ~~REQUIRED (exactly~~ **OPTIONAL (zero or one)**
- `langCode` REQUIRED (exactly one)

##### Child nodes if implementing the Crosslingual Module

- `translationLanguage` REQUIRED (one or more)

##### Child nodes if implementing the Controlled Values Module

- `definitionTypeTag` OPTIONAL (zero or more)

- [inflectedFormTag](#) OPTIONAL (zero or more)
- [labelTag](#) OPTIONAL (zero or more)
- [labelTypeTag](#) OPTIONAL (zero or more)
- [partOfSpeechTag](#) OPTIONAL (zero or more)
- [sourceIdentityTag](#) OPTIONAL (zero or more)
- [transcriptionSchemeTag](#) **OPTIONAL (zero or more)**

*Child nodes if implementing the Linking Module*

- [relation](#) OPTIONAL (zero or more)
- [relationType](#) OPTIONAL (zero or more)

*Child nodes if implementing the Etymology Module*

- [etymonLanguage](#) OPTIONAL (zero or more)
- [etymonType](#) OPTIONAL (zero or more)

#### **A.2.2.2 NVH node: *entry***

Implements the [entry](#) object type.

*Value*

- headword REQUIRED

*Child nodes*

- homographNumber OPTIONAL (zero or one)
- [partOfSpeech](#) OPTIONAL (zero or more)
- [label](#) OPTIONAL (zero or more)
- [pronunciation](#) OPTIONAL (zero or more)
- [inflectedForm](#) OPTIONAL (zero or more)
- [sense](#) OPTIONAL (zero or more)

*Child nodes if implementing the Linking Module*

- id OPTIONAL (zero or one). Entries that have an ID ~~can be members~~ **may use this ID as object reference instead of the IRI addressing mechanism described in Section 3.2.1, “DMLex fragment identification strings”** ~~relations.~~

*Child nodes if implementing the Annotation Module*

- [placeholderMarker](#) OPTIONAL (zero or more)

*Child nodes if implementing the Etymology Module*

- [etymology](#) OPTIONAL (zero or more)

#### **A.2.2.3 NVH node: *partOfSpeech***

Implements the [partOfSpeech](#) object type.

*Value*

- tag REQUIRED

#### ***A.2.2.4 NVH node: [inflectedForm](#)***

Implements the [inflectedForm](#) object type.

*Value*

- text REQUIRED

*Child nodes*

- tag OPTIONAL (zero or one)
- [label](#) OPTIONAL (zero or more)
- [pronunciation](#) OPTIONAL (zero or more)

#### ***A.2.2.5 NVH node: [sense](#)***

Implements the [sense](#) object type.

*Value*

- indicator OPTIONAL (zero or one)

*Child nodes*

- [label](#) OPTIONAL (zero or more)
- [definition](#) OPTIONAL (zero or more)
- [example](#) OPTIONAL (zero or more)

*Child nodes if implementing the Linking Module*

- id OPTIONAL (zero or one). Senses that have an ID ~~can be members~~ [may use this ID as object reference instead of the IRI addressing mechanism described in Section 3.2.1, “DMLex fragment identification strings”](#) ~~relations.~~

*Child nodes if implementing the Crosslingual Module*

- [headwordExplanation](#) OPTIONAL (zero or more)
- [headwordTranslation](#) OPTIONAL (zero or more)

#### ***A.2.2.6 NVH node: [definition](#)***

Implements the [definition](#) object type.

*Value*



- REQUIRED, implements the `text` property.

*Child nodes*

- `definitionType` OPTIONAL (zero or one)

*Child nodes if implementing the Annotation Module*

- [headwordMarker](#) node OPTIONAL (zero or more)
- [collocateMarker](#) node OPTIONAL (zero or more)

#### **A.2.2.7 NVH node: *label***

Implements the [label](#) object type.

*Value*

- `tag` REQUIRED

#### **A.2.2.8 NVH node: *pronunciation***

Implements the [pronunciation](#) object type.

*Value*

- `soundFile` REQUIRED if `transcription` is present, OPTIONAL otherwise

*Child nodes*

- [transcription](#) OPTIONAL (zero or more) if `soundFile` is present, REQUIRED (one or more) otherwise
- [label](#) OPTIONAL (zero or more)

#### **A.2.2.9 NVH node: *transcription***

Implements the [transcription](#) object type.

*Value*

- REQUIRED, implements the `text` property.

*Child nodes*

- `scheme` OPTIONAL (zero or one)

#### **A.2.2.10 NVH node: *example***

Implements the [example](#) object type.

*Value*

- `text` REQUIRED

*Child nodes*

- `sourceIdentity` OPTIONAL (zero or one)
- `sourceElaboration` OPTIONAL (zero or one)
- `soundFile` OPTIONAL (zero or one)
- [label](#) OPTIONAL (zero or more)

*Child nodes if implementing the Annotation Module*

- [headwordMarker](#) OPTIONAL (zero or more)
- [collocateMarker](#) OPTIONAL (zero or more)

*Child nodes if implementing the Crosslingual Module*

- [exampleTranslation](#) OPTIONAL (zero or more)

**A.2.2.11 NVH node: *translationLanguage***

Implements the [translationLanguage](#) object type from the Crosslingual Module.

*Value*

- `langCode` REQUIRED

**A.2.2.12 NVH node: *headwordTranslation***

Implements the [headwordTranslation](#) object type from the Crosslingual Module.

*Value*

- REQUIRED, implements the `text` property.

*Child nodes*

- `langCode` OPTIONAL (zero or one) if the ancestor `lexicographicResource` contains exactly one `translationLanguage`, REQUIRED otherwise
- [partOfSpeech](#) OPTIONAL (zero or more)
- [label](#) OPTIONAL (zero or more)
- [pronunciation](#) OPTIONAL (zero or more)
- [inflectedForm](#) OPTIONAL (zero or more)

*Child nodes if implementing the Annotation Module*

- [placeholderMarker](#) OPTIONAL

**A.2.2.13 NVH node: *headwordExplanation***

Implements the [headwordExplanation](#) object type from the Crosslingual Module.

#### Value

- REQUIRED, implements the `text` property.

#### Child nodes

- `langCode` OPTIONAL (~~ZERO OR ONE~~ zero or one) if the ancestor `lexicographicResource` contains exactly one `translationLanguage`, REQUIRED otherwise

#### A.2.2.14 NVH node: *exampleTranslation*

Implements the [exampleTranslation](#) object type from the Crosslingual Module.

#### Value

- `text` REQUIRED

#### Child nodes

- `langCode` OPTIONAL (zero or one)
- `soundFile` OPTIONAL (zero or one) if the ancestor `lexicographicResource` contains exactly one `translationLanguage`, REQUIRED otherwise
- [label](#) OPTIONAL (zero or more)

Child nodes if implementing the Annotation Module

- [headwordMarker](#) OPTIONAL (zero or more)
- [collocateMarker](#) OPTIONAL (zero or more)

#### A.2.2.15 NVH node: *definitionTypeTag*

Implements the [definitionTypeTag](#) object type from the Controlled Values Module.

#### Value

- [tag](#) REQUIRED

#### Child nodes

- [description](#) OPTIONAL (zero or one)
- [sameAs](#) OPTIONAL (zero or more)

#### ~~partOfSpeechTag~~

~~Implements the [partOfSpeechTag](#) object type from the Controlled Values Module.~~

#### ~~Value~~

- ~~• [tag](#) REQUIRED~~

#### *Child nodes*

- ~~description OPTIONAL (zero or one)~~
- ~~for OPTIONAL (zero or one)~~
- ~~sameAs OPTIONAL (zero or more)~~

#### **A.2.2.16 NVH node: *inflectedFormTag***

Implements the [inflectedFormTag](#) object type from the Controlled Values Module.

#### *Value*

- tag REQUIRED

#### *Child nodes*

- description OPTIONAL (zero or one)
- for OPTIONAL (zero or one)
- [sameAs](#) OPTIONAL (zero or more)

#### **A.2.2.17 NVH node: ~~definitionTypeTag~~**

~~Implements the [definitionTypeTag](#) object type from the Controlled Values Module.~~

#### ~~*Value*~~

- ~~• tag REQUIRED~~

#### ~~*Child nodes*~~

- ~~• description OPTIONAL (zero or one)~~
- ~~• [sameAs](#) OPTIONAL (zero or more)~~

#### **A.2.2.18 NVH node: *labelTag***

Implements the [labelTag](#) object type from the Controlled Values Module.

#### *Value*

- tag REQUIRED

#### *Child nodes*

- description OPTIONAL (zero or one)
- typeTag OPTIONAL (zero or one)
- for OPTIONAL (zero or one)
- [sameAs](#) OPTIONAL (zero or more)

#### **A.2.2.19~~18~~ NVH node: *labelTypeTag***

Implements the [labelTypeTag](#) object type from the Controlled Values Module.

*Value*

- tag REQUIRED

*Child nodes*

- description OPTIONAL (zero or one)
- [sameAs](#) OPTIONAL (zero or more)

***A.2.2.19 NVH node: [partOfSpeechTag](#)***

Implements the [partOfSpeechTag](#) object type from the Controlled Values Module.

*Value*

- tag REQUIRED

*Child nodes*

- description OPTIONAL (zero or one)
- for OPTIONAL (zero or one)
- sameAs OPTIONAL (zero or more)

***A.2.2.20 NVH node: [sourceIdentityTag](#)***

Implements the [sourceIdentityTag](#) object type from the Controlled Values Module.

*Value*

- tag REQUIRED

*Child nodes*

- description OPTIONAL (zero or one)
- [sameAs](#) OPTIONAL (zero or more)

***A.2.2.21 NVH node: [transcriptionSchemeTag](#)***

Implements the [transcriptionSchemeTag](#) object type from the Controlled Values Module.

*Value*

- tag REQUIRED

*Child nodes*

- description OPTIONAL (zero or one)
- for OPTIONAL (zero or one)

#### ***A.2.2.22 NVH node: [sameAs](#)***

Implements the [sameAs](#) object type from the Controlled Values Module.

*Value*

- `uri` REQUIRED

#### ***A.2.2.23 NVH node: [relation](#)***

Implements the [relation](#) object type from the Linking Module.

*Value*

- `type` REQUIRED

*Child nodes*

- `description` OPTIONAL (zero or one)
- [member](#) REQUIRED (two or more)

#### ***A.2.2.24 NVH node: [member](#)***

Implements the [member](#) object type from the Linking Module.

*Value*

- `ref` REQUIRED: the `id` of an entry, a sense or a `collocateMarker`

*Child nodes*

- `role` OPTIONAL (zero or one)
- `obverseListingOrder` OPTIONAL (exactly one)

#### ***A.2.2.25 NVH node: [relationType](#)***

Implements the [relationType](#) object type from the Linking Module.

*Value*

- `type` REQUIRED

*Child nodes*

- `scopeRestriction` REQUIRED (exactly one)
- `description` OPTIONAL (zero or one)
- [memberType](#) OPTIONAL (zero or more)
- [sameAs](#) OPTIONAL (zero or more)

#### ***A.2.2.26 NVH node: [memberType](#)***

Implements the [memberType](#) object type from the Linking Module.

*Value*

- `type` REQUIRED

*Child nodes*

- `role` OPTIONAL (exactly one)
- `min` OPTIONAL (zero or one)
- `max` OPTIONAL (zero or one)
- `hint` OPTIONAL (zero or one)
- `description` OPTIONAL (zero or one)
- [sameAs](#) OPTIONAL (zero or more)

#### ***A.2.2.27 NVH node: [placeholderMarker](#)***

Implements the [placeholderMarker](#) object type from the Annotation module. Contains the text which it marks up.

*Value*

- REQUIRED, implements the `text` property.

#### ***A.2.2.28 NVH node: [headwordMarker](#)***

Implements the [headwordMarker](#) object type from the Annotation module. Contains the text which it marks up.

*Value*

- REQUIRED, implements the `text` property.

#### ***A.2.2.29 NVH node: [collocateMarker](#)***

Implements the [collocateMarker](#) object type from the Annotation module. Contains the text which it marks up and optional child nodes.

*Value*

- REQUIRED, implements the `text` property.

*Child nodes*

- `lemma` OPTIONAL (zero or one)
- [label](#) OPTIONAL (zero or more)

*Child nodes if implementing the Linking Module*

- id OPTIONAL (zero or one). Collocate markers that have an ID ~~can appear~~ may use this ID as members object reference instead of the IRI addressing mechanism described in Section 3.2.1, "DMLex fragment identification strings" relations.

#### **A.2.2.30 NVH node: *etymology***

Implements the [etymology](#) object type from the Etymology Module.

##### *Value*

- description OPTIONAL (zero or one)

##### *Child nodes*

- [etymon](#) OPTIONAL (zero or more)

#### **A.2.2.31 NVH node: *etymon***

Implements the [etymon](#) object type from the Etymology Module.

##### *Child nodes*

- when OPTIONAL (zero or one)
- type OPTIONAL (zero or one)
- note OPTIONAL (zero or one)
- [etymonUnit](#) REQUIRED (one or more)
- translation OPTIONAL (zero or one)

#### **A.2.2.32 NVH node: *etymonUnit***

Implements the [etymonUnit](#) object type from the Etymology Module.

##### *Value*

- text REQUIRED

##### *Child nodes*

- langCode REQUIRED (exactly one)
- reconstructed OPTIONAL (zero or one)
- [partOfSpeech](#) OPTIONAL (zero or more)
- translation OPTIONAL (zero or one)

#### **A.2.2.33 NVH node: *etymonType***

Implements the [etymonType](#) object type from the Etymology Module.

##### *Value*

- type REQUIRED



Child nodes

- description OPTIONAL (zero or one)

#### A.2.2.34 NVH node: *etymonLanguage*

Implements the [etymonLanguage](#) object type from the Etymology Module.

Value

- langCode REQUIRED

Child nodes

- displayName OPTIONAL (zero or one)

## Appendix B References (Normative)

This appendix contains the normative and informative references that are used in this document. Normative references are specific (identified by date of publication and/or edition number or Version number) and Informative references are either specific or non-specific.

While any hyperlinks included in this appendix were valid at the time of publication, OASIS cannot guarantee their long-term validity.

### B.1 Normative references (Normative)

[BCP 14] is a concatenation of [RFC 2119] and [RFC 8174]

[RFC 3986] Tim Berners-Lee, Roy T. Fielding, Larry M Masinter *Uniform Resource Identifier (URI): Generic Syntax*, <https://datatracker.ietf.org/doc/rfc3986/> IETF (Internet Engineering Task Force) RFC 3986, January 2005.

[RFC 3987] Martin J. Dürst, Michel Suignard, *Internationalized Resource Identifiers (IRIs)*, <https://datatracker.ietf.org/doc/rfc3987/> IETF (Internet Engineering Task Force) RFC 3987, January 2005.

[RFC 2119] S. Bradner, *Key words for use in RFCs to Indicate Requirement Levels*, <https://www.ietf.org/rfc/rfc2119.txt> IETF (Internet Engineering Task Force) RFC 2119, March 1997.

[RFC 8174] B. Leiba, *Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words*, <https://www.ietf.org/rfc/rfc8174.txt> IETF (Internet Engineering Task Force) RFC 8174, May 2017.

[BCP 47] M. Davis, *Tags for Identifying Languages*, <http://tools.ietf.org/html/bcp47> IETF (Internet Engineering Task Force).

[ITS] David Filip, Shaun McCance, Dave Lewis, Christian Lieske, Arle Lommel, Jirka Kosek, Felix Sasaki, Yves Savourel *Internationalization Tag Set (ITS) Version 2.0*, <http://www.w3.org/TR/its20/> W3C Recommendation 29 October 2013.

[JSON] *The JavaScript Object Notation (JSON) Data Interchange Format*, <https://tools.ietf.org/html/rfc8259> IETF RFC 8259 December 2017.

[RFC 3987] M. Duerst and M. Suignard, *Internationalized Resource Identifiers (IRIs)*, <https://www.ietf.org/rfc/rfc3987.txt> IETF (Internet Engineering Task Force) RFC 3987, January 2005.

[RFC 7303] H. Thompson and C. Lilley, *XML Media Types*, <https://www.tools.ietf.org/html/rfc7303> IETF (Internet Engineering Task Force) RFC 7303, July 2014.

[Schematron] International Standards Organization, *ISO/IEC 19757-3, Information Technology - Document Schema Definition Languages (DSDL) - Part 3: Rule-Based Validation — Schematron (Second Edition)*, [http://standards.iso.org/ittf/PubliclyAvailableStandards/c055982\\_ISO\\_IEC\\_19757-3\\_2016.zip](http://standards.iso.org/ittf/PubliclyAvailableStandards/c055982_ISO_IEC_19757-3_2016.zip) ISO, January 15, 2016.

[SQL] International Standards Organization, *ISO/IEC 9075, Information technology – Database languages – SQL*, <https://www.iso.org/standard/76583.html> ISO, 2023.

[UAX #9] M. Davis, A. Lanin, A. Glass, *UNICODE BIDIRECTIONAL ALGORITHM*, <http://www.unicode.org/reports/tr9/tr9-35.html> Unicode Bidirectional Algorithm, May 18, 2016.

[UAX #15] M. Davis, K. Whistler, *UNICODE NORMALIZATION FORMS*, <http://www.unicode.org/reports/tr15/tr15-44.html> Unicode Normalization Forms, February 24, 2016.

[Unicode] The Unicode Consortium, *The Unicode Standard*, <http://www.unicode.org/versions/Unicode9.0.0/> Mountain View, CA: The Unicode Consortium, June 21, 2016.

[XLIFF 2.1] David Filip, Tom Comerford, Soroush Saadatfar, Felix Sasaki, and Yves Savourel, eds. *XLIFF Version 2.0*, <http://docs.oasis-open.org/xliff/xliff-core/v2.1/os/xliff-core-v2.1-os.html> OASIS Standard 13 February 2018

[XML] W3C, *Extensible Markup Language (XML) 1.0*, <http://www.w3.org/TR/xml/> (Fifth Edition) W3C Recommendation 26 November 2008.

[XML namespace] W3C, *Schema document for namespace* <http://www.w3.org/XML/1998/namespace> <http://www.w3.org/2001/xml.xsd> [<http://www.w3.org/2009/01/xml.xsd>], at <https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd03/schemas/informativeCopiesOf3rdPartySchemas/w3c/xml.xsd> in this distribution

[XML Catalogs] Norman Walsh, *XML Catalogs*, <https://www.oasis-open.org/committees/download.php/14809/xml-catalogs.html> OASIS Standard V1.1, 07 October 2005.

[XML Schema] W3C, *XML Schema*, refers to the two part standard comprising [[XML Schema Structures](#)] and [[XML Schema Datatypes](#)] (Second Editions) W3C Recommendations 28 October 2004.

[XML Schema Datatypes] W3C, *XML Schema Part 2: Datatypes*, <http://www.w3.org/TR/xmlschema-2/> (Second Edition) W3C Recommendation 28 October 2004.

[XML Schema Structures] W3C, *XML Schema Part 1: Structures*, <https://www.w3.org/TR/xmlschema-1/> (Second Edition) W3C Recommendation 28 October 2004.

[DCMI Metadata Terms] DCMI Usage Board, *DCMI Metadata Terms*, <https://www.dublincore.org/specifications/dublin-core/dcmi-terms/> Date issued: 2020-01-20.

## B.2 Informative references (Informative)

[NVH] *Name-Value Hierarchy*, <https://www.namevaluehierarchy.org/> .

[LDML] *Unicode Locale Data Markup Language* <http://unicode.org/reports/tr35/>

[UAX #29] M. Davis, *UNICODE TEXT SEGMENTATION*, <http://www.unicode.org/reports/tr29/> Unicode text Segmentation.

## Appendix C Machine Readable Validation Artifacts (**Informative**Normative)

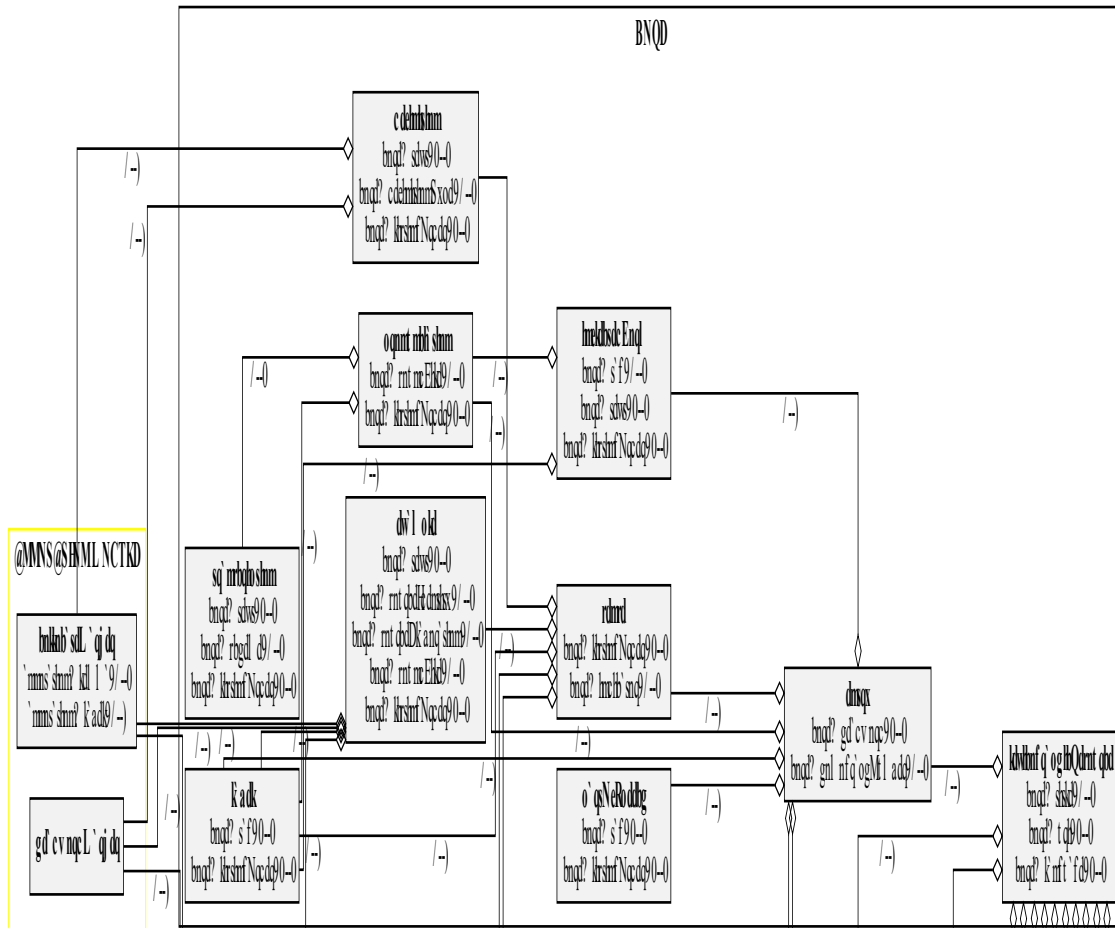
- [XML schema](#)
- [XML schema excluding the cross-lingual module](#)
- [JSON schema](#)
- [JSON schema excluding the cross-lingual module](#)
- [NVH schema \(informative\)](#)

~~CURRENTLY NO VALIDATION ARTIFACTS FORESEEN FOR THE OM... JUST FOR SERIALIZATIONS~~

~~MAY LIST CONFORMANT ARTIFACTS FOR SPECIFIC SERIALIZATIONS AT A LATER STAGE~~

## Appendix D DMLex UML diagram (Normative)

BNQD



@MNS@KMLNCTKD

BXBRKMF@KLNCTKD

BNSQKKDC U@KTRLNCTKD

hēhē' sēL' q' dē  
 'mns' shh' kēl' '9' /-  
 'mns' shh' k' adē /-

gē' ev nqēL' q' dē

o k' bēgē' dēL' q' dē

dwl o kē' q' rnt' shh  
 vhh' f' 'lē' s' hē' /-  
 vhh' f' 'lē' k' n' f' ' f' /-  
 vhh' f' 'lē' k' adē /-  
 vhh' f' 'lē' rnt' nē' hē' /-  
 vhh' f' 'lē' h' rsh' Nq' dē' /-

gē' ev nqē' D' w' k' m' shh  
 vhh' f' 'lē' s' hē' /-  
 vhh' f' 'lē' k' n' f' ' f' /-

gē' ev nqē' S' q' rnt' shh  
 vhh' f' 'lē' s' hē' /-  
 vhh' f' 'lē' k' n' f' ' f' /-  
 vhh' f' 'lē' h' rsh' Nq' dē' /-  
 vhh' f' 'lē' o' qē' Vē' dē' hē' /-  
 vhh' f' 'lē' k' adē /-  
 vhh' f' 'lē' o' qunt' nshh' /-  
 vhh' f' 'lē' hēhē' dē' Enq' /-

k' adē' f'  
 u' k' dē' s' f' /-  
 u' k' dē' e' dē' hē' dē' shh' /-  
 u' k' dē' s' o' dē' s' f' /-  
 u' k' dē' e' nq' /-  
 u' k' dē' r' l' dē' /-

k' adē' s' dē' f'  
 u' k' dē' s' f' /-  
 u' k' dē' e' dē' hē' dē' shh' /-  
 u' k' dē' r' l' dē' /-

o' qē' Vē' dē' hē' s' f'  
 u' k' dē' s' f' /-  
 u' k' dē' e' dē' hē' dē' shh' /-  
 u' k' dē' e' nq' /-  
 u' k' dē' r' l' dē' /-

rnt' qē' hē' dē' hē' s' f'  
 u' k' dē' s' f' /-

k' w' h' h' f' q' o' g' h' q' d' n' t' q' d'  
 b' n' a' d' /-  
 b' n' a' d' t' d' /-  
 b' n' a' d' k' n' f' f' /-



## Appendix E DMLex plain text description (Informative)

This description uses the [NVH schema notation style](#) and the convention that all object types and their properties are prefixed by either `core@` or `MODULENAME@` strings. Their arity is denoted by a `*` (zero or more), `+` (one or more), `?` (zero or one) or `N+` (`N` or more). If no arity is given, it indicates the arity of 1, i.e. being required and occurring exactly once.

```
core@lexicographicResource:
  core@title: ?
  core@uri: ?
  core@language:
  core@entry: *
  xlingual@translationLanguage: +
  values@definitionTypeTag: *
  values@inflectedFormTag: *
  values@labelTag: *
  values@labelTypeTag: *
  values@partOfSpeechTag: *
  values@sourceIdentityTag: *
  values@transcriptionSchemeTag: *
  linking@relation: *
  linking@relationType: *
  etymology@etymonLanguage: *
  etymology@etymonType: *
core@entry:
  core@headword:
  core@homographNumber: ?
  core@partOfSpeech: *
  core@label: *
  core@pronunciation: *
  core@inflectedForm: *
  core@sense: *
  annotation@placeholderMarker: *
  etymology@etymology: *
core@partOfSpeech:
  core@tag:
  core@listingOrder:
core@inflectedForm:
  core@tag: ?
  core@text:
  core@label: *
  core@pronunciation: *
  core@listingOrder:
core@sense:
  core@listingOrder:
  core@indicator: ?
  core@label: *
  core@definition: *
  core@example: *
  xlingual@headwordExplanation: *
  xlingual@headwordTranslation: *
core@definition:
  core@text:
```

```
    core@definitionType: ?
    core@listingOrder:
    annotation@headwordMarker: *
    annotation@collocateMarker: *
core@label:
    core@tag:
    core@listingOrder:
core@pronunciation:
    core@soundFile: ?
    core@transcription: ?
    core@listingOrder:
    core@label: *
core@transcription:
    core@text:
    core@scheme: ?
    core@listingOrder:
core@example:
    core@text:
    core@sourceIdentity: ?
    core@sourceElaboration: ?
    core@label: *
    core@soundFile: ?
    core@listingOrder:
    xlingual@exampleTranslation: *
    annotation@headwordMarker: *
    annotation@collocateMarker: *
xlingual@translationLanguage:
    xlingual@langCode:
    xlingual@listingOrder:
xlingual@headwordExplanation:
    xlingual@text:
    xlingual@language: *
xlingual@headwordTranslation:
    xlingual@text:
    xlingual@language: *
    xlingual@listingOrder:
    xlingual@partOfSpeech: *
    xlingual@label: *
    xlingual@pronunciation: *
    xlingual@inflectedForm: *
    annotation@placeholderMarker: *
xlingual@exampleTranslation:
    xlingual@text:
    xlingual@language: *
    xlingual@label: *
    xlingual@soundFile: *
    xlingual@listingOrder:
    annotation@headwordMarker: *
    annotation@collocateMarker: *
values@definitionTypeTag:
    values@tag:
    values@description: ?
    values@sameAs: *
values@inflectedFormTag:
    values@tag:
    values@description: ?
    values@for: ?
```

```
    values@sameAs: *
values@labelTag:
  values@tag:
  values@description: ?
  values@typeTag: ?
  values@for: ?
  values@sameAs: *
values@labelTypeTag:
  values@tag:
  values@description: ?
  values@sameAs: *
values@partOfSpeechTag:
  values@tag:
  values@description: ?
  values@for: ?
  values@sameAs: *
values@sourceIdentityTag:
  values@tag:
  values@description: ?
  values@sameAs: *
values@transcriptionSchemeTag:
  values@tag:
  values@description: ?
  values@for: ?
linking@relation:
  linking@type:
  linking@description: ?
  linking@member: 2+
linking@member:
  linking@memberID:
  linking@role: ?
  linking@listingOrder:
  linking@obverseListingOrder:
linking@relationType:
  linking@type:
  linking@description: ?
  linking@scope: ? (any, sameEntry, sameResource)
  linking@memberType: *
  linking@sameAs: *
linking@memberType:
  linking@role:
  linking@description: ?
  linking@memberType: (sense, entry, collocate)
  linking@min: ?
  linking@max: ?
  linking@action: (embed, navigate, none)
  linking@sameAs: *
annotation@placeholderMarker:
annotation@headwordMarker:
annotation@collocateMarker:
  annotation@lemma: ?
  annotation@label: *
etymology@etymology:
  etymology@description: ?
  etymology@etymon: *
  etymology@listingOrder:
etymology@etymon:
```



```
etymology@when: ?
etymology@type: ?
etymology@note: ?
etymology@etymonUnit: +
etymology@translation: ?
etymology@listingOrder:
etymology@etymonUnit:
etymology@language:
etymology@text:
etymology@reconstructed: ?
etymology@partOfSpeech: *
etymology@translation: ?
etymology@listingOrder:
etymology@etymonLanguage:
etymology@langCode:
etymology@displayName: ?
etymology@etymonType:
etymology@type:
etymology@description: ?
```

## Appendix F Specification Change Tracking (Informative)

~~This appendix will contain tracked changes after the csd01 phase will have been reached.~~

### F.1 Tracking of changes made in response to Public Reviews

This is to facilitate human tracking of changes in the specification made since the first Public Review publication on 8th September 2023.

#### F.1.1 Tracking of changes in response to the ~~1st~~**2nd** Public Review

This section tracks major changes made to this specification compared to the Committee Specification Draft **02** <https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd02/dmlex-v1.0-csd02.pdf>. ~~The first Public Review~~ **The second Public Review** took place from 31st January 2024 until 29th February 2024.

- [1. A method for constructing fragment identification strings has been added to the specification \(GitHub issue 97\).](#)
- [2. Consistency has been introduced to the use of <text> child elements in the XML serialisation. \(GitHub issue 72\).](#)
- [3. Clarifications have been added on how whitespace should be handled, both at the level of the data model and in the XML serialisation \(GitHub issue 84\).](#)
- [4. Several validation artifacts have been added to the specification, including an XML schema and a JSON schema \(GitHub issue 93\).](#)
- [5. A clarification has been added on element ordering in the XML serialisation \(GitHub issue 117\).](#)
- [6. A clarification has been added to the XML serialisation on which elements can be the top-level elements in a DMLex-comformant XML document \(GitHub issue 118\).](#)

#### F.1.2 Tracking of changes in response to the **1st** Public Review

This section tracks major changes made to this specification compared to the Committee Specification Draft **01** <https://docs.oasis-open.org/lexidma/dmlex/v1.0/csd02/dmlex-v1.0-csd02.pdf>~~01~~. The initial Public Review took place from 22nd September 2023 until 17th November 2023.

1. Clarifications have been added to explain why DMLex allows lexicographic resources with zero entries and entries with zero senses, in response to a review comment (GitHub issue [52](#)).
2. The data type of `pronunciation.soundFile` has been changed to Internationalized Resource Identifier (IRI), in response to a review comment (GitHub issue [54](#)).
3. In the Controlled Values Module, the properties of various object types such as `forHeadwords`, `forPartOfSpeech` and so on, which had originally been intended to model various constraints on the use of tags, have been radically simplified into a single property called `for` – in response to a review comment (GitHub issue [55](#)).
4. In the Linking Module, uniqueness constraints on `relation` have been removed in response to a review comment (GitHub issue [56](#)).
5. The Etymology Module has been updated to disallow the existence of empty etymologies, in response to a review comment (GitHub issue [59](#)).
6. In the XML and JSON serializations (which are normative) and in the NVH serialization (which is informative), items which have `id` properties (those are: entries, senses, collocate markers) have been changed so that the `id` property is relevant only if implementing the Linking Module – in response to a review comment (GitHub issue [65](#)).
7. In the Linking Module, the `member_id` property of `member` has been changed to `ref` to avoid a potential misunderstanding about the purpose of this property, in response to a review comment (GitHub issue [66](#)).
8. In the Linking Module, the `obverseListingOrder` property of `member` has been made optional in response to a review comment (GitHub issue [67](#)).
9. In the Linking Module, the `role` property of `memberType` has been made optional in response to a review comment (GitHub issue [68](#)).
10. In the XML serialization, whitespace handling rules have been added, both at the default level for all elements, and specifically for elements which are allowed to contain inline markup, in response to a review comment (GitHub issue [69](#)).
11. Multiple changes have been made to the Examples section to correct inconsistencies between the examples and the rest of the specification, in response to various review comments (including GitHub issue [62](#)).

## Appendix G Acknowledgements (Informative)

The following individuals have participated in the creation of this specification and are gratefully acknowledged:

- Erjavec, Tomaž - Jožef Stefan Institute
- Filip, David - Masaryk University (previously TCD, ADAPT Centre)
- Jakubiček, Miloš - Lexical Computing
- Kallas, Jelena - Institute of the Estonian Language
- Kernerman, Ilan - K Dictionaries
- Kokol, Marko - Jožef Stefan Institute
- Kosem, Iztok - Jožef Stefan Institute
- Krek, Simon - Jožef Stefan Institute
- McCrae, John - University of Galway
- Měchura, Michal - Masaryk University
- Tiberius, Carole - Dutch Language Institute

## Appendix H Notices (Informative)

Copyright © OASIS Open 2024. All Rights Reserved.

All capitalized terms in the following text have the meanings assigned to them in the OASIS Intellectual Property Rights Policy (the "OASIS IPR Policy"). The full [Policy](#) may be found at the OASIS website.

This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published, and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this section are included on all such copies and derivative works. However, this document itself may not be modified in any way, including by removing the copyright notice or references to OASIS, except as needed for the purpose of developing any document or deliverable produced by an OASIS Technical Committee (in which case the rules applicable to copyrights, as set forth in the OASIS IPR Policy, must be followed) or as required to translate it into languages other than English.

The limited permissions granted above are perpetual and will not be revoked by OASIS or its successors or assigns.

This document and the information contained herein is provided on an "AS IS" basis and OASIS DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY OWNERSHIP RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

As stated in the OASIS IPR Policy, the following three paragraphs in brackets apply to OASIS Standards Final Deliverable documents (Committee Specification, OASIS Standard, or Approved Errata).

[OASIS requests that any OASIS Party or any other party that believes it has patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable, to notify OASIS TC Administrator and provide an indication of its willingness to grant patent licenses to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this deliverable.]

[OASIS invites any party to contact the OASIS TC Administrator if it is aware of a claim of ownership of any patent claims that would necessarily be infringed by implementations of this OASIS Standards Final Deliverable by a patent holder that is not willing to provide a license to such patent claims in a manner consistent with the IPR Mode of the OASIS Technical Committee that produced this OASIS Standards Final Deliverable. OASIS may include such claims on its website, but disclaims any obligation to do so.]

[OASIS takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this OASIS Standards Final Deliverable or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Information on OASIS' procedures with respect to rights in any document or deliverable produced by an OASIS Technical Committee can be found on the OASIS website. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this OASIS Standards Final Deliverable, can be obtained from the OASIS TC Administrator. OASIS makes no representation that any information or list of intellectual property rights will at any time be complete, or that any claims in such list are, in fact, Essential Claims.]

The name "OASIS" is a trademark of [OASIS](https://www.oasis-open.org), the owner and developer of this specification, and should be used only to refer to the organization and its official outputs. OASIS welcomes reference to, and implementation and use of, specifications, while reserving the right to enforce its marks against misleading uses. Please see <https://www.oasis-open.org/policies-guidelines/trademark/> for above guidance.