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 headwords, part-of-speech labels, senses, definitions and so on.

3.1 `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 database. Note that the correct name of this data type in DMLex is lexicographic, not lexical, resource.

Contents

- `id` REQUIRED (exactly one). An identifier of the lexicographic resource, unique among all lexicographic resources in the implementation. If the implementation contains only one lexicographic resource then `id` MAY be understood as implicit and MAY be left unimplemented.
- `title` OPTIONAL (zero or one). Non-empty string. A human-readable title of the lexicographic resource.
- `uri` OPTIONAL (zero or one). The URI of the lexicographic resource, identifying it on the Web.
- `language` REQUIRED (exactly one). The IETF language code of the language that this lexicographic resource describes.
- `entry` OPTIONAL (zero, one or more)
- `tag` OPTIONAL (zero, one or more)

Comments

- `language` 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 occur in the lexicographic resource, are in that language too. The `language` child object of `lexicographicResource` informs potential users of the lexicographic resource which language that is.
- The main role of a lexicographic resource is to contain entries (`entry` objects). The other object type that can optionally occur inside a `lexicographicResource`, `tag`, is for lists of look-up values such as part-of-speech labels.

Example 1. XML

```
<lexicographicResource id="..." uri="..." language="...">
    <title>...</title>
    <entry.../>
    <tag.../>
</lexicographicResource>
```

Example 2. JSON

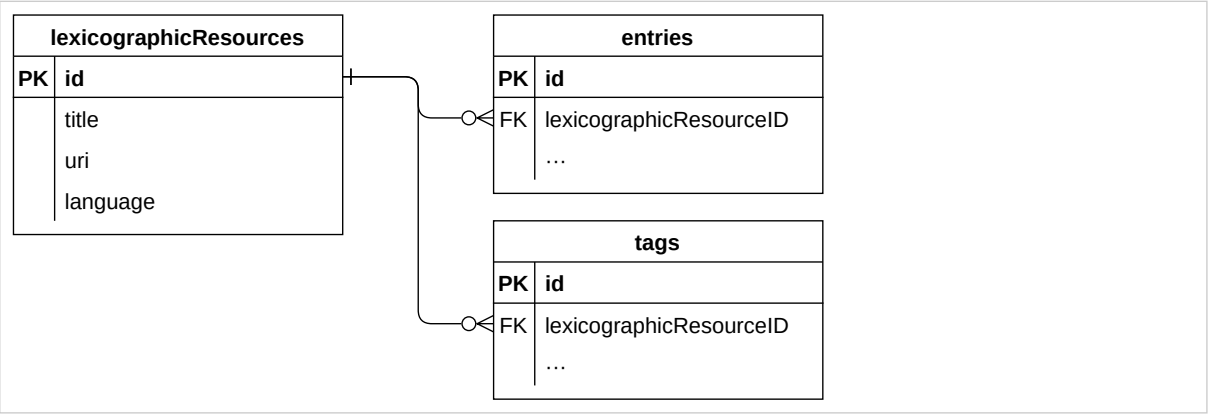
```
{
    "id": "...",
    "title": "...",
    "language": "...",
    "entries": [...],
    "tags": [...]
}
```

Example 3. RDF

```
@prefix dmlex: <http://www.oasis-open.org/to-be-confirmed/dmlex> .

<id> a dmlex:Lexicographic Resource ;
  dmlex:title "..." ;
  dmlex:uri "..." ;
  dmlex:language "..." ;
  dmlex:entry <entry1> , ... ;
  dmlex:tag ... .
```

Example 4. Relational database



3.2 entry

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

Child of

- `lexicographicResource`

Contents

- `id` REQUIRED (exactly one). An identifier of the entry, unique among all lexicographic resources in the implementation.
- `headword` REQUIRED (exactly one). Non-empty string. The entry's headword.
- `homographNumber` OPTIONAL (zero or one). The entry's homograph number, as a guide for humans to distinguish entries with the same headword.

- `partOfSpeech` OPTIONAL (zero, one or more).
- `label` OPTIONAL (zero, one or more).
- `pronunciation` OPTIONAL (zero, one or more).
- `inflectedForm` OPTIONAL (zero, one or more).
- `sense` OPTIONAL (zero, one or more).

Comments

- 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.
- 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.
- DMLex Core does not have a concept of "subentry". To model subentries (ie. entries inside entries) in a lexicographic resource, you should use object types from the Linking Module for that.

Example 5. XML

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

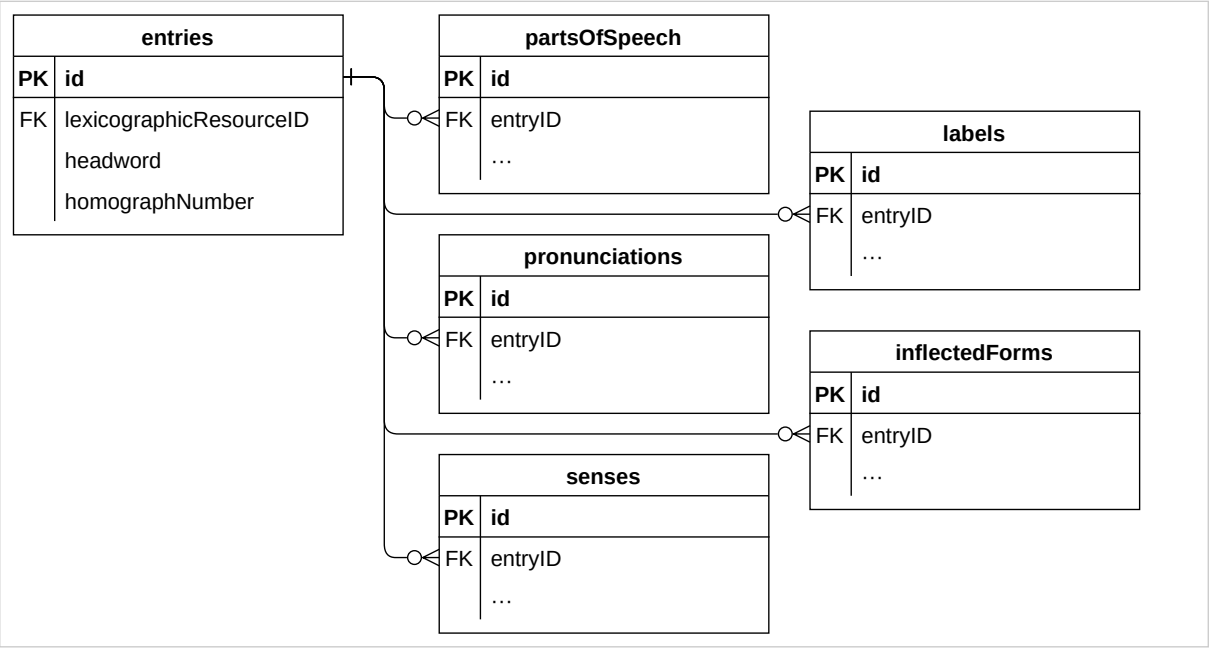
Example 6. JSON

```
{
    "id": "...",
    "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



3.3 `partOfSpeech`

Represents a part-of-speech label.

Child of

- `entry`

Contents

- `value` REQUIRED (exactly one). Non-empty 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 `tag` 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

- If you want to model other grammatical properties of the headword besides part of speech, such as gender (of nouns) or aspect (of verbs), the way to do that in DMLex is to conflate them to the part-of-speech label, for example `noun-masc` and `noun-fem`, or `v-perf` and `v-imperf`.

Example 9. XML

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

Example 10. JSON

```
"..."
```

Example 11. RDF

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

Example 12. Relational database

partsOfSpeech	
PK	**id**
FK	entryID
	value
	listingOrder

3.4 `inflectedForm`

Represents one (of possibly many) inflected forms of the headword. Example: Section 8.2, “How to use `inflectedForm`”.

Child of

- `entry`

Contents

- `inflectedTag` OPTIONAL (zero or one). Non-empty 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 `tag` 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.
- `text` REQUIRED (exactly one). Non-empty string. The text of the inflected form.

- `label` OPTIONAL (zero, one or more).
- `pronunciation` OPTIONAL (zero, one 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 inflectedTag="...">
    <text>...</text>
    <label.../>
    <pronunciation.../>
</inflectedTag>
```

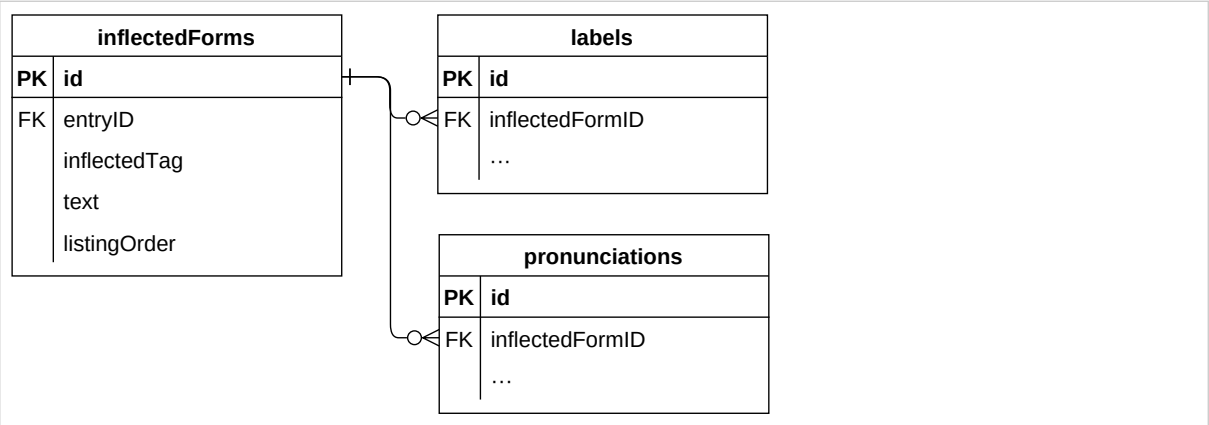
Example 14. JSON

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

Example 15. RDF

```
<entry> dmlex:inflectedForm [
  a dmlex:InflectedForm ;
  dmlex:inflectedTag "..." ;
  dmlex:listingOrder 1 ;
  dmlex:label ... ;
  dmlex:pronunciation ... .
```

Example 16. Relational database



Comments

- The `inflectedForm` object is intended to model the **inflectional morphology** of a headword. To model derivational morphology, for example feminine forms of maculine 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.

3.5 sense

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

Child of

- `entry`

Contents

- `id` REQUIRED (exactly one). An identifier of the entry, unique among all lexicographic resources in the implementation.
- `listingOrder` REQUIRED (exactly one). Number. The position of this sense among other senses of the same entry. Can be implicit from the serialization.
- `indicator` OPTIONAL (zero or one). 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, one or more).
- `definition` OPTIONAL (zero, one or more).
- `example` OPTIONAL (zero, one or more).

Comments

- An **entry** is a container for 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. If you have an entry where each sense has slightly different morphological properties (eg. a noun has a weak plural in one sense and a strong plural in another) then, in DMLex, you need to treat it as two entries (homographs), and you can use the Linking Module two link the two entries together and to make sure they are always shown together to human users.

Example 17. XML

```
<sense id="...">
    <indicator>...</indicator>
    <label.../>
    <definition.../>
    <example.../>
</sense>
```

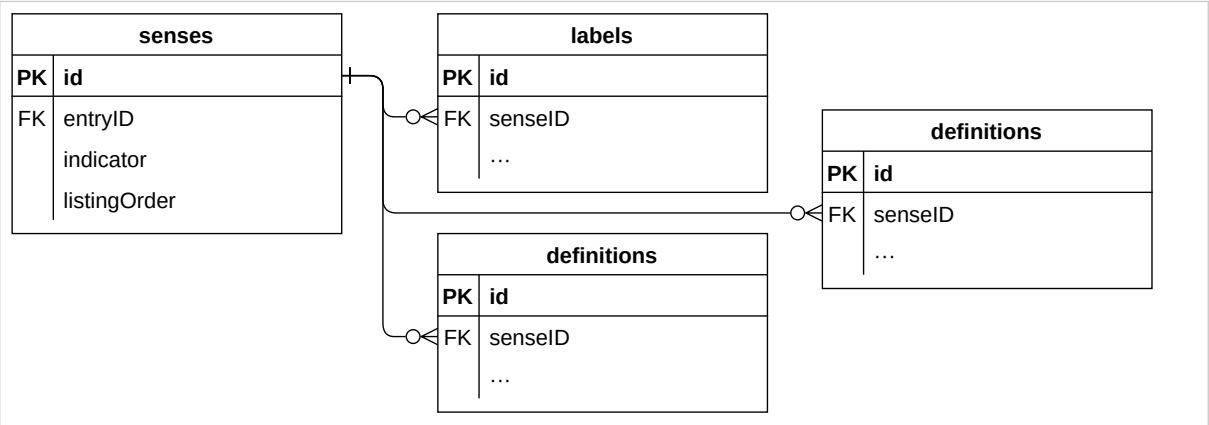
Example 18. JSON

```
{
    "id": "...",
    "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



3.6 definition

Represents one of possibly several definitions of a sense.

Child of

- sense

Contents

- value REQUIRED (exactly one). Non-empty 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). If a sense contains multiple definitions, indicates the difference between them, for example that they are intended for different audiences. The tag 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="...">...</definition>
```

Example 22. JSON

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

Example 23. RDF

```
<entry> dmlex:definition [
  a dmlex:Definition ;
  rdf:value "..." ;
  dmlex:definitionType "..." ;
  dmlex:listingOrder 1 ] .
```

Example 24. Relational database

definitions	
PK	**id**
FK	senseID
	text
	definitionType
	listingOrder

3.7 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).

Child of

- `entry`
- `sense`
- `inflectedForm`
- `pronunciation`
- `example`

Contents

- `value` REQUIRED (exactly one). Non-empty 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 `tag` 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 pronuciation of the headword (in all senses).

Example 25. XML

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

Example 26. JSON

```
"..."
```

Example 27. RDF

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

Example 28. Relational database

labels	
PK	**id**
FK	entryID
FK	senseID
FK	inflectedFormID
FK	pronunciationID
FK	exampleID
	value
	listingOrder

3.8 pronunciation

Represents the pronunciation of its parent. Examples: Section 8.3, “Pronunciation given as transcription”, Section 8.4, “Pronunciation given as a sound file”, Section 8.5, “Pronunciation given both ways”.

Child of

- `entry`
- `inflectedForm`

Contents

- `soundFile` OPTIONAL (zero or one). A pointer to a file, such as a filename or a URI, containing a sound recording of the pronunciation
- `transcription` OPTIONAL (zero, one or more).
- `listingOrder` REQUIRED (exactly one). Number. The position of this pronuciation object among other pronuciation objects of the same entry. This can be implicit from the serialization.
- `label` OPTIONAL (zero, one or more).

Example 29. XML

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

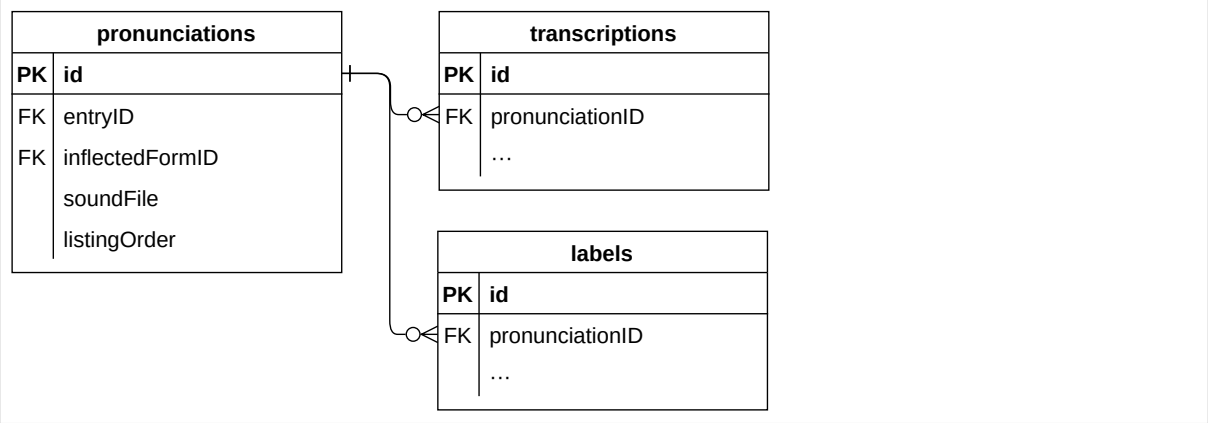
Example 30. JSON

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

Example 31. RDF

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

Example 32. Relational database



3.9 transcription

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

Child of

- `pronunciation`

Contents

- `text` REQUIRED (exactly one). Non-empty 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.
- `listingOrder` REQUIRED (exactly one). Number. The position of this transcription object among transcriptions of the same pronunciations. This can be implicit from the serialization.

Example 33. XML

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

Example 36. Relational database

transcriptions	
PK	**id**
FK	pronunciationID
	text
	scheme
	listingOrder

3.10 example

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

Child of

- sense

Contents

- `text` REQUIRED (exactly one). Non-empty string. The example itself.
- `sourceIdentity` OPTIONAL (zero or one). An abbreviation, a code or some other string of text which identifies the source. The `tag` 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 string. A free-form statement about the source of the example. If `source` is present, then `sourceElaboration` can be used for information 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, one or more).
- `soundFile` OPTIONAL (zero or one). A pointer to a file, such as a filename or a URI, containing 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>
```

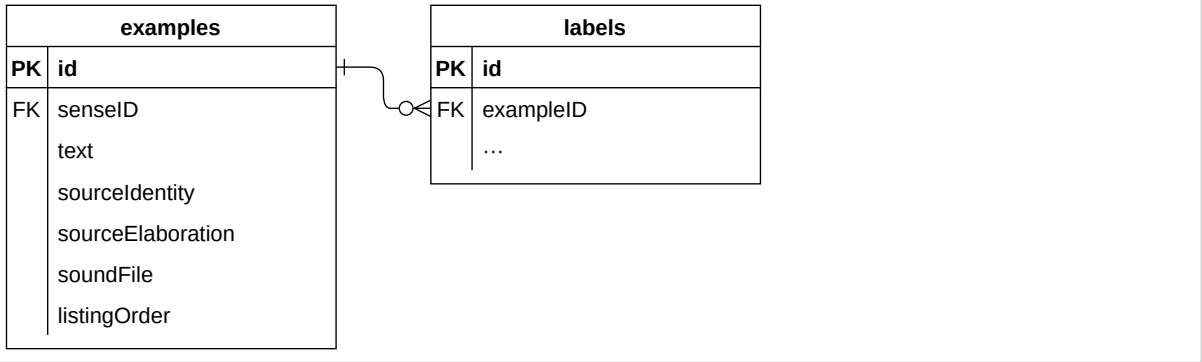
Example 38. JSON

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

Example 39. RDF

```
<entry> dmlex:example [
  a dmlex:Example ;
  rdf:value "..." ;
  dmlex:sourceIdentity "..." ;
  dmlex:sourceElaboration "..." ;
  dmlex:label ... ;
  dmlex:soundFile <...> ;
  dmlex:listingOrder 1 ] .
```

Example 40. Relational database



3.11 `tag`

Represents one (of many) possible values for `value` of `partOfSpeech`, `inflectedTag` of `inflectedForm`, `definitionType` of `definition`, `value` of `label` or `sourceIndentity` of `example`. Example: Section 8.6, “How to use `tag`”.

Child of

- `lexicographicResource`

Contents

- `value` REQUIRED (exactly one). Non-empty string. An abbreviation, a code or some other string of text.
- `description` OPTIONAL (zero or one). Non-empty string. A human-readable description of what the tag means.

- `target` OPTIONAL (zero, one or more). Indicates how the value of this tag can be used in this lexicographic resource. One of: `partOfSpeech`, `inflectedTag`, `definitionType`, `label`, `sourceIndentity`. If omitted, then all.

- `partOfSpeechConstraint` OPTIONAL (zero, one or more). If present, says that this tag is only intended to be used inside entries that are labelled with this part of speech. This can be used to constrain that, for example, only nouns and adjectives can have plurals but other parts of speech cannot.

- `sameAs` OPTIONAL (zero, one or more).

Comments

- In a lexicographic resource, `tag` objects can used to define controlled vocabularies (inventories of part-of-speech tags, labels etc.) and constrains on where they are expected to be used (e.g. part-of-speech tags are expected to be used as `value` of `partOfSpeech` objects).

  Enforcing these constraints in an implementation of DMLex, for example as business rules in a dictionary-writing system, is OPTIONAL: an implementation of DMLex is compliant regardless of whether it enforces the constraints defined by `tag` objects or not.

Example 41. XML

```
<tag value="...">
    <description>...</description>
    <target value="..."/>
    <partOfSpeechConstraint value="..."/>
    <sameAs.../>
</tag>
```

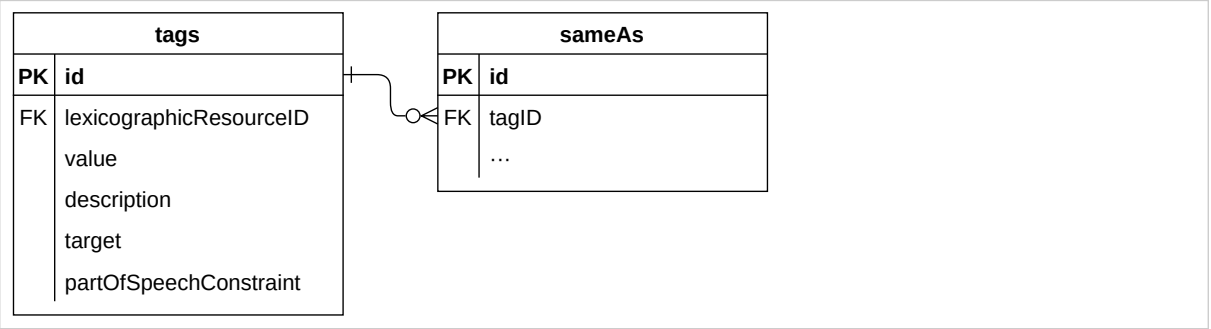
Example 42. JSON

```
{
    "value": "...",
    "description": "...",
    "targets": ["..."],
    "partOfSpeechConstraints": ["..."],
    "sameAs": [...]
}
```

Example 43. RDF

```
<entry> dmlex:tag [
  a dmlex:Tag ;
  dmlex:description "..." ;
  dmlex:target "..." ;
  dmlex:partOfSpeechConstraints "..." ;
  dmlex:sameAs ... ] .
```

Example 44. Relational database



3.12 sameAs

Represents the fact that the parent object is equivalent to an item available from an external authority. Example: Section 8.7, “Mapping `tag` to external inventories”.

Child of

- `tag`

Contents

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

Example 45. XML

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

Example 46. JSON

```
"..."
```

Example 47. Relational database