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

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

Example 2. JSON

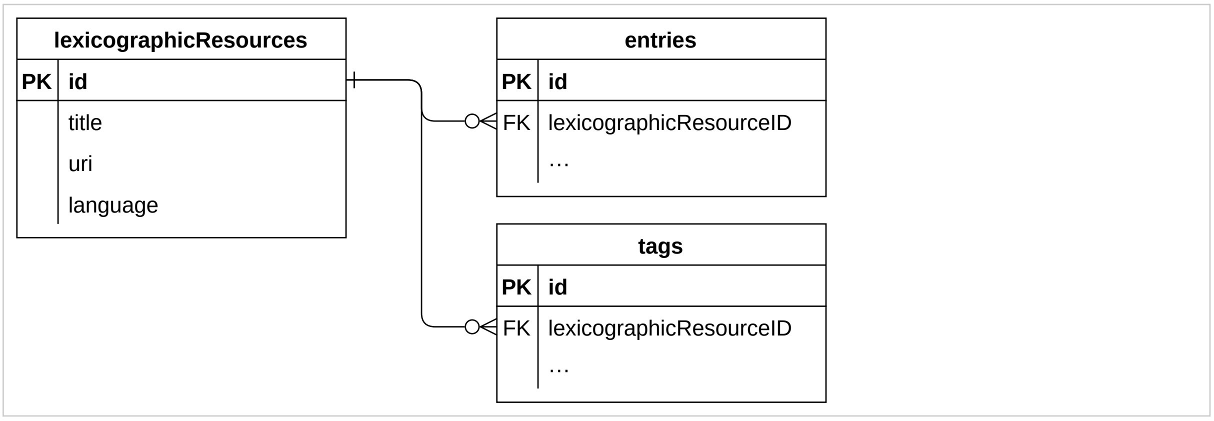
```
{
    "title": "...",
    "language": "...",
    "entries": [...],
    "tags": [...]
}
```

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` OPTIONAL (zero or one). An unique identifier of the entry. Entries which have idenfitiers are capable of being involved in relations created with the Linking module.
- `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>
```

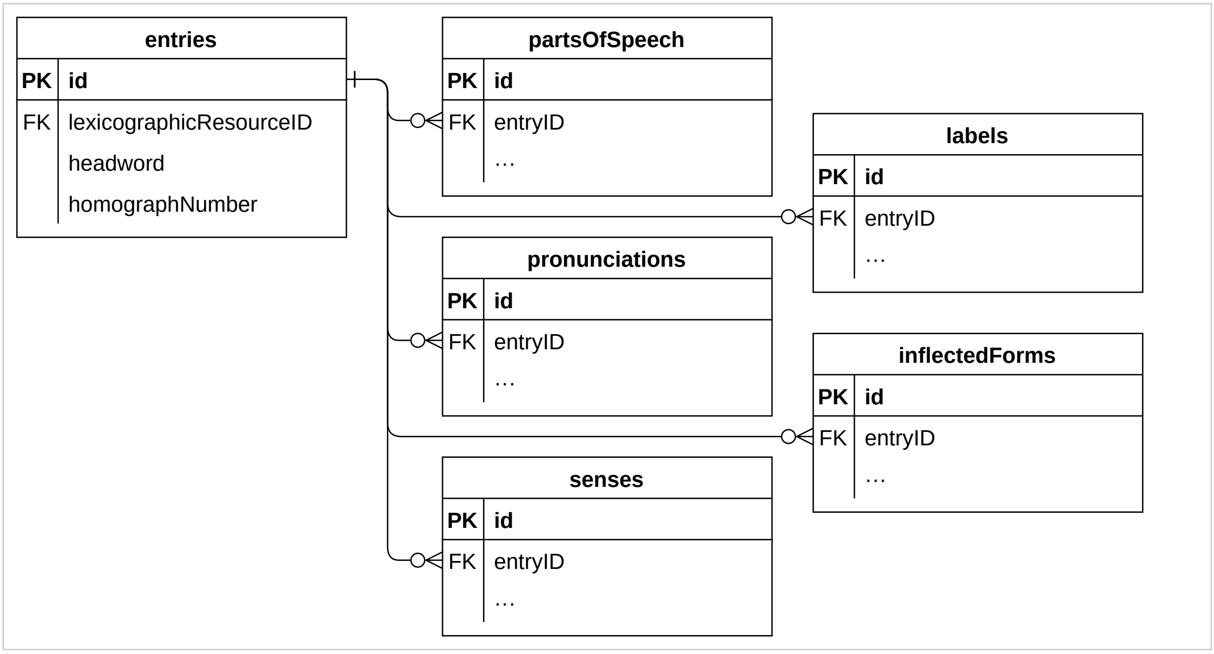
Example 6. JSON

```
{
    "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="..."/>
```

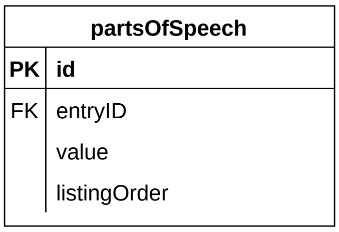
Example 10. JSON

```
"..."
```

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>
```

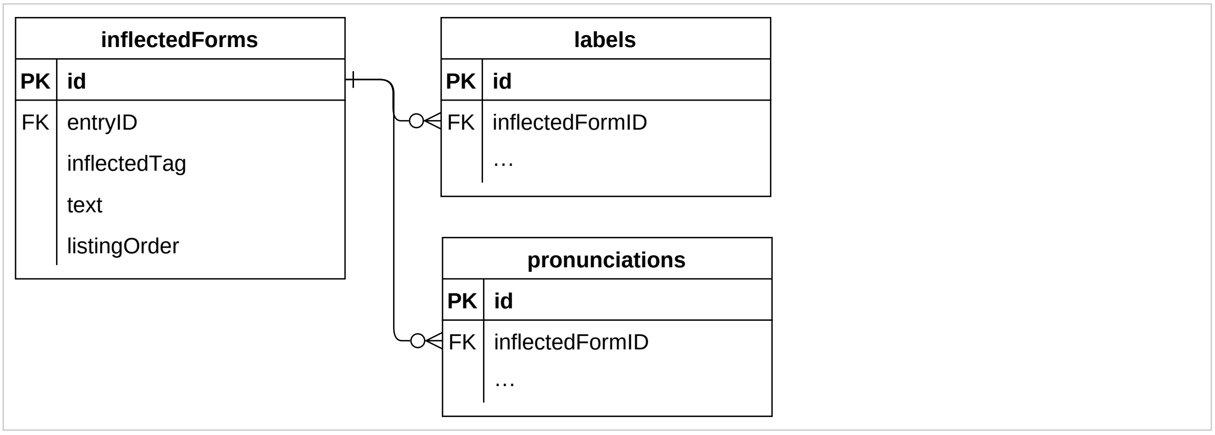
Example 14. JSON

```
{
    "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` OPTIONAL (zero or one). A unique identifier of the sense. Senses which have idenfitiers are capable of being involved in relations created with the Linking module.
- `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. **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>
```

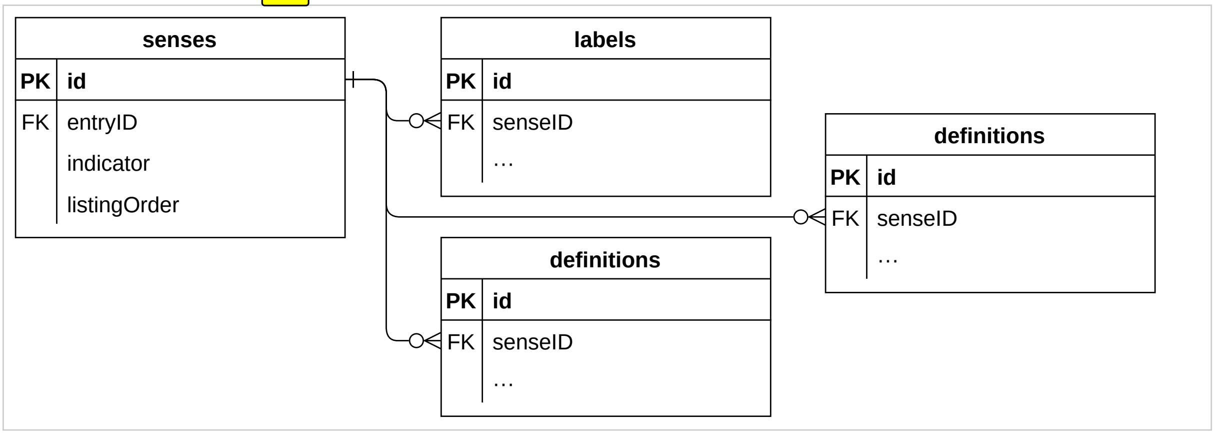
Example 18. JSON

```
{
    "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.