Person

Cf. diagram > iceg-person

Where relevant, the discrepancy with the OSLO Persoon Basis model is explained.

Class  
Attribute

* RegisteredPerson:
  + def: Person whose details are officially listed in a register.
  + usage\_note: This register is usually a population register, but it can also be an electoral register, for example. The registered data relates to the identity (e.g. surname and first name) and the place of residence of the person and to important life events such as birth, marriage, decease, etc. These data are typically registered by the government, they offer the registered person legal protection and they allow the government to compile basic statistics about its population.
  + uri / mapping:

Make RegisteredPerson subclass of registry.Record. The base registry data model is yet to be specified.

The navigation from a registry.Record to the registries it is associated with is provided in the registry data model.

* RegisteredPerson.identifier:
  + def:
  + usage\_note:

1..\* cardinality because person can be associated with multiple identifiers.

Since the entity base registry is explicity modeled as in the base registry data model, it is possible (recommended) to speciliaze Identifier in a BaseRegistryIdentifier which is associated with a registry.

* Person:
  + def: A natural person.
  + usage\_note: In legal terms, it concerns a person (in the legal sense, ie with its own legal personality) of the human species, ie a physical person. The counterpart is the legal person, a legal construction that gives a private or public organization the same legal personality as a natural person (eg can also have debts, conclude contracts, be sued, etc.).
  + uri / mapping:
    - <http://www.w3.org/ns/person#Person> (direct)
* Person.alternativeName:
  + def: Alternative name of the person.
  + usage\_note:
  + uri / mapping:

Seems that this IT covers different scenarios (cf. excel for more details).

I suggest to model the range of this attribute as an object (AlternativeName) with holds a reference to a codelist of types (AlternativeNameType). Another candidate to name this attribute and class is Person.alias → Alias.

* Person.birthName:
  + def: Full name of person at birth.
  + usage\_note: A person's names can change over time, eg can change the last name through marriage. However, the original name is often still used.
  + uri / mapping:

Remark Jonas: not present in RR.

* Person.civilStatus:
  + def: The civil status of the person
  + usage\_note:
  + uri / mapping:
* CivilStatus:
  + def: Civil status of a person.
  + usage\_note: Refers to marriage, civil partnership, descent, guardianship, etc. It is, in other words, the condition of specific relationships between individuals.
  + uri / mapping:

In addition to OSLO Persoon Basis, we suggest to add possible associations to PersonRelation from the class CivilState. This adds another navigation option to relevant personrelations.

Set a codelist.

* Person.contactDetails:
  + def: Information such as email, telephone that allows the Person to be contacted.
  + usage\_note:
  + uri / mapping:

Contact details are defined by Belgian law and should (at a minimum) include:

* een vast telefoonnummer
* een gsm-nummer
* een faxnummer
* een e-mailadres
* Person.familiyName:
  + def: Part of person's full name received from the previous generation.
  + usage\_note:
  + uri / mapping:

Set a codelist for specific cases (e.g. unknown last name).

* Person.fullName:
  + def: The full name of the person, usually the combination of given names and family name.
  + usage\_note:
  + uri / mapping:
* Person.gender:
  + def: The administrative gender of the person.
  + usage\_note:
  + uri / mapping:

Remark Jonas resolved.

* GenderCode:
  + def: Gender of a person, following the ISO 5218 standard: 0 = unknown, 1 = male, 2 = female
  + usage\_note: See <https://en.wikipedia.org/wiki/ISO/IEC_5218> (excluded value: 9)
  + uri / mapping:
* Person.givenName:
  + def: Name given to a child at birth. Distinguishes the child from the other children in the family.
  + usage\_note:
  + uri / mapping:
* Person.nobilityTitle:
  + def: Honorary distinction.
  + usage\_note:
  + uri / mapping:

0..\* cardinality? Can a person have multiple titles?

* NobilityTitle:

Set a codelist.

* Person. patronymicName:
  + def: Name based on the given name of the Person's father.
  + usage\_note:
  + uri / mapping:

Remark Jonas: Remark Jonas: not present in RR.

* Person.usedGivenName:
  + def: Most important of the given names of the person (given name aka first name).
  + usage\_note:
  + uri / mapping:
* Person.citizenship:
  + def: Citizenship of the person.
  + usage\_note: The citizenship entity describes citizenship in more detail (including the jurisdiction in which it is defined).
  + uri / mapping:

Review need to record both nationality and citizenship. Should different codelists be used? In line with SEMIC Core Person, we recommend to use only citizenship.

The act of granting the citizenship is probably best modelled as a public service combined with a base registry transaction. Place of grant / reason of grant are attributes which should be set there.

Use a Jurisdiction as place of grant. It is advisable to organise jurisdictions in Belgium in a well-maintained SKOS BE jurisdiction concept scheme. Include the NIS codes of municipalities and other administrative levels.

Set a codelist for citizenship. This should encompass the jurisdiction concept scheme and also include the special codes for asylum seekers and stateless persons. Also special code for voided citizenship.

Set a codelist for legal reason.

* Citizenship:
  + def: Legal connection of a Person with a state, with which certain rights and obligations are attached.
  + usage\_note:
  + uri / mapping:
* Jurisdiction:
  + def: The territory over which the jurisdiction of a government extends.
  + usage\_note: Typically a country or a state.
  + uri / mapping:

We suggest to add following attributes to Jurisdiction: auhority and location (not present in OSLO Persoon Basis).

* Jurisdiction.authority:
* Jurisdiction.location:
* Person.birth:
  + def: Refers to the birth details of the person.
  + usage\_note:
  + uri / mapping:
* LifeEvent
  + def: Significant event in the life of a person.
  + usage\_note:
  + uri / mapping:
* Birth
  + def: The person being born.
  + usage\_note:
  + uri / mapping:
* Person.death:
  + def: Refers to the death details of the person.
  + usage\_note:
  + uri / mapping:
* Death
  + def: Death of a person.
  + usage\_note:
  + uri / mapping:
* Person.residency:
  + def: Residency of the person.
  + usage\_note:
  + uri / mapping:
* Residency
  + def: The fact that a person resides in a place or country.
  + usage\_note:
  + uri / mapping:
* Residency.interval:
  + def:
  + usage\_note:
  + uri / mapping:
* Residency.residence:
  + def: Refers to the residence details of the residency.
  + usage\_note:
  + uri / mapping:
* Residency.residencyStatus:
  + def:
  + usage\_note:
  + uri / mapping:
* ResidencyStatus:
  + def:
  + usage\_note:
  + uri / mapping:
* Residence
  + def: Place where a Person lives or stays permanently or temporarily.
  + usage\_note:
  + uri / mapping:
* Residence.address:
  + def: The address of the residence.
  + usage\_note:
  + uri / mapping:
* Residence.residenceType:
  + def:
  + usage\_note:
  + uri / mapping:
* Person.memberOf:
  + def: Refers to the household to which the person belongs.
  + usage\_note:
  + uri / mapping:
* Household:
  + def: Form of cooperation that forms a recognizable social unit (eg through marriage), with persons who are blood relatives or not, who have lasting ties and provide mutual support and care.
  + usage\_note: It is a broader concept than family where the family consists not solely of close relatives, eg a single-person family or living separately together. Household members usually share the same residence.
  + uri / mapping:
* Household.householdAddress:
  + def: The address of the residence of the household.
  + usage\_note: Often used as a criterion to determine whether a person belongs to a given household.
  + uri / mapping:
* Person.hasRelationWith:
  + def: Other person with which the person is related.
  + usage\_note:
  + uri / mapping:
* PersonRelation:
  + def: Relation between two or more persons.
  + usage\_note: Typically these are civil law relationships (see civil status) but not necessarily limited thereto.
  + uri / mapping:
* HouseholdRelation:
  + def: Relationship between members of the same family.
  + usage\_note: E.g. husband, son, mother-in-law.
  + uri / mapping:
* Guardianship:
  + def: Situation where a person acting as guardian serves another person and in that capacity is responsible for this person and his goods.
  + usage\_note: This is the case, for example, for minors who have lost their parents or for adults who have lost their intellectual faculties.
  + uri / mapping:
* StatutoryCohabitation:
  + def: Arrangement whereby two people who are not married live together.
  + usage\_note:
  + uri / mapping:
* Marriage:
  + def: A form of cohabitation of two persons, organized by civil or religious law.
  + usage\_note: Can, just like living together, form the basis of a household.
  + uri / mapping:
* Descent:
  + def: Descent is the descending line of blood relationship between different generations.
  + usage\_note: The descent can be biological as well as non-biological, eg adoption. Descent can also be viewed in the ascending line. The concept is narrower than kinship that also includes marriage. There must be a direct link between the persons (no link between child and grandparent).
  + uri / mapping:

Generic

Cf. diagram > iceg-generic

We suggest to rely on following generic entities.

Class  
Attribute

20240321 – Historic versioning

An authentic source[[1]](#footnote-1) must implement some form of historic versioning. It is highly likely that the registry will also have to track the why and who of inserts and updates in the source. Here we focus on the modeling approaches to represent evolution over time of underlying data, and consider the why and who as temporarily out of scope.

The approach we suggest is to work with an EAV (entity-attribute-value) model. THE [EAV model](https://en.wikipedia.org/wiki/Entity%E2%80%93attribute%E2%80%93value_model) is a well-known technique to have an explicit record of changes happening to data over time.

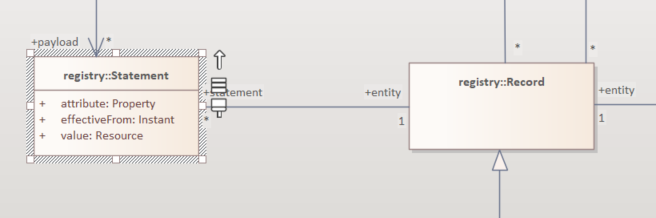


Figure - Statement ↔ Record relationship

This approach does not have have to be implemented at the persistence level and be chosen solely as a way to expose evolution of data over time.

Applied to person data, the logic goes as follows. A person:RegisteredPerson is a registry:Record. A registry:record has a many-to-many relationship with registry:Registry. It has a one-to-many relationship with registry:Statement. Such as statement combines:

* An entity, the record which is the subject of the statement.
* An attribute, the data element which is manipulated.
* A value.

The statement should also contain the following time-based data points:

* effectiveFrom timestamp, namely a timestamp as of which the data is applicable.
* Transaction timestamp namely the timestamp of the transaction in the registry which encapsulates the statement.

An important thing to note is that we rely on some awareness of the underlying data model. In the above example, the attributes of the person data model are made available at registry level in order to be referenced from the statement table.

A short implementation of these concepts is available [here](https://github.com/belgif/ICEGthema-person/tree/main/resources/examples/registry).

This approach has a number of benefits. One of them is to decouple underlying and registry data. It is preferable not to add snapshot (aka materialised version) to each aspect of the underlying data model. Such as approach would create a tight coupling between the underlying schema and the registry. The materialised version approach also results in a higher volume of data.

1. We use the terms base registry or authentic source interchangeably. [↑](#footnote-ref-1)