# 571: Cardinality of typed properties

this is the new Applied Form section

**Applied Form**

The CIDOC CRM is an ontology in the sense used in computer science. It has been expressed as an object-oriented semantic model, in the hope that this formulation will be comprehensible to both documentation experts and information scientists alike, while at the same time being readily converted to machine-readable formats such as RDF Schema or OWL. A CRM conformant documentation system can be implemented using RDF Schema or OWL, but also in Relational or Object-Oriented schema. CIDOC CRM instances can be encoded in RDF, JSON LD, XML, OWL and others.

More specifically, the CIDOC CRM is expressed in terms of the primitives of semantic data modelling. As such, it consists of:

* *classes*, which represent general notions in the domain of discourse, such as the CIDOC CRM class E21 Person which represents the notion of person;
* *properties*, which represent the binary relations that link the individuals in the domain of discourse, such as the CIDOC CRM property P152 has parent linking a person to one of the person’s parent.
* *properties of properties*, such as the property *P14.1 in the role of*, of the CIDOC CRM property P14 carried out by (see also section “About Types”).   
  They do not appear in the property hierarchy list, but are included as part of their base property declaration and are referred to in the class declarations. They all have the implicit quantification “many to many” (see also section “Property Quantifiers”)

Although the definition of the CIDOC CRM provided here is complete, it is an intentionally compact and concise presentation of the CIDOC CRM’s 81 classes and 160 unique properties. It does not attempt to articulate the inheritance of properties by subclasses throughout the class hierarchy (this would require the declaration of several thousand properties, as opposed to 160). However, this definition does contain all of the information necessary to infer and automatically generate a full declaration of all properties, including inherited properties.

1. ***Naming Conventions***

The following naming conventions have been applied throughout the CIDOC CRM:

* Classes are identified by numbers preceded by the letter “E” (historically classes were sometimes referred to as “Entities”), and are named using noun phrases (nominal groups) using title case (initial capitals). For example, E63 Beginning of Existence.
* Properties are identified by numbers preceded by the letter “P,” and are named in both directions using verbal phrases in lower case. Properties with the character of states are named in the present tense, such as “has type”, whereas properties related to events are named in past tense, such as “carried out.” For example, *P126 employed (was employed in)*.
* Property names should be read in their non-parenthetical form for the domain-to-range direction, and in parenthetical form for the range-to-domain direction. Reading a property in range-to-domain direction is equivalent to the inverse of that property. Following a current notational practice in OWL knowledge representation language, we represent inverse properties in this text by adding a letter “i” following the identification number and the parenthetical form of the full property name, such as *P59i is located on or within*, which is the inverse of *P59 has section (is located on or within).*
* Properties with a range that is a subclass of E59 Primitive Value (such as E1 CRM Entity*. P3 has note:* E62 String, for example) have no parenthetical name form, because reading the property name in the range-to-domain direction is not regarded as meaningful.
* Properties that have identical domain and range are either symmetric or transitive. Instantiating a symmetric property implies that the same relation holds for both the domain-to-range and the range-to-domain directions. An example of this is E53 Place*. P122 borders with:* E53 Place. The names of symmetric properties have no parenthetical form, because reading in the range-to-domain direction is the same as the domain-to-range reading. Transitive asymmetric properties, such as E4 Period*. P9 consist of (forms part of):* E4 Period, have a parenthetical form that relates to the meaning of the inverse direction.
* The choice of the domain of properties, and hence the order of their names, are established in accordance with the following priority list:
  + Temporal Entity and its subclasses
  + Thing and its subclasses
  + Actor and its subclasses
  + Other
* Properties of properties are identified by “P”, followed by the number of the base property extended with “.1” and are named in one direction using a verbal phrase in lower case in the present tense. For example: the property *P62.1 mode of depiction* of the property *P62 depicts (is depicted by)*

***Inheritance and Transitivity***

CIDOC CRM is formulated as a class system with inheritance. A property P with domain A and range B will also be a property between any possible subclasses of A and of B. In many cases there will be a common subclass C of both A and B. In these cases, when the property is restricted to C, that is, with C as domain and range, the restricted property could be transitive. For instance, an E73 Information Object can be incorporated into an E90 Symbolic Object and thus an information object can be incorporated in another information object.

In the definition of CIDOC CRM the transitive properties are explicitly marked as such in the scope notes. All unmarked properties should be considered as not transitive.

***Shortcuts***

Some properties are declared as shortcuts of longer, more comprehensively articulated paths that connect the same domain and range classes as the shortcut property via one or more intermediate classes. For example, the property E18 Physical Thing*. P52 has current owner (is current owner of)*:E39 Actor, is a shortcut for a fully articulated path from E18 Physical Thing through E8 Acquisition to E39 Actor. An instance of the fully-articulated path always implies an instance of the shortcut property. However, the inverse may not be true; an instance of the fully-articulated path cannot always be inferred from an instance of the shortcut property inside the frame of the actual KB

The class E13 Attribute Assignment allows for the documentation of how the assignment of any property came about, and whose opinion it was, even in cases of properties not explicitly characterized as “shortcuts”.

1. ***About the logical expressions used in the CIDOC CRM***

The present CIDOC CRM specifications are annotated with logical axioms, providing an additional formal expression of the CIDOC CRM ontology. This section briefly introduces the assumptions that are at the basis of the logical expression of the CIDOC CRM (for a fully detailed account of the logical expression of semantic data modelling, see (Reiter,1984)).

In terms of semantic data modelling, classes and properties are used to express ontological knowledge by means of various kinds of constraints, such as sub-class/sub-property links, e.g., E21 Personis a sub-class ofE20 Biological Object, or domain/range constraints, e.g., the domain of *P152 has parent* is class E21 Person*.*

In contrast, first-order logic-based knowledge representation relies on a language for formally encoding an ontology. This language can be directly put in correspondence with semantic data modelling in a straightforward way:

* classes are named by *unary predicate symbols*; conventionally, we use E21 as the unary predicate symbol corresponding to class E21 Person;
* properties are named by *binary predicate symbols*; conventionally, we use P152 as the binary predicate symbol corresponding to property *P152 has parent.*
* properties of properties, “.1 properties” are named by *ternary predicate symbols*; conventionally, we use P14.1 as the ternary predicate symbol corresponding to property *P14.1 in the role of.*

Ontology is expressed in logic by means of *logical axioms*, which correspond to the constraints of semantic modelling. In the definition of classes and properties of the CIDOC CRM the axioms are placed under the heading ‘In first order logic’. There are several options for writing statements in first order logic. In this document we use a standard compact notation widely used in text books and scientific papers. The definition is given in the table below.

*Table 1: Symbolic Operators in First Order Logic Representation*

|  |  |  |  |
| --- | --- | --- | --- |
| Symbol | Name | reads | Truth value |
| Operators |  |  |  |
| ∧ | conjunction | and | (φ ∧ ψ) is true  if and only if both *φ* and *ψ* are true |
| ∨ | disjunction | or | (φ ∨ ψ) is true  if and only if at least one of either φ or ψ is true |
| ¬ | negation | not | ¬φ is true if and only if *φ* is false |
| ⇒ | implication | implies,  if … then ... | (φ ⇒ ψ) is true  if and only if it is not the case that *φ* is true and *ψ* is false |
| ⇔ | equivalence | is equivalent to,  if … and only if … | φ ⇔ ψ is true  if and only if both *φ* and *ψ* are true or both φ and ψ are false |
| Quantifiers |  |  |  |
| ∃ | existential quantifier | exists,  there exists at least one |  |
| ∀ | Universal quantifier | forall,  for all |  |

For instance, the above sub-class link between E21 PersonandE20 Biological Object can be formulated in first order logic as the axiom:

(∀x) [E21(x) ⇒E20(x)]

(reading: for all individuals x, if x is a E21 then x is an E20).

In the definitions of classes and properties in this document the universal quantifier(s) are omitted for simplicity, so the above axiom is simply written:

E21(x) ⇒E20(x)

Likewise, the above domain constraint on property *P152 has parent* can be formulated in first order logic as the axiom:

P152(x,y) ⇒E21(x)

(reading: for all individuals x and y, if x is a P152 of y, then x is an E21).

These basic considerations should be used by the reader to understand the logical axioms that are used into the definition of the classes and properties. Further information about the first order formulation of CIDOC CRM can be found in (Meghini & Doerr, 2018)

***Property Quantifiers***

Quantifiers for properties are provided for the purpose of semantic clarification only, and should **not** be treated as implementation recommendations. The CIDOC CRM has been designed to accommodate alternative opinions and incomplete information, and therefore **all** properties should be implemented as optional and repeatable for their domain and range (“many to many (0,n:0,n)”). Therefore, the term “cardinality constraints” is avoided here, as it typically pertains to implementations.

The following table lists all possible property quantifiers occurring in this document by their notation, together with an explanation in plain words. In order to provide optimal clarity, two widely accepted notations are used redundantly in this document, a verbal and a numeric one. The verbal notation uses phrases such as “one to many”, and the numeric one, expressions such as “(0,n:0,1)”. While the terms “one”, “many” and “necessary” are quite intuitive, the term “dependent” denotes a situation where a range instance cannot exist without an instance of the respective property. In other words, the property is “necessary” for its range. (Meghini, C. & Doerr, M., 2018)

|  |  |
| --- | --- |
| **many to many (0,n:0,n)** | Unconstrained: An individual domain instance and range instance of this property can have zero, one or more instances of this property. In other words, this property is optional and repeatable for its domain and range. |
| **one to many**  **(0,n:0,1)** | An individual domain instance of this property can have zero, one or more instances of this property, but an individual range instance cannot be referenced by more than one instance of this property. In other words, this property is optional for its domain and range, but repeatable for its domain only. In some contexts, this situation is called a “fan-out”. |
| **many to one**  **(0,1:0,n)** | An individual domain instance of this property can have zero or one instance of this property, but an individual range instance can be referenced by zero, one or more instances of this property. In other words, this property is optional for its domain and range, but repeatable for its range only. In some contexts, this situation is called a “fan-in”. |
| **many to many, necessary (1,n:0,n)** | An individual domain instance of this property can have one or more instances of this property, but an individual range instance can have zero, one or more instances of this property. In other words, this property is necessary and repeatable for its domain, and optional and repeatable for its range. |
| **one to many, necessary**  **(1,n:0,1)** | An individual domain instance of this property can have one or more instances of this property, but an individual range instance cannot be referenced by more than one instance of this property. In other words, this property is necessary and repeatable for its domain, and optional but not repeatable for its range. In some contexts, this situation is called a “fan-out”. |
| **many to one, necessary**  **(1,1:0,n)** | An individual domain instance of this property must have exactly one instance of this property, but an individual range instance can be referenced by zero, one or more instances of this property. In other words, this property is necessary and not repeatable for its domain, and optional and repeatable for its range. In some contexts, this situation is called a “fan-in”. |
| **one to many, dependent**  **(0,n:1,1)** | An individual domain instance of this property can have zero, one or more instances of this property, but an individual range instance must be referenced by exactly one instance of this property. In other words, this property is optional and repeatable for its domain, but necessary and not repeatable for its range. In some contexts, this situation is called a “fan-out”. |
| **one to many, necessary, dependent**  **(1,n:1,1)** | An individual domain instance of this property can have one or more instances of this property, but an individual range instance must be referenced by exactly one instance of this property. In other words, this property is necessary and repeatable for its domain, and necessary but not repeatable for its range. In some contexts, this situation is called a “fan-out”. |
| **many to one, necessary, dependent**  **(1,1:1,n)** | An individual domain instance of this property must have exactly one instance of this property, but an individual range instance can be referenced by one or more instances of this property. In other words, this property is necessary and not repeatable for its domain, and necessary and repeatable for its range. In some contexts, this situation is called a “fan-in”. |
| **one to one**  **(1,1:1,1)** | An individual domain instance and range instance of this property must have exactly one instance of this property. In other words, this property is necessary and not repeatable for its domain and for its range. |

The CIDOC CRM defines some dependencies between properties and the classes that are their domains or ranges. These can be one or both of the following:

* the property is necessary for the domain
* the property is necessary for the range, or, in other words, the range is dependent on the property.

The possible kinds of dependencies are defined in the table above. Note that if a dependent property is not specified for an instance of the respective domain or range, it means that the property exists, but the value on one side of the property is unknown. In the case of optional properties, the methodology proposed by the CIDOC CRM does not distinguish between a value being unknown or the property not being applicable at all. For example, one may know that an object has an owner, but the owner is unknown. In a CIDOC CRM instance this case cannot be distinguished from the fact that the object has no owner at all. Of course, such details can always be specified by a textual note.

Note that the quantification of all properties of properties, “.1” properties, is “many-to-many” and, therefore, does not appear explicitly in their definitions.