3. CSDP Step 6

> Step 6: Add value, set comparison and subtyping constraints.

General Rule about Populations of Entity Types

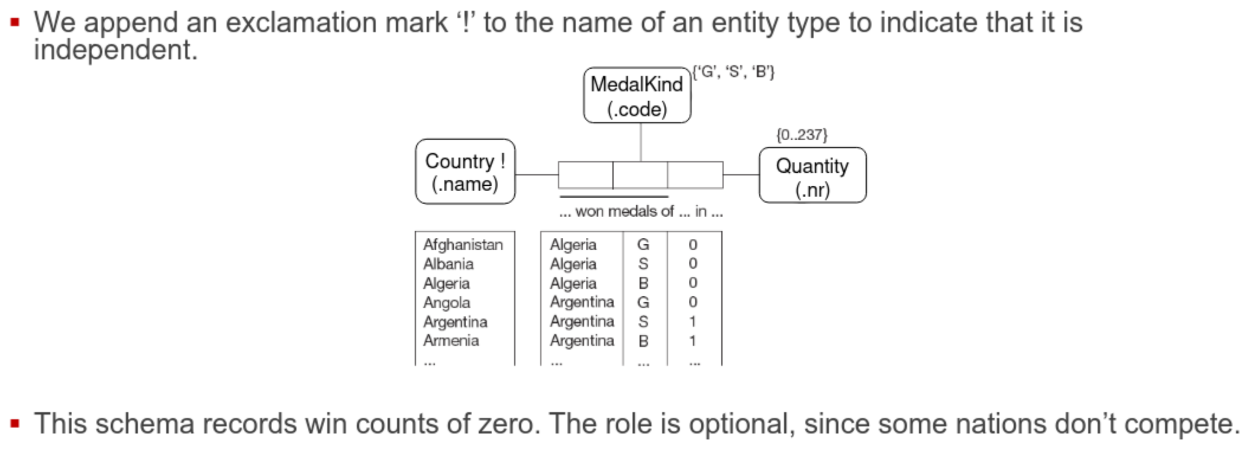
The population of an entity type equals the union of the population of its fact roles. This is a general rule about populations of entity types.

Exception to the Rule: Independence

An independent object type is a primitive entity type, or a nested fact type whose fact roles are collectively optional. An independent object type exists in a reference table or an object table, but does not need to do anything.

- If an independent object type plays one or more fact roles, the disjunction of these roles is not mandatory.

- To indicate that an entity is independent, we add an exclamation mark to the name of the entity.



- We append an exclamation mark '!' to the name of an entity type to indicate that it is independent.

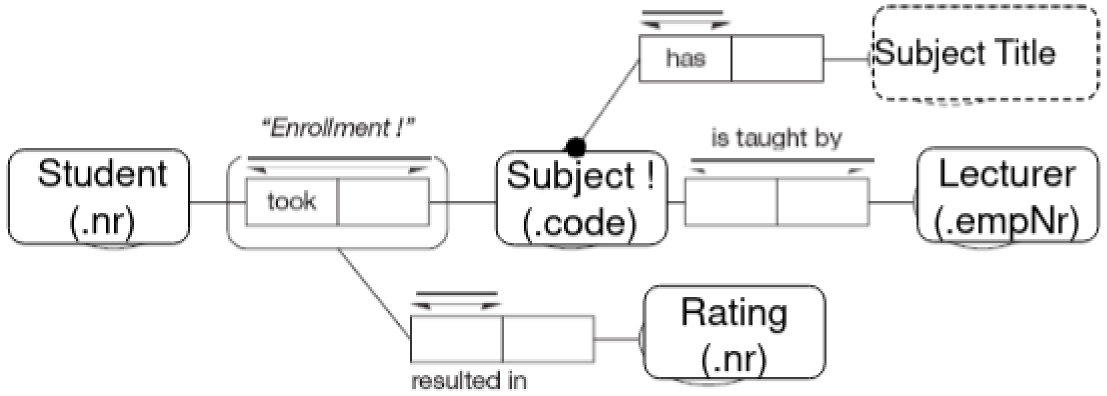
Afghanistan
Albania
Algeria
Angola
Argentina
Armenia
...

Algeria	G	0
Algeria	S	0
Algeria	B	0
Argentina	G	0
Argentina	S	1
Argentina	B	1
...	...	...

- This schema records win counts of zero. The role is optional, since some nations don't compete.

Nested fact types are more likely to be made independent because certain associations often need to be recorded before recording any facts about them as a collective.

For example, in a student system where students enroll in subjects and we record their grades, the schema allows us to record the fact that a student enrolled in a subject before knowing what result they achieved for that enrollment.

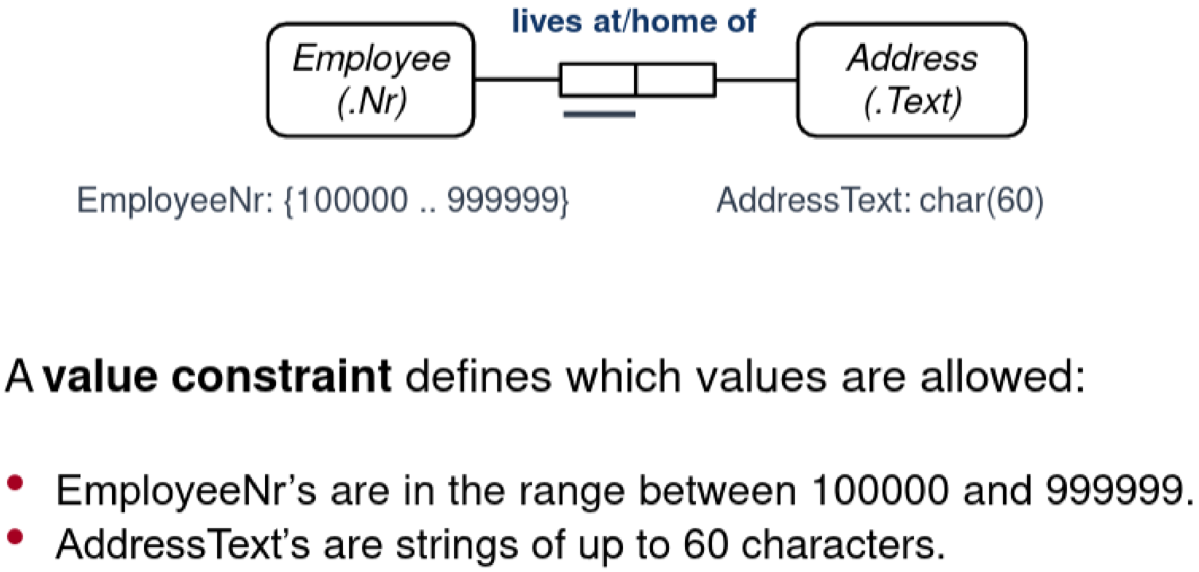


Subject is also independent in this example because a subject needs to exist before we can add an enrollment of a student to a subject.

Value Constraints

Value constraints define the values which are allowed for a particular attribute. This could be:

- A range
- Above or below a certain value
- Distinct values



A **value constraint** defines which values are allowed:

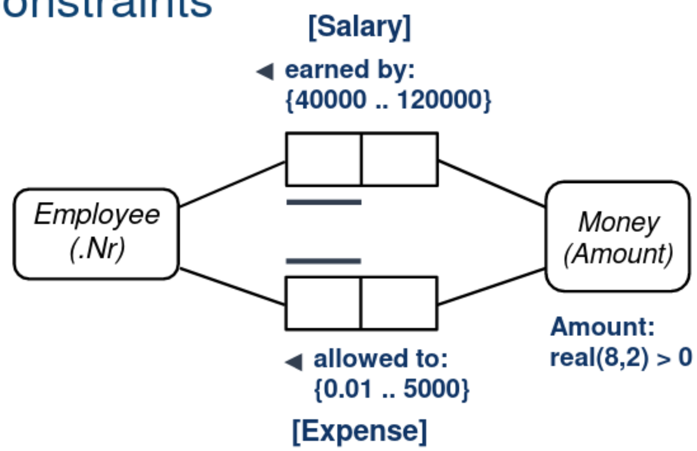
- EmployeeNr's are in the range between 100000 and 999999.
- AddressText's are strings of up to 60 characters.

Role value constraints can also be shown. For example:

- Money is any amount captured as a real number larger than zero, less than 999,999.99
- Salary can only hold values between $40,100 and $120,000

- Expenses are only allowed under $5,000

Role Value Constraints


[Salary]

◀ earned by: {40000 .. 120000}

Employee (.Nr)

Money (Amount)

Amount: real(8,2) > 0

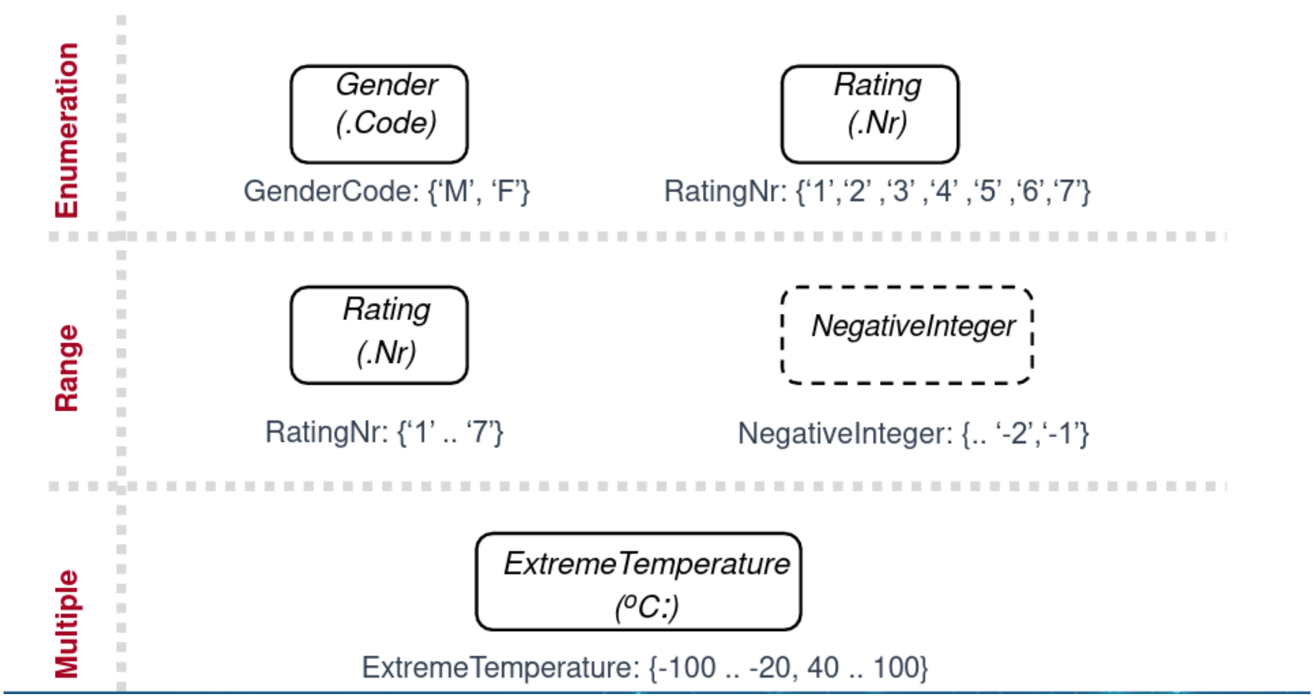
◀ allowed to: {0.01 .. 5000}

[Expense]

- **Money** is identified by any **Amount** captured as a real number larger than 0 and less than 999999.99 with two digits after the dot.
- **Salary** can only hold values between 40 000 and 120 000.
- Only under 5 000 **Expenses** are allowed.

In an ORM diagram, value type constraints can be shown using:

- Enumeration
- Ranges
- Multiple values


Enumeration

Gender (.Code)

GenderCode: {'M', 'F'}

Rating (.Nr)

RatingNr: {'1','2' ,'3' ,'4' ,'5' ,'6','7'}

Range

Rating (.Nr)

RatingNr: {'1' .. '7'}

NegativeInteger

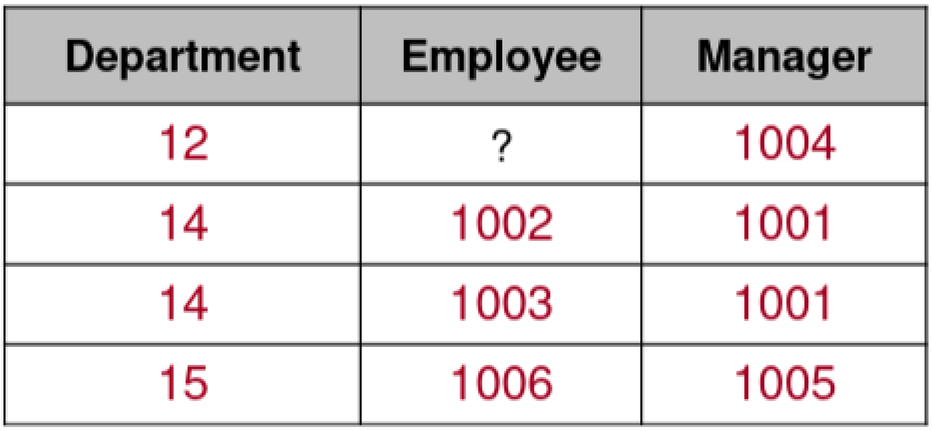
NegativeInteger: {.. '-2','-1'}

Multiple

ExtremeTemperature (ºC:)

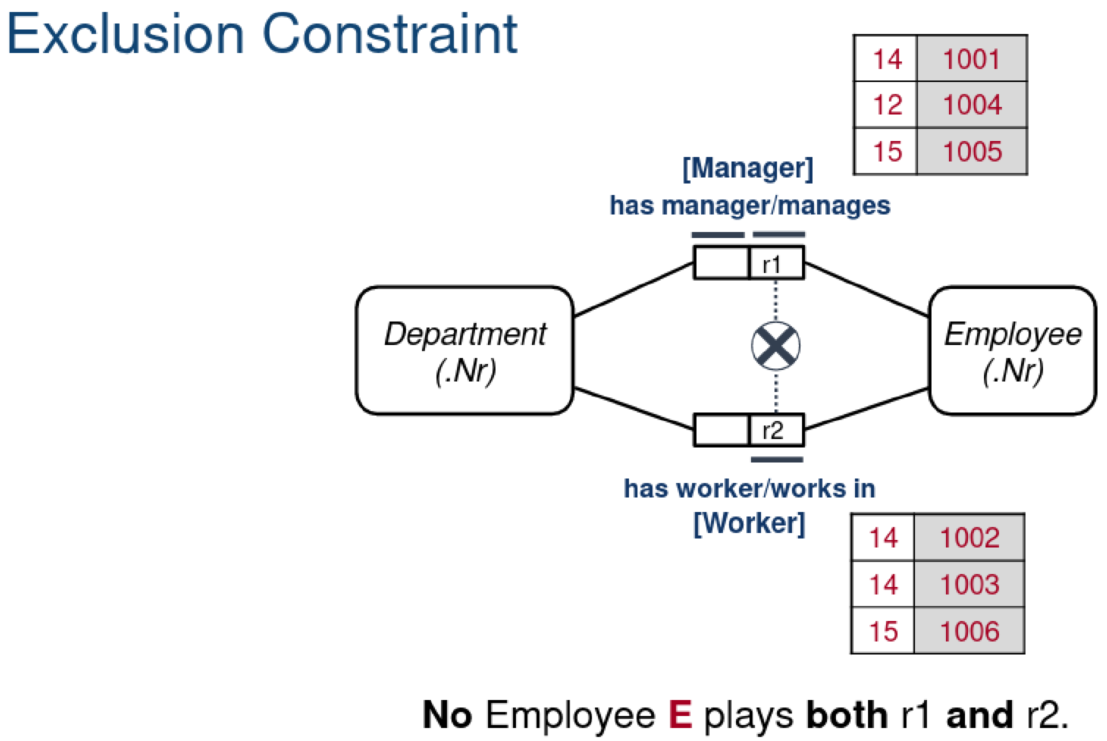
ExtremeTemperature: {-100 .. -20, 40 .. 100}

Exclusion Constraint



Department	Employee	Manager
12	?	1004
14	1002	1001
14	1003	1001
15	1006	1005

The exclusion constraint is shown as a circle with a cross in it.



Exclusion Constraint

14	1001
12	1004
15	1005

14	1002
14	1003
15	1006

No Employee **E** plays **both** r1 **and** r2.

In the example provided, we have a department with workers and managers. The exclusion constraint specifies that no employee can have both roles, i.e., be a worker and a manager.

See Also

1. CSDP Step 4