Data models and flexibility patterns

To support change efficiently

Table of content

What is flexibility ?

and what is it good for from a data perspective ?

- Integrate data model changes rapidly
- Manage "objects" shared by business domains
- Manage specific "objects" attributes per business domain
- Manage permissions per object per attribute per domain per user
- Manage "views" per object/attribute/domain/user
- With no or minimum impact on business application
- With no or minimum impact on the infrastructure
- With no or minimum development needed
- With no or minimum re-deployment needed

What is an "object" ?

In the scope of this presentation, an "object" must be understood in its Object Oriented (OO) meaning i.e.:

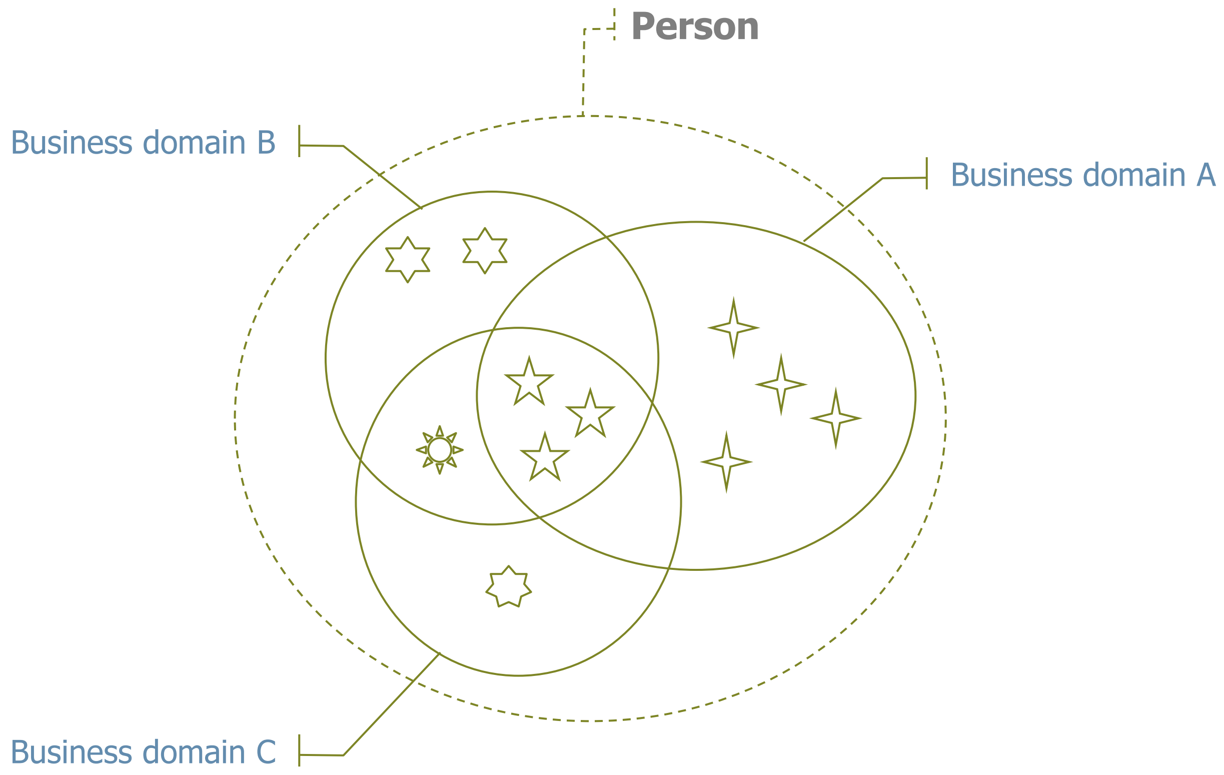
"Objects in object-oriented programming basically are data structures (properties) combined with the associated processing routines (methods)." Wikipedia

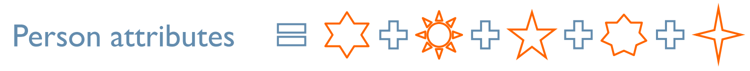
In the scope of a database, an object can be for example a person with its attributes (properties) and the associated "stored procedures" to handle it (methods).

Objects and business domains

- Let's take the example of a person
- A person has a set of attributes like a first and last name, a national registration number, an address, a phone number, etc.
- A person has also a set of attributes which is only meaningful e.g. in the context of the insurability business domain e.g. an insurability flag.
- Each business domain has its specific set of attributes meaningful in their context only
- We can say that a person (an object) has:
  - Attributes **common** to all business domains (name, etc)
  - Attributes **specific to** one or more particular business **domains**

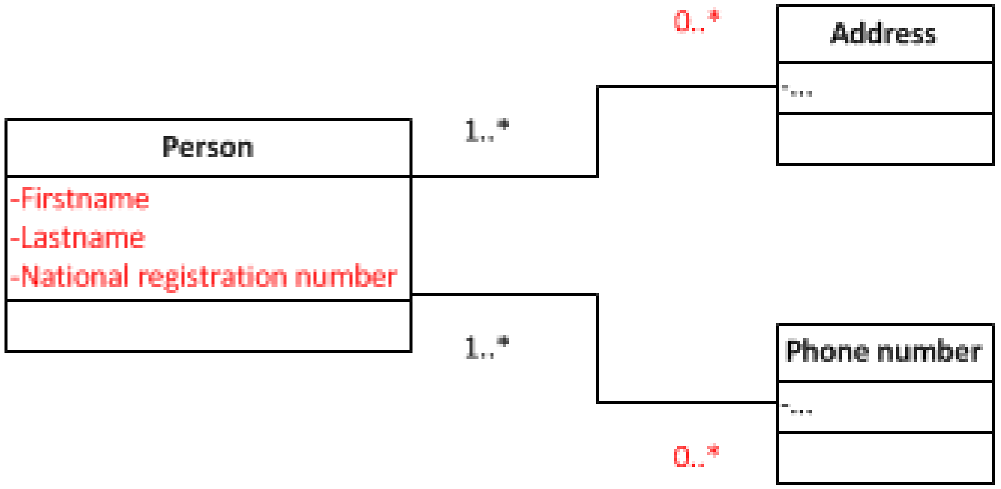
Objects attributes and business domains





Object attributes and cardinality

- Let’s take the example of a person
- A person has one national registration number
- A person can have more than one address
- A person can have more than one telephone number
- Etc.

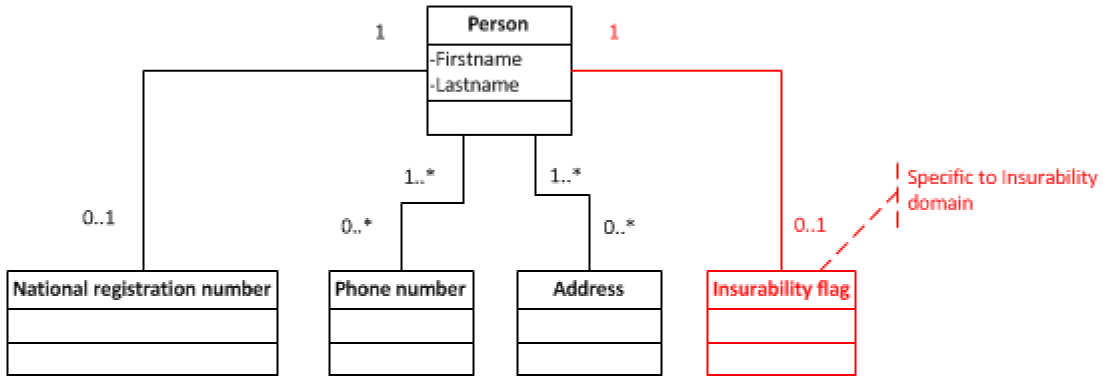


Flexibility patterns :

Attributes externalization

It may be useful to externalize an object attribute even when this object has one and only one attribute of this type. E.g. :

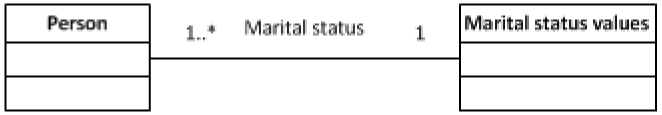
- When the attribute is specific to a particular business domain for data segregation and access control purposes.
- When the attribute is optional
- When the allowed attribute value is picked from a predefined list



Flexibility patterns :

Predefined list of values

Some database objects attributes have a limited set of possible values, e.g. the marital status of a person: bachelor, married, divorced, … It may be useful to externalize these attributes by maintaining a specific table listing these possible values.



Advantages:

- Centralized list of values, no need to search all database to achieve a view of possible values
- Centralized maintenance of values list in database instead of into multiple business applications
- Mandate business applications to pick from the predefined list
- When the naming of a marital status changes it is de-facto reflected in the entire database
- …

Flexibility patterns :

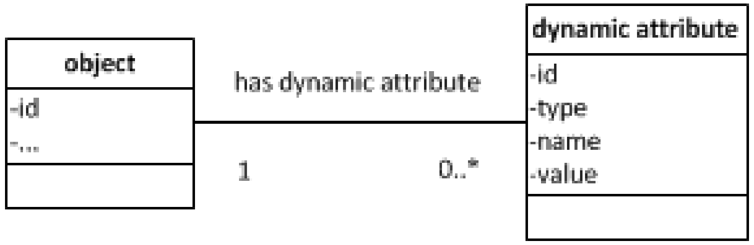
Dynamic attributes

Dynamic object attributes handling consists in maintaining (add, update, delete) object attributes dynamically i.e. without altering the data model. This can be consequently done while the database is up and running.

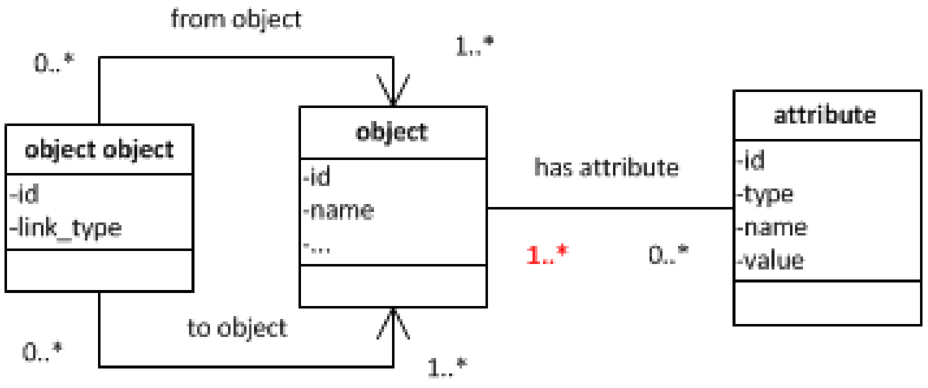
- This is useful when a particular database user or set of users needs to enrich the object with additional information specific to him/it.
- This mechanism is dedicated to users and is not indicated for business domain specific information.
- This concerns small amounts of data for a limited set of users (of a same type)
- Attributes that can be added are of a predefined list of types e.g. text, number, boolean.

Flexibility patterns :

Dynamic attributes model



- It must be noted that it is possible to have a fully generic database implementation where the entire logical model is provisioned dynamically:

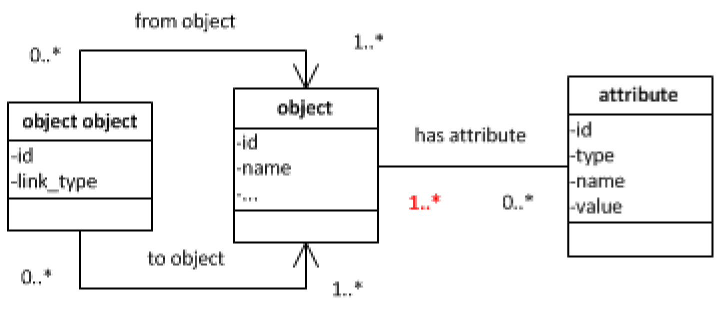


Flexibility patterns :

Fully generic data model

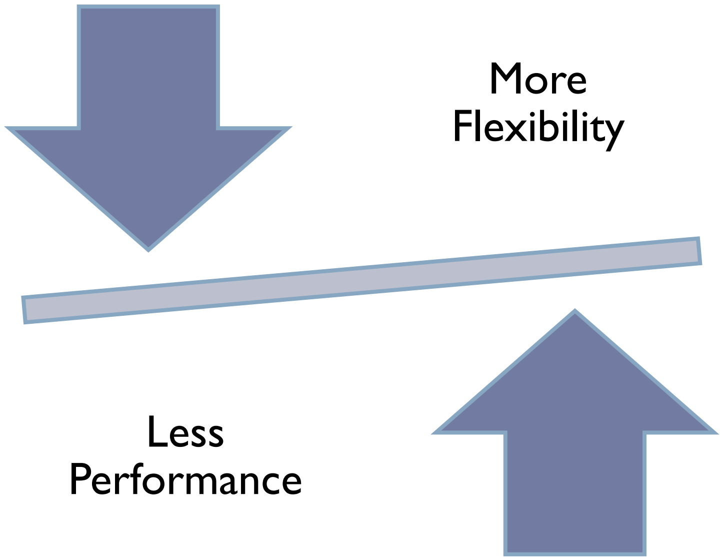
A fully generic database implementation allows to provision a specific logical data model dynamically: objects, objects relationship and objects attributes.

- Offers the maximum flexibility possible
- Requires and additional software layer "knowing" the logical model (referential integrity, circular references, etc management)
- Not suitable for high performance environments



Flexibility Vs performance

- More functional flexibility means less performance
- Business performance needs dictates implementation in this regard



Objects and name spaces

Since an object can have attributes which make sense only in specific business domains, it is useful to know in which business domains an attribute is meaningful. Especially when several business domains have attributes with the same name but different usage/meaning. This can be achieved through the usage of "name spaces".

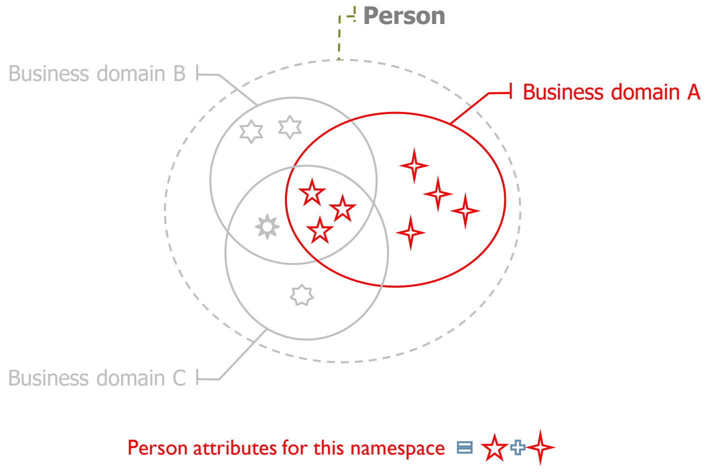
"A namespace is a container for a set of identifiers (names), and allows the disambiguation of homonym identifiers residing in different namespaces. Namespaces usually group names based on their functionality."

This is conventional and could be <domain name>.<object name>.<attribute name>, e.g. insurability.person.insurability_flag.
If one refers to a "common" attribute: person.name

Data access segregation

Objects attributes externalization allows to expose data only meaningful to a specific business domain.

- It must also not be possible for a business domain to modify data it does not own. A business domain can only access/manage data in its namespace.
- It must be possible to expose a logical data model for a specific domain/namespace.



Data model and business logic

In order to minimize the impact of a data model change it is useful to isolate the business application from the data model through:

- Encapsulation of business elementary functions in "stored procedure"
- Stored procedure are stored in the database itself
- But, from a three-tier logic point of view, stored procedures implement part of the business logic and consequently conceptually belong to the business logic layer (not to the data layer).

Stored procedures

- A unique central point of implementation of elementary business functions (business objects methods)
- Minimize impact on business application when data model changes
- Hides database complexity to business applications
- Ensures low level database services like referential integrity

Other topics

- Data cleansing and sanity checks
- Time-stamping
- Referential integrity
- Objects aggregation
- Technical Vs business unique identifiers
- …

End of presentation

Questions