

# Jiří Brožek, Vojtěch Merunka, Martin Molhanec, Martin Šebek

- 1) Czech University of Life Sciences Prague, Faculty of Economics and Management,  
Dept. of Information Engineering
- 2) Czech Technical University, Faculty of Nuclear Sciences and Physical Engineering,  
Dept. of Software Engineering in Economy

## Normalization Rules of the Object- Oriented Data Model

# How many conceptual data models we have?

feature	model	description
R	Entity-Relational	This is the traditional RDBMS model based on Chen. <i>Here is the area of the relational normalization.</i>
C	Network	This is the model of the network databases (IDMS).
I	???	??? This conceptual model does not exist. ???
RC	Hybrid Network and Entity-Relational	This is the RDBMS model combined with data containers (e.g. NF <sup>2</sup> = non-first normal form databases).
RI	Extended Entity-Relational	This is the RDBMS model extended by the inheritance (e.g. IDEF1X and some object-relational DBMS).
CI	OOP Data Model	This is the data model of the recent object-oriented programming languages (e.g. Java, Smalltalk, C#, ... and many programming-language-based OODBMS). <i>Here is the area of our approach.</i>
RCI	The full/universal conceptual model	This data model includes all conceptual features and reflects the proposed ODMG 3.0 and UML 2.0 standards, but is not yet directly implemented in recent object-oriented programming languages. <i>Do we need a universal normalization technique?</i>

# Object Oriented Applications

- Nowadays many various kinds of object-oriented software applications are used practically. Although there already are many theoretical works, which individually demonstrate suitability of non-relational object-oriented data model, only the procedures based on experience with imperative object-oriented programming languages are used in the area of analysis and design of data structures.
- However techniques like behavioral design patterns or object library components, which are optimal for algorithms in software application, can fundamentally complicate effective data processing of OODB.
- Moreover, relational design techniques as data normalization, data decomposition and data synthesis cannot be easily used in object-oriented data structures.

# Miscellaneous approaches of object-oriented normalization

- Nootenboom's OONF
- Khodorkovsky's ONF, 4ONF, 5ONF and 6ONF
- Australian-Swiss ONF
- ...
- ...
- **Three Ambler-Beck's object normal forms**

# Three Ambler-Beck's object normal forms

Ambler and Beck are pioneers of agile approach in programming. They introduced three object-oriented normal forms for object-oriented applications.

These normal forms are analogous with first, second and third relational normal forms. The authors talk about these object normal forms as a tool for class structure normalization complementary with technique of design patterns.

# Our Experience

We have good results with Ambler-Beck approach. But we have found that object-oriented community expects bit different technique:

1. It has to be very simple, and understandable and should work with minimum of abstract concepts, similarly as "the classical relational normalization". We suppose that introduction of difficult definitions distinctively exceeding over the range of classical normal forms by having a lot of types of concepts and relations, is not the right way.
2. It should be focused concretely on object-oriented modeling of data structures. We need to model structures of objects used for data storage and data manipulation. We do not need to model objects responsible for functional behavior of applications. For these another "behavioral" objects we already have design patterns and other programming techniques. We do not need to duplicate these proven techniques. We think that original Ambler-Beck's approach needlessly tries to solve everything in one.

# First Normal Form

**Definition 1.** *A class is in the first object normal form (1ONF) when its objects do not contain group of repetitive attributes. Repetitive attributes must be extracted into objects of a new class. The group of repetitive attributes is then replaced by the link at the collection of the new objects. An object schema is in 1ONF when all of its classes are in 1ONF.*

More formally; Let us have an object  $a$  in the object system  $\Omega$  as  $a \in \Omega$ , where for  $k > 1$  (length of collections of similar attributes) and  $n > 1$  (number of repetition of these collections) is  $data(a) = [\dots, x_1^1, \dots, x_k^1, \dots, x_1^n, \dots, x_k^n, \dots]$  having  $\forall i \in (1, \dots, k) : class(x_i^1) = class(x_i^2) = \dots = class(x_i^n)$ .

Then it is required to modify object  $a$  and create new objects  $b_j \in \Omega$  for  $j \in (1, \dots, n)$  as  $data(a) = [\dots, \{b_j\}, \dots]$  and  $data(b_j) = [x_1^j, \dots, x_k^j]$ .

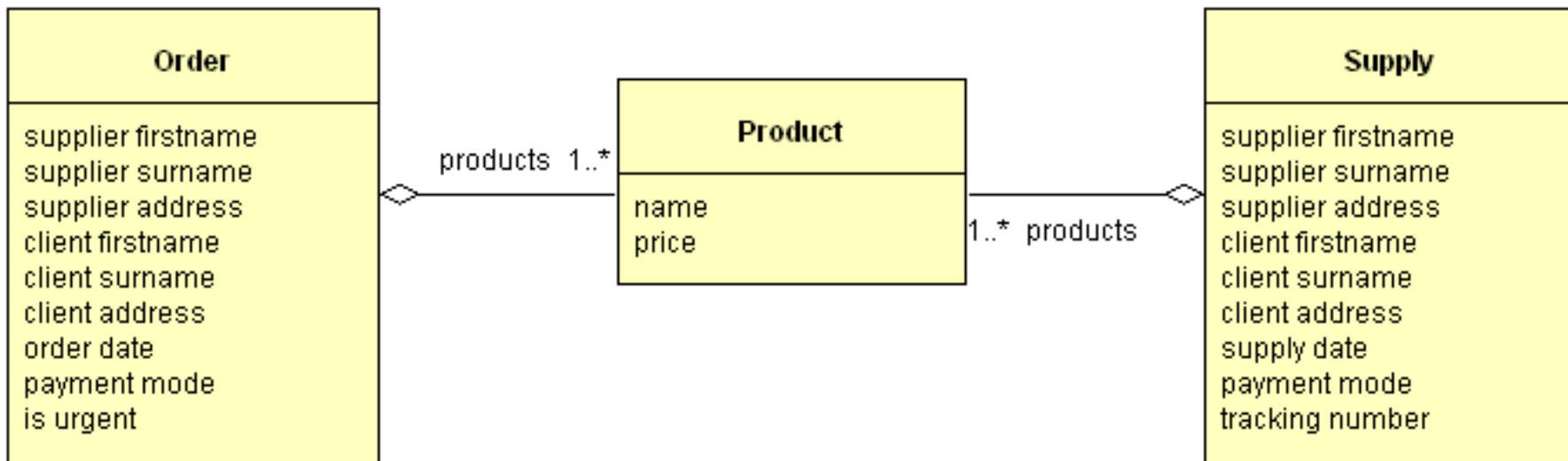
# First Normal Form Example

before normalization

Order	Supply
supplier firstname	supplier firstname
supplier surname	supplier surname
supplier address	supplier address
client firstname	client firstname
client surname	client surname
client address	client address
order date	supply date
payment mode	payment mode
is urgent	tracking number
first product name	first product name
first product price	first product price
second product name	second product name
second product price	second product price
third product name	third product name
...	...

# First Normal Form Example

after normalization



# Second Normal Form

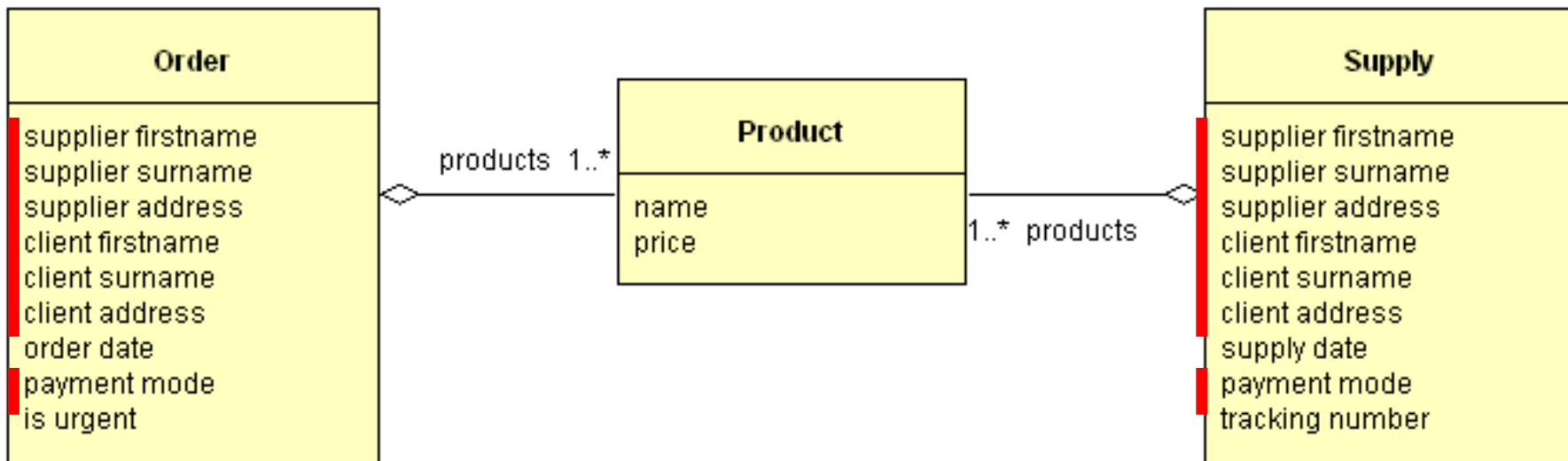
**Definition 2.** *A class is in the second object normal form (2ONF) when it is in 1ONF and when its objects do not contain attribute or group of attributes, which are shared with another object. Shared attributes must be extracted into new objects of a new class, and in all objects, where they appeared, must be replaced by the link to the object of the new class. An object schema is in 2ONF when all of its classes are in 2ONF.*

More formally; Let us have two objects  $a, b \in \Omega$  for  $k > 1$  (length of a collection of shared attributes) as  $data(a) = [\dots, x_1, \dots, x_k, \dots]$  and  $data(b) = [\dots, y_1, \dots, y_k, \dots]$  having  $\forall i \in (1, \dots, k) : x_i = y_i$ .

Then it is required to modify objects  $a$  and  $b$  and create new object  $c \in \Omega$  as  $data(a) = [\dots, c, \dots]$  and  $data(b) = [\dots, c, \dots]$  and  $data(c) = [x_1, \dots, x_k] = [y_1, \dots, y_k]$ .

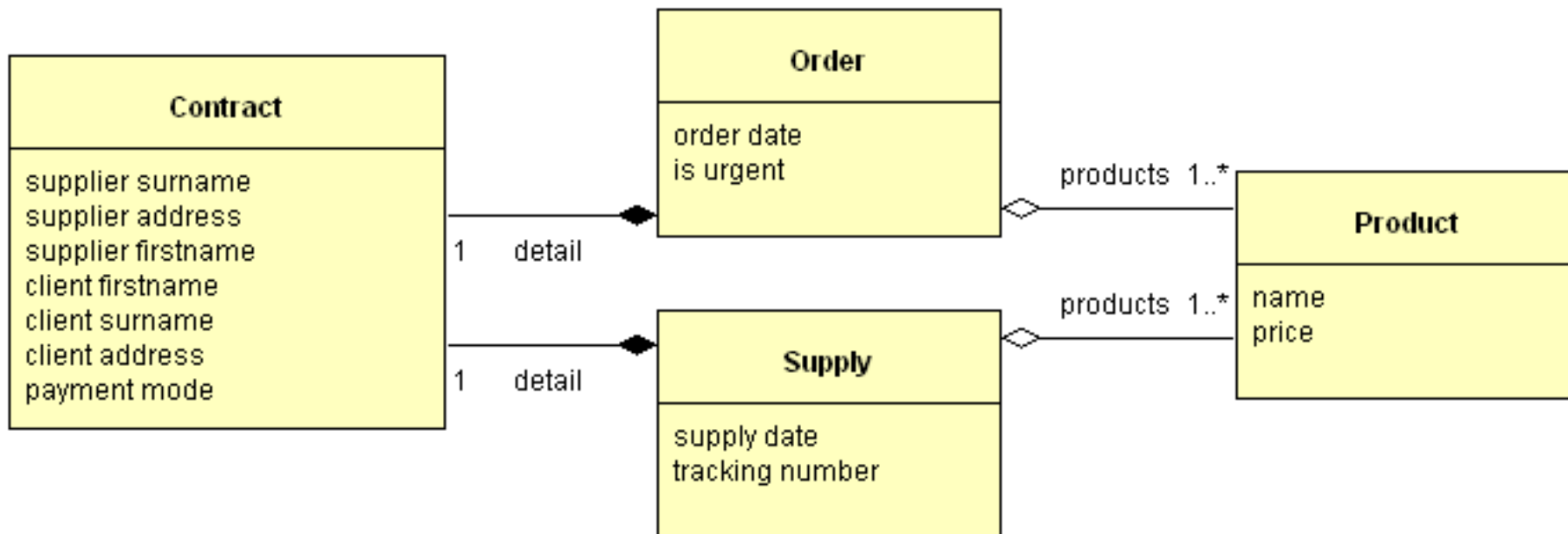
# Second Normal Form Example

before normalization



# Second Normal Form Example

after normalization



# Third Normal Form

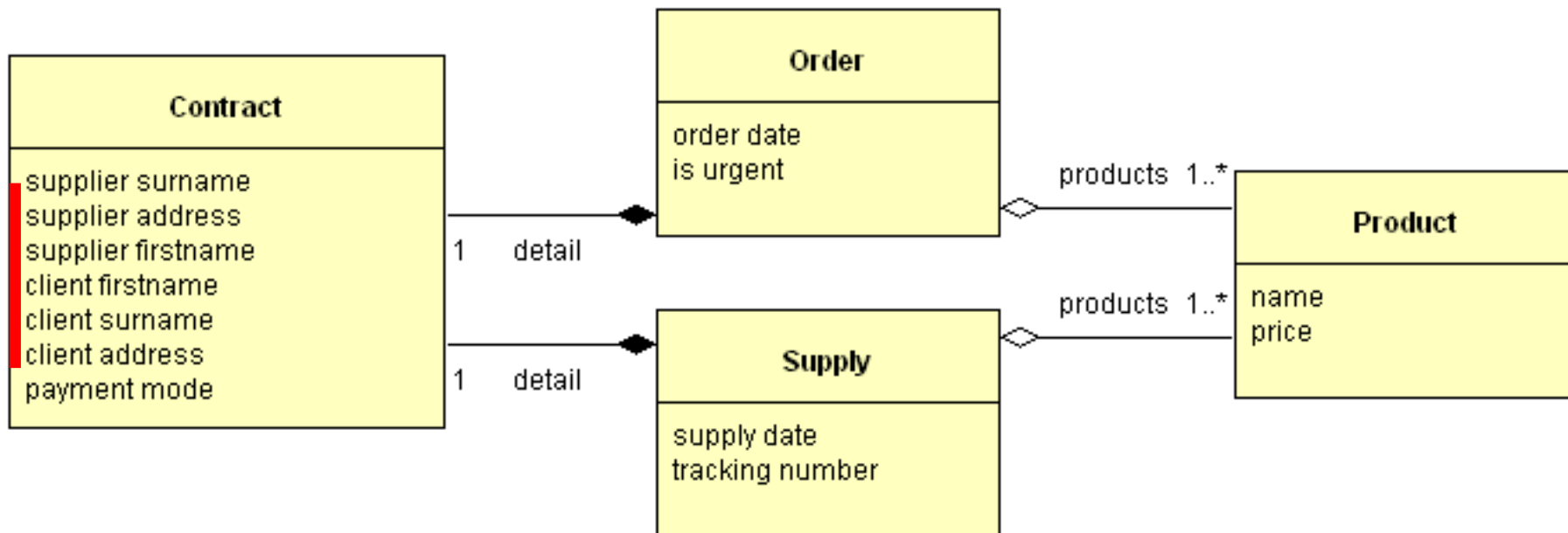
**Definition 3.** *A class is in the third object normal form (3ONF) when it is in 2ONF and when its objects do not contain attribute or group of attributes, which have the independent interpretation in the modeled system. These attributes must be extracted into objects of a new class and in objects, where they appeared, must be replaced by the link to this new object. An object schema is in 3ONF when all of its classes are in 3ONF.*

More formally; Let us have an object  $a \in \Omega$  for  $k > 1$  (length of a collection of independent attributes) having  $data(a) = [\dots, x_1, \dots, x_k, \dots]$ , where  $[x_1, \dots, x_k]$  is collection of independent attributes.

Then it is required to create new object  $b \in \Omega$  and modify object  $a$  as  $data(a) = [\dots, b, \dots]$  and  $data(b) = [x_1, \dots, x_k]$ .

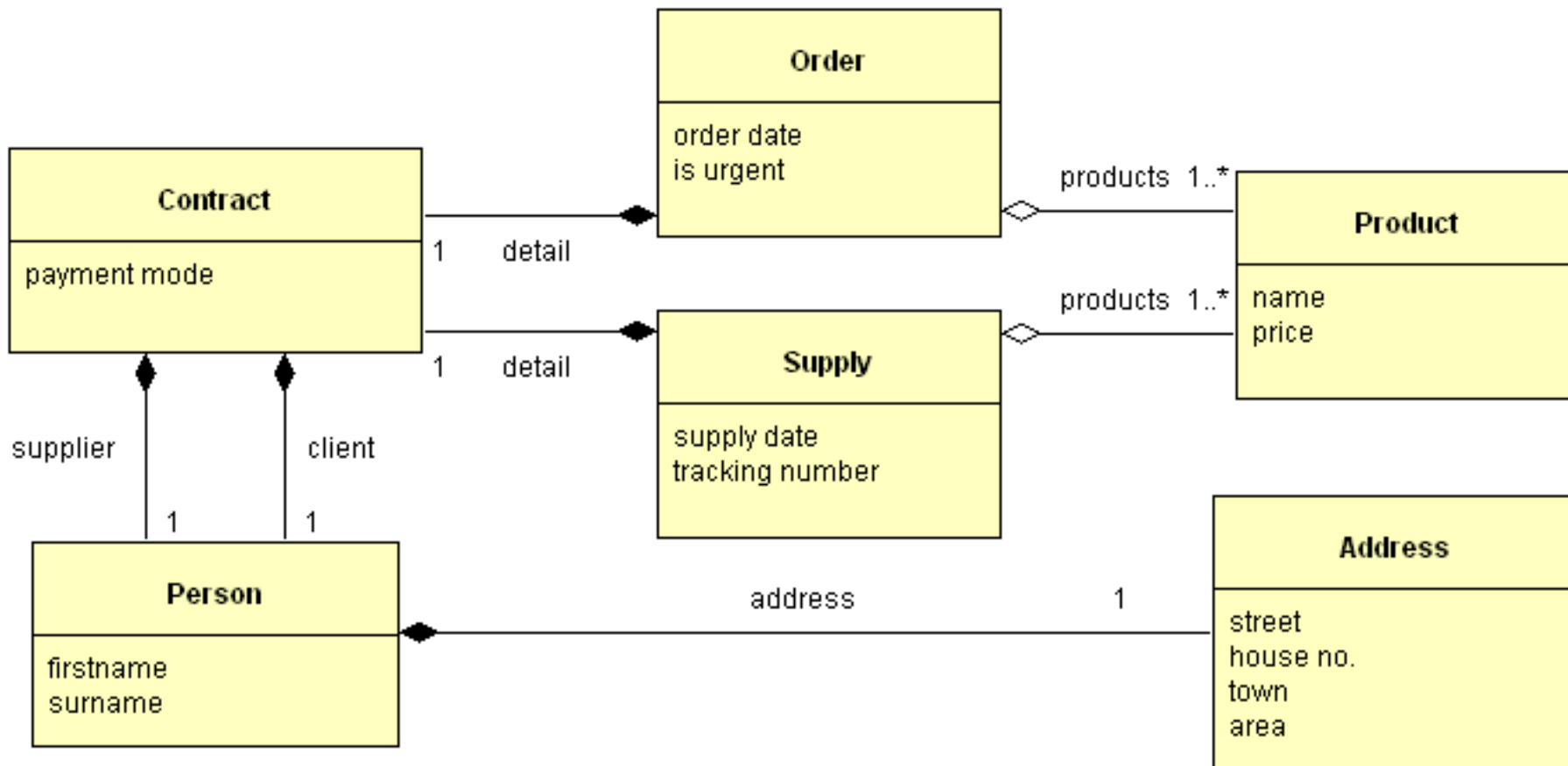
# Third Normal Form Example

before normalization



# Third Normal Form Example

after normalization



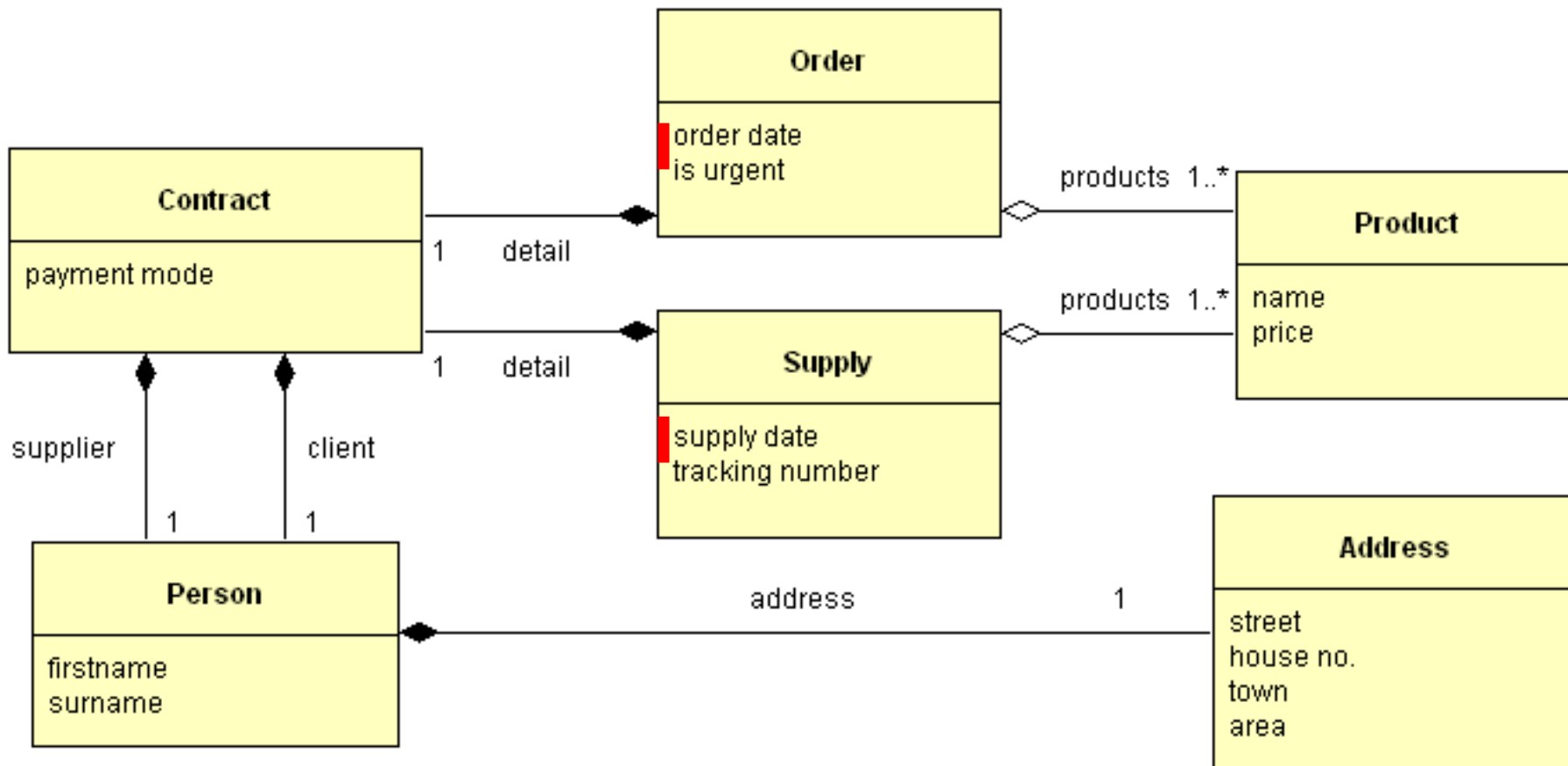
# Fourth Normal Form

**Definition 4.** *A class is in the fourth object normal form (4ONF) when it is in 3ONF and when there is no other class in the system, which defines the same attributes. These attributes must be extracted from classes, where they are duplicated, and affected classes must be connected using class inheritance in order to exclude data definition duplicates. If there is no existing class to be reused as a inheritance superclass, a new superclass must be added into the system. An object schema is in 4ONF when all of its classes are in 4ONF.*

More formally; For each two objects  $a, b$  in the object system  $\Omega$  as  $a, b \in \Omega$  having  $data(a) = [x_1, \dots, x_k]$  and  $data(b) = [\dots, y_1, \dots, y_k, \dots]$  where  $\forall i \in (1, \dots, k) : class(x_i) = class(y_i)$ , classes of these objects  $a, b$  must have inheritance relationship as  $class(a) \prec class(b)$  in order to avoid duplicates.

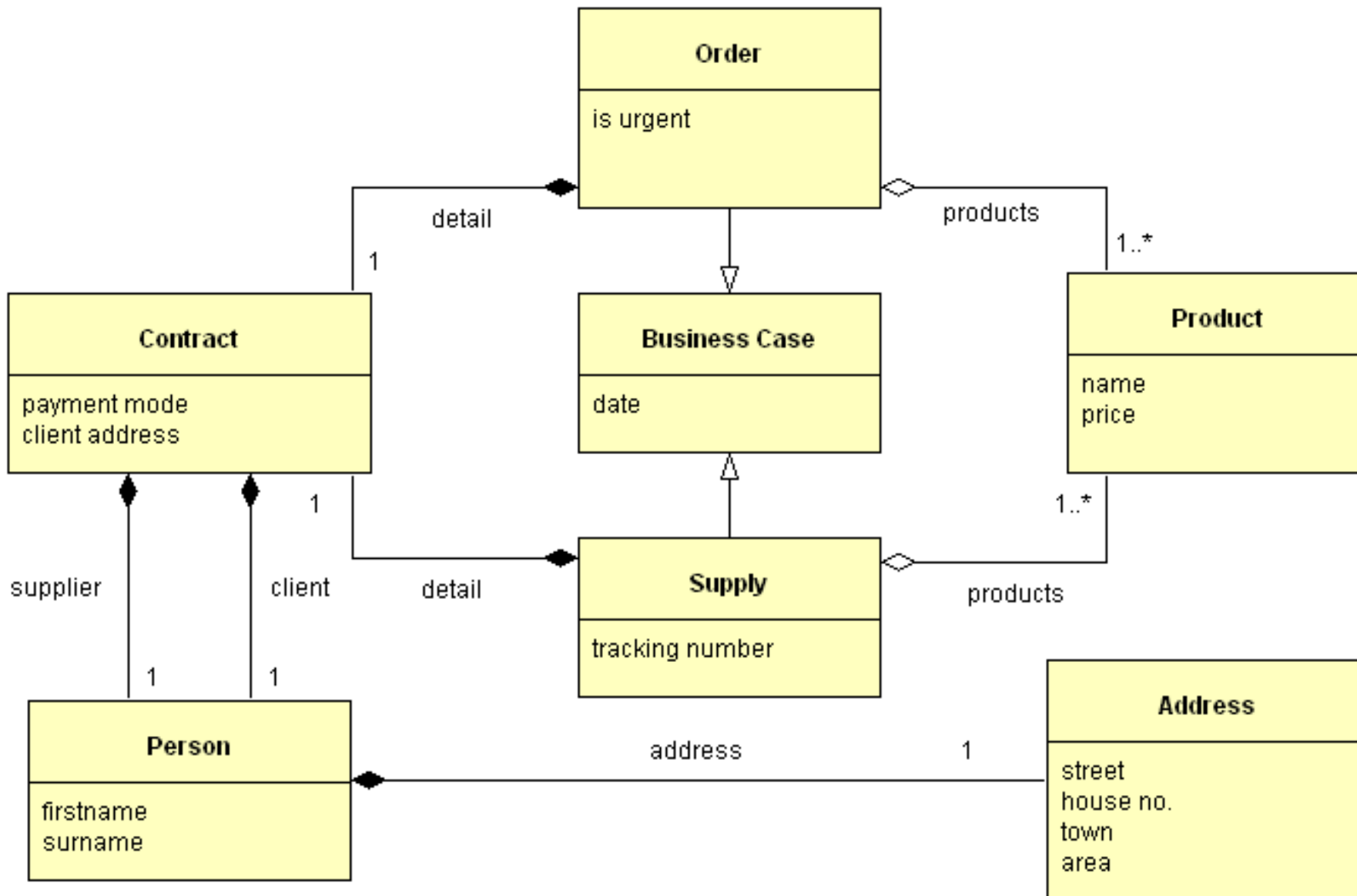
# Fourth Normal Form Example

before normalization

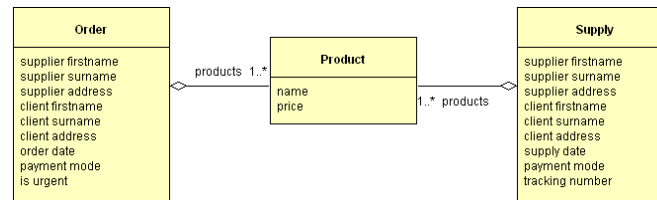


# Fourth Normal Form Example

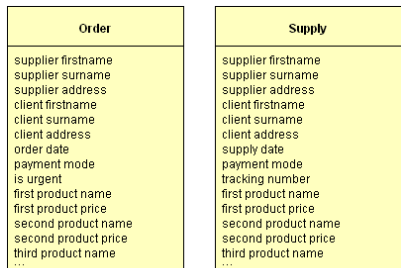
after normalization



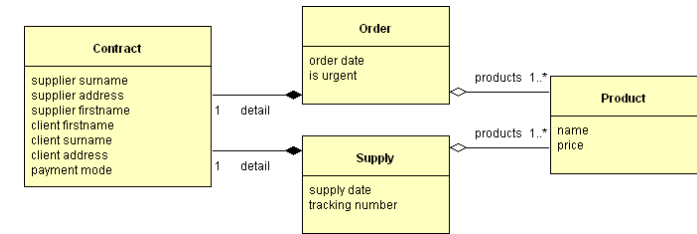
# Revision



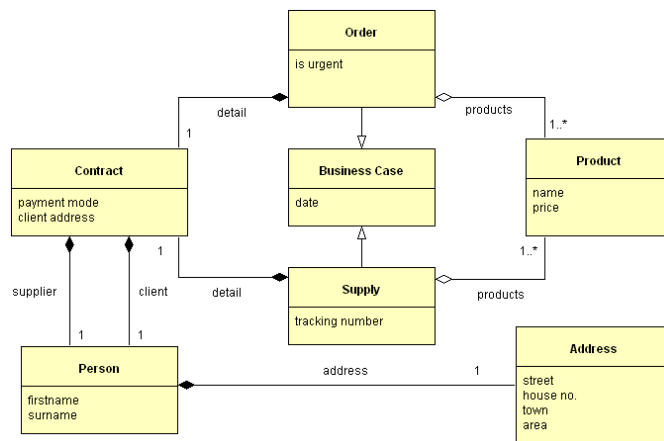
1 OONF



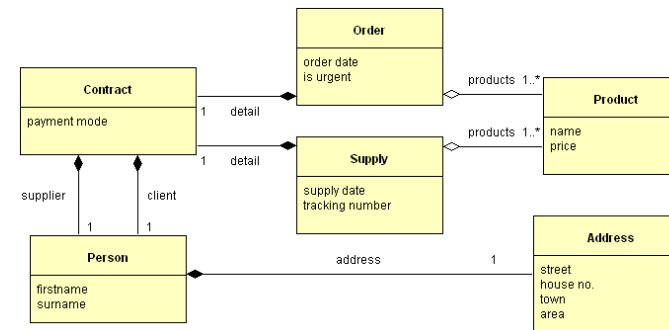
no OONF



2 OONF



4 OONF



3 OONF

# Conclusion

It is a pity, that so perspective and practically used technology - the object-oriented approach - still does not have comprehensible and universally accepted theoretical foundation and formal techniques. It is known, that several research centers are interested in this theme, but any coherent and widely accepted results were not yet published in recent years. Absence of reputable formal tools and techniques is the big problem of this promising technology. Therefore we suppose that near future may bring maybe some alternative approaches, more or less similar to our approach we presented here.

Our future research will focus on formal describing the rules of our object-oriented normal forms as a sequence of refactoring steps.