

Object-oriented class normalisation from a conceptual modelling perspective

Vojtěch Merunka, Himesha Wijekoon, Boris Shegolev

Department of Information Engineering
Faculty of Economics and Management
Czech University of Life Sciences in Prague

Agenda

- introduction, motivation, symmetries
- Ambler's 1st OONF, 2nd OONF, 3rd OONF on the data level
- next new normal forms on the metadata level
- discussion and conclusion

Introduction

Software engineers and practitioners are often confronted with the question “How to design the best structure of classes?”


In the past, by the enthusiasm of the benefits of object-oriented programming, it was widespread that the object-oriented computation model automatically supports the right design, so programmers don't need much of formal techniques.

Existing O-O normalisation approaches

Software engineering community knows a few different approaches for the object-oriented class normalisation, but one can find some common ground.


The first three object-oriented normal forms are more or less similar in most cases. Significant differences occur only in higher object-oriented normal forms, if they are defined at all. **Finding a unifying view of this issue would certainly be of great benefit.**

Our idea on an Electronic Flight Ticket Example



e-Ticket Receipt & Itinerary

Scan the barcode or use the e-ticket number below for self service kiosk check-in



1762175501628

Your electronic ticket is stored in our computer reservations system. This e-Ticket receipt/itinerary is your record of your electronic ticket and forms part of your contract of carriage. You may need to show this receipt to enter the airport and/or to prove return or onward travel to customs and immigration officials.

Your attention is drawn to the Conditions of Contract and Other Important Notices set out in the attached document. Please visit us on www.emirates.com to check-in online and for more information.

Emirates' check-in counters open no less than three hours before the flight. You should check in no later than 90 minutes before departure. Boarding starts 45 minutes before your flight and gates close 20 minutes before departure. If you report late we will not be able to accept you for travel.

Please check with departure airport for restrictions on the carriage of liquids, aerosols and gels in hand baggage.

Below are the details of your electronic ticket. Note: all timings are local.

PASSENGER AND TICKET INFORMATION					
PASSENGER NAME	BINNORDDIN NORHIZAM				
BOOKING REFERENCE	MZXTIF				
E-TICKET NUMBER	176 2175501628				
ISSUED BY / DATE	DUBAI / EMIRATES EZM 10SEP2013EKBOMP0				

TRAVEL INFORMATION					
FLIGHT	DEPART/ARRIVE	AIRPORT/TERMINAL	CHECK-IN OPENS	CLASS	COUPON VALIDITY
EK 345 CONFIRMED	13 OCT 13 1030	KUALA LUMPUR (KUL) TERMINAL 0	13 OCT 13 0730	ECONOMY	NOT AFTER 13 JUL 14 27 K
	13 OCT 13 1330	DUBAI INTNL (DXB) TERMINAL 3		BAGGAGE ALLOWANCE 30KGS	
EK 923 CONFIRMED	13 OCT 13 1510	DUBAI INTNL (DXB) TERMINAL 3	13 OCT 13 1210	ECONOMY	NOT AFTER 13 JUL 14 24 A
	13 OCT 13 1700	CAIRO (CAI) TERMINAL 1		BAGGAGE ALLOWANCE 30KGS	

FARE AND ADDITIONAL INFORMATION		
FARE	EGP3342.00	ADDITIONAL INFORMATION
TAXES/FEES/CHARGES	MYR2.0QH MYR47.0YR PD100.0EG PD150.0XK PD11.0XL PD137.0QH PD7.0EQ PD50.0JK PD148.0MY	* NONREF NON-END/SKYWARDS SAVER/VLD ON EK/PENALTIES APPLY
TOTAL	MYR49A	
FORM OF PAYMENT	CREDIT CARD	
*AT CHECK-IN YOU MAY NEED TO PRESENT THE CREDIT CARD USED FOR PAYMENT OF THE TICKET		

FARE CALCULATIONS
CAI EK X/DXB Q55.00EK KUL 192.53UEE1 YEG1/EGH3 EK X/ ND ROE6.74963

©2012 Emirates. All rights reserved.
Page 1 of 1

no OONF

Electronic Ticket
+booking ref
+issue date
+ticket nr
+traveller name
+traveller category
+traveller passport nr
+traveller phone
+traveller e-mail
+total fare
+agency name
+agency phone
+agency e-mail
+1st flight nr
+1st flight class
+1st flight seat nr
+1st flight meal
+1st flight operator name
+1st flight equipment name
+1st flight departure airport name
+1st flight departure datetime
+1st flight arrival airport name
+1st flight arrival datetime
+2nd flight nr
...
+3rd flight nr
...
+nth flight nr
...

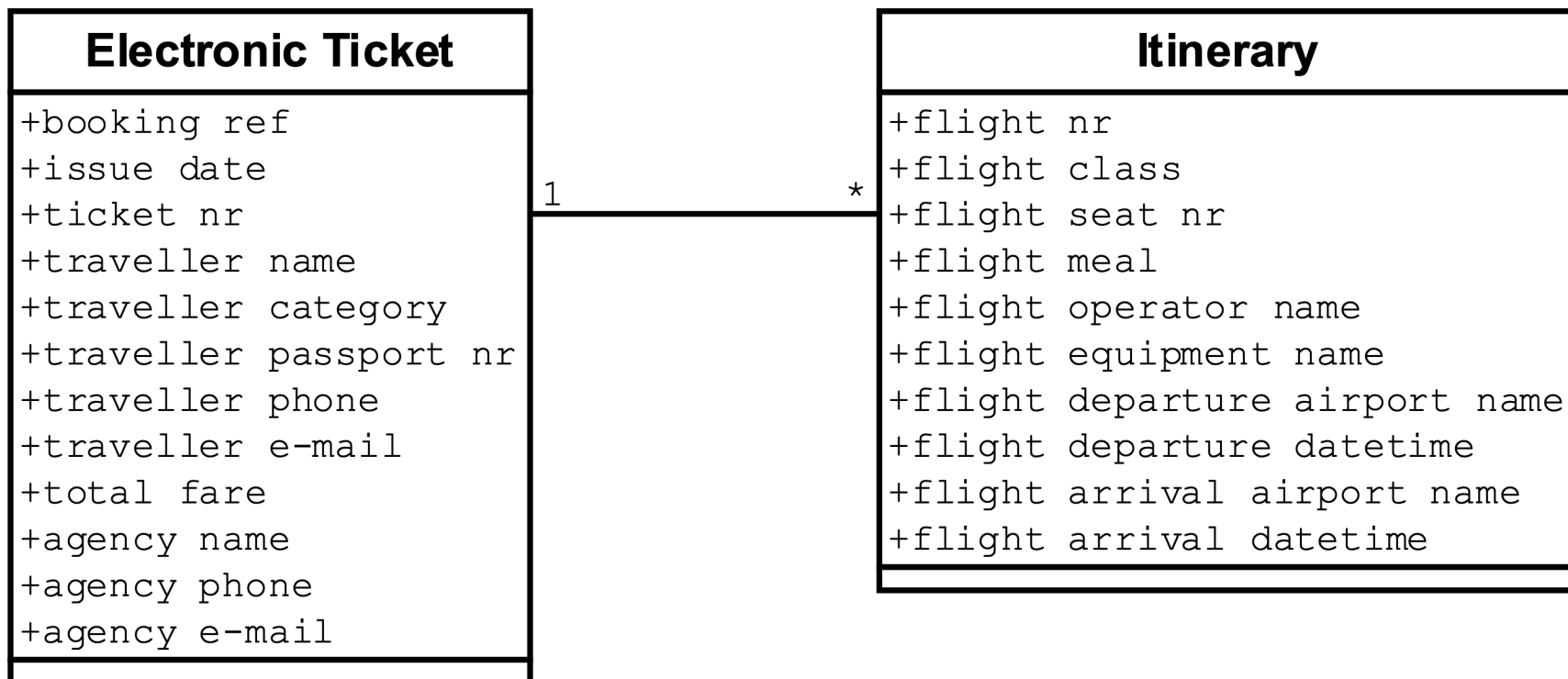
1st OONF - multivalued

Rule 1. *A class is in the first object-oriented normal form (1stOONF) 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 to the collection of the new objects. An object schema is in the 1stOONF when all of its classes are in the 1stOONF.*

Definition 1. Let us have an object a , where for $k \geq 1$ (length of collections of repeating attributes) and $n > 1$ (number of repetitions of these repeating collections) is $data(a) = [\dots, x_1^1, \dots, x_1^k, \dots, x_n^1, \dots, x_n^k, \dots]$, having $\forall i \in (1, \dots, k): domain(x_1^i) = domain(x_2^i) = \dots = domain(x_n^i)$. Then it is required to modify the object a and create a collection of new objects $\{b_j\}$ for $j \in (1, \dots, n)$ as $data(a) = [\dots, \{b_j\}, \dots]$ and $data(b_j) = [x_j^1, \dots, x_j^k]$.

no OONF

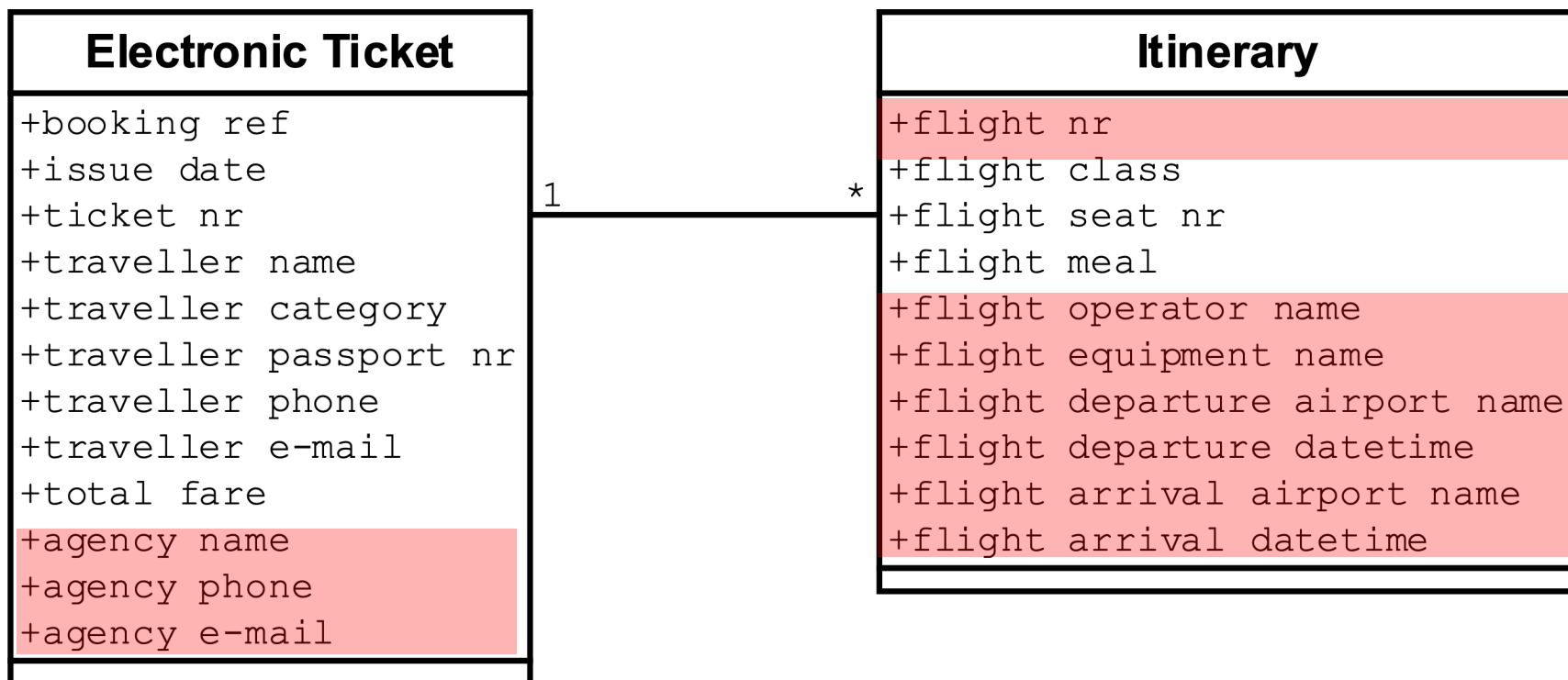
Electronic Ticket
+booking ref
+issue date
+ticket nr
+traveller name
+traveller category
+traveller passport nr
+traveller phone
+traveller e-mail
+total fare
+agency name
+agency phone
+agency e-mail
+1st flight nr
+1st flight class
+1st flight seat nr
+1st flight meal
+1st flight operator name
+1st flight equipment name
+1st flight departure airport name
+1st flight departure datetime
+1st flight arrival airport name
+1st flight arrival datetime
+2nd flight nr
...
+3rd flight nr
...
+nth flight nr
...

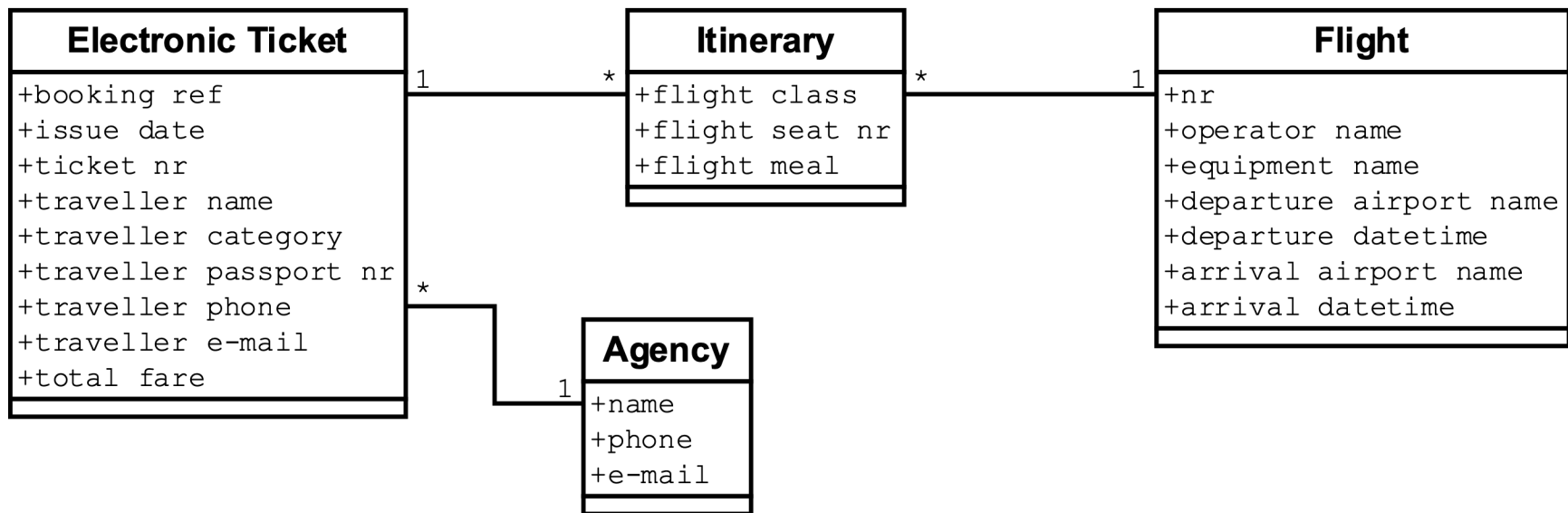
1st OONF

2nd OONF - shared values

Rule 2. *A class is in the second object-oriented normal form (2ndOONF) when it is in the 1stOONF and when its objects do not contain an attribute or a group of attributes, which is 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 the 2ndOONF when all of its classes are in the 2ndOONF.*

Definition 2. Let us have two objects a, b for $k \geq 1$ (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 $\forall i \in (1, \dots, k): x_i \equiv y_i$. Then it is required to modify objects a, b and to create new object c as $data(c) = [x_1, \dots, x_k] = [y_1, \dots, y_k]$ and $data(a) = [\dots, c, \dots]$ and $data(b) = [\dots, c, \dots]$.

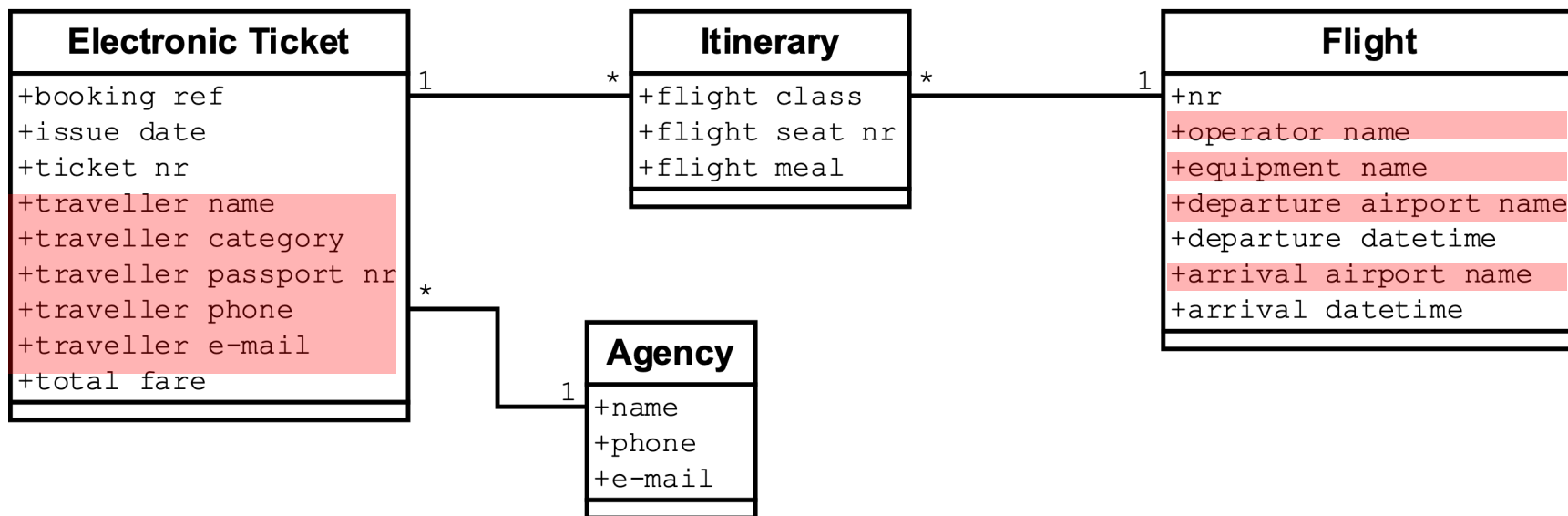
1st OONF

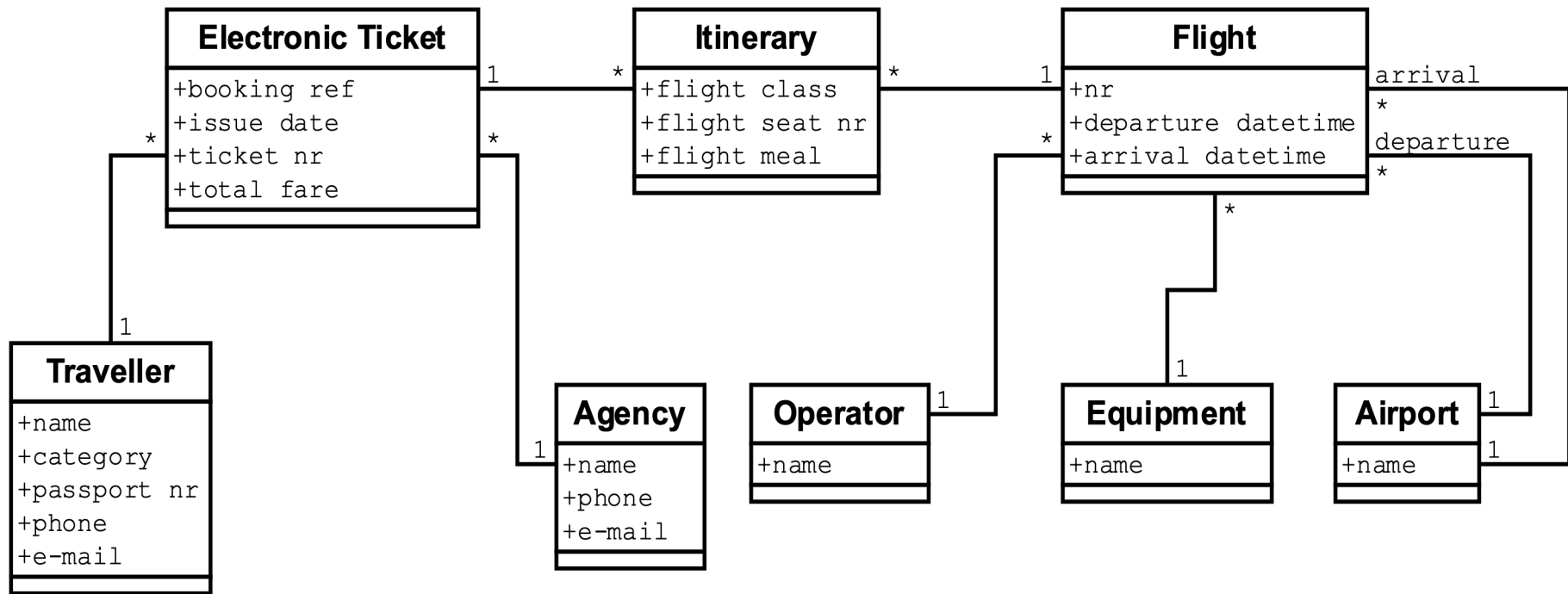
2nd OONF

3rd OONF – independent values

Rule 3. *A class is in the third object-oriented normal form (3rdOONF) when it is in the 2ndOONF and when its objects do not contain an attribute or a group of attributes, which has an independent interpretation in the modelled system. The independent attributes must be extracted into object of a new class and in objects, where they originally appeared, must be replaced by the link to this new object. An object schema is in the 3rdOONF when all of its classes are in the 3rdOONF.*

Definition 3. Let us have an object a for $k \geq 1$ (length of a collection of independent attributes) having $data(a) = [\dots, x_1, \dots, x_k, \dots]$, where $[x_1, \dots, x_k]$ is a collection of independent attributes. Then it is required to create a new object b and modify the object a as $data(a) = [\dots, b, \dots]$ and $data(b) = [x_1, \dots, x_k]$.

2nd OONF

3rd OONF

New normal forms

Our basic idea is as follows:

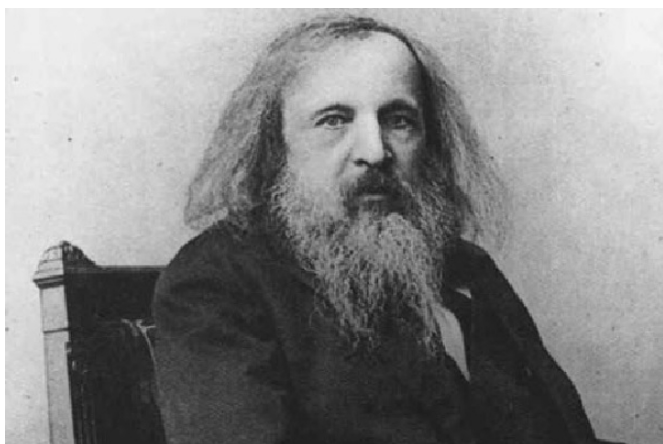
Existing three rules for object-oriented normal forms relate to the properties of attribute data values and result in the *composing structure (has-a hierarchy)* of classes.

We could use the same rules once more but on the level of attribute data types (e.g. one meta-level higher) and obtain the result in the *inheritance and inheritance-like structures (is-a hierarchy)* of classes.

The idea behind - Symmetries

Symmetries and analogies are maybe the most essential principles of how God is building the world and deserves our special attention. For example, the Russian chemist Dmitri Mendeleev did the same when he published the first widely recognised periodic table of chemical elements in 1869. He developed his periodic table to illustrate repeating properties of the then-known elements, and he also predicted some properties of then-unknown elements that would be expected to fill gaps in this table. Most of his predictions were proven correct when the new chemical elements were subsequently discovered.

Analogically, we expected similar effect if the object-oriented normalisation rules were systematised in a similar way.



Group	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18														
Period 1	1 H 1.00794																		2 He 4.0026													
2	3 Li 6.941	4 Be 9.01218																5 B 10.811	6 C 12.0107	7 N 14.0067	8 O 15.9994	9 F 18.9984	10 Ne 20.1797									
3	11 Na 22.9897	12 Mg 24.305																13 Al 26.9815	14 Si 28.0855	15 P 30.9737	16 S 32.06	17 Cl 35.4527	18 Ar 39.948									
4	19 K 39.0983	20 Ca 40.078	21 Sc 44.9559	22 Ti 47.887	23 V 50.9415	24 Cr 51.9961	25 Mn 54.938	26 Fe 55.845	27 Co 58.9332	28 Ni 58.6934	29 Cu 63.546	30 Zn 65.38	31 Ga 69.723	32 Ge 72.61	33 As 74.9218	34 Se 78.96	35 Br 79.904	36 Kr 83.798														
5	37 Rb 85.4678	38 Sr 87.62	39 Y 88.9058	40 Zr 91.224	41 Nb 92.9063	42 Mo 95.94	43 Tc 98	44 Ru 101.07	45 Rh 101.905	46 Pd 106.42	47 Ag 107.868	48 Cd 112.411	49 In 114.818	50 Sn 118.71	51 Sb 121.76	52 Te 127.6	53 I 126.905	54 Xe 131.293														
6	55 Cs 132.905	56 Ba 137.327	57 ** La 138.905	58 ** Ce 137.49	59 ** Pr 140.908	60 ** Nd 144.24	61 ** Pm 145	62 ** Sm 150.36	63 ** Eu 151.964	64 ** Gd 157.25	65 ** Tb 158.925	66 ** Dy 162.5	67 ** Ho 164.93	68 ** Er 167.26	69 ** Tm 168.934	70 ** Yb 173.054	71 ** Lu 174.967	72 Hf 178.49	73 Ta 180.948	74 W 183.84	75 Re 186.207	76 Os 190.23	77 Ir 192.217	78 Pt 195.078	79 Au 196.966	80 Hg 200.59	81 Tl 204.383	82 Pb 207.2	83 Bi 208.98	84 Po 209	85 At 210	86 Rn 222
7	87 Fr 223	88 Ra 226	89 ** Ac 227	90 ** Th 232.038	91 ** Pa 231.036	92 ** U 238.028	93 ** Np 237.048	94 ** Pu 244	95 ** Am 243	96 ** Cm 247	97 ** Bk 247	98 ** Cf 251	99 ** Es 252	100 Fm 257	101 Md 258	102 No 259	103 Lr 260	104 Nh 285	105 Fl 289	106 Mc 289	107 Lv 289	108 Ts 289	109 Og 294									
*Lanthanides																			58 Ce 140.116	59 Pr 140.907	60 Nd 144.24	61 Pm 145	62 Sm 150.36	63 Eu 151.964	64 Gd 157.25	65 Tb 158.925	66 Dy 162.5	67 Ho 164.93	68 Er 167.26	69 Tm 168.934	70 Yb 173.054	71 Lu 174.967
**Actinides																			90 Th 232.038	91 Pa 231.036	92 U 238.028	93 Np 237.048	94 Pu 244	95 Am 243	96 Cm 247	97 Bk 247	98 Cf 251	99 Es 252	100 Fm 257	101 Md 258	102 No 259	103 Lr 260

Class inheritance

Of course, *composing* object classes is a significant building element for the object-oriented approach, but equally important is class *inheritance*. Although the presence of inheritance between object classes is not a prerequisite for an object-oriented model of calculation, because there exist object-oriented systems without classes or without inheritance, but class-based object-oriented systems with the inheritance constitute a de-facto standard.

It is, therefore, striking that the development of formal class design techniques does not address the inheritance sufficiently.

Class mixins

In object-oriented programming languages, a *mixin* is a structure similar to a class that contains methods for use by other classes without having to be the parent class of those other classes. *Mixins* are *traits* (e.g. sets of independent methods that can be used to extend the functionality of objects) which are used to compose classes. *Mixins* usage is sometimes described as being "included" rather than "inherited".

At the conceptual level, it can be said that *mixins* are technical tools for better software implementation of the *decorator* design pattern.

Solution

<p>data types level attribute data types INHERITANCE</p>	<p>4th OONF ?</p>	<p>5th OONF sharing metadata in subclasses</p>	<p>6th OONF independent metadata in mixins</p>
<p>data values level attribute data values COMPOSING</p>	<p>1st OONF multivalued within objects</p>	<p>2nd OONF sharing data values within objects</p>	<p>3rd OONF independent data values within objects</p>

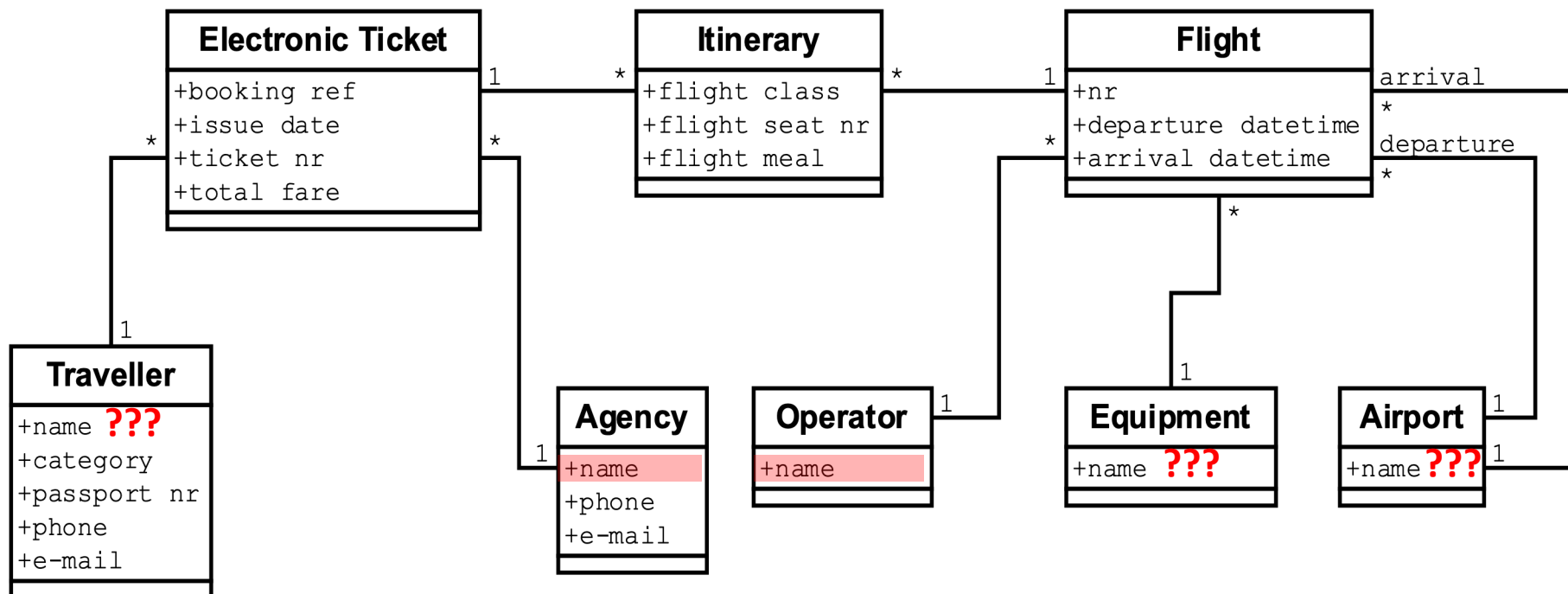
Solution - alternative

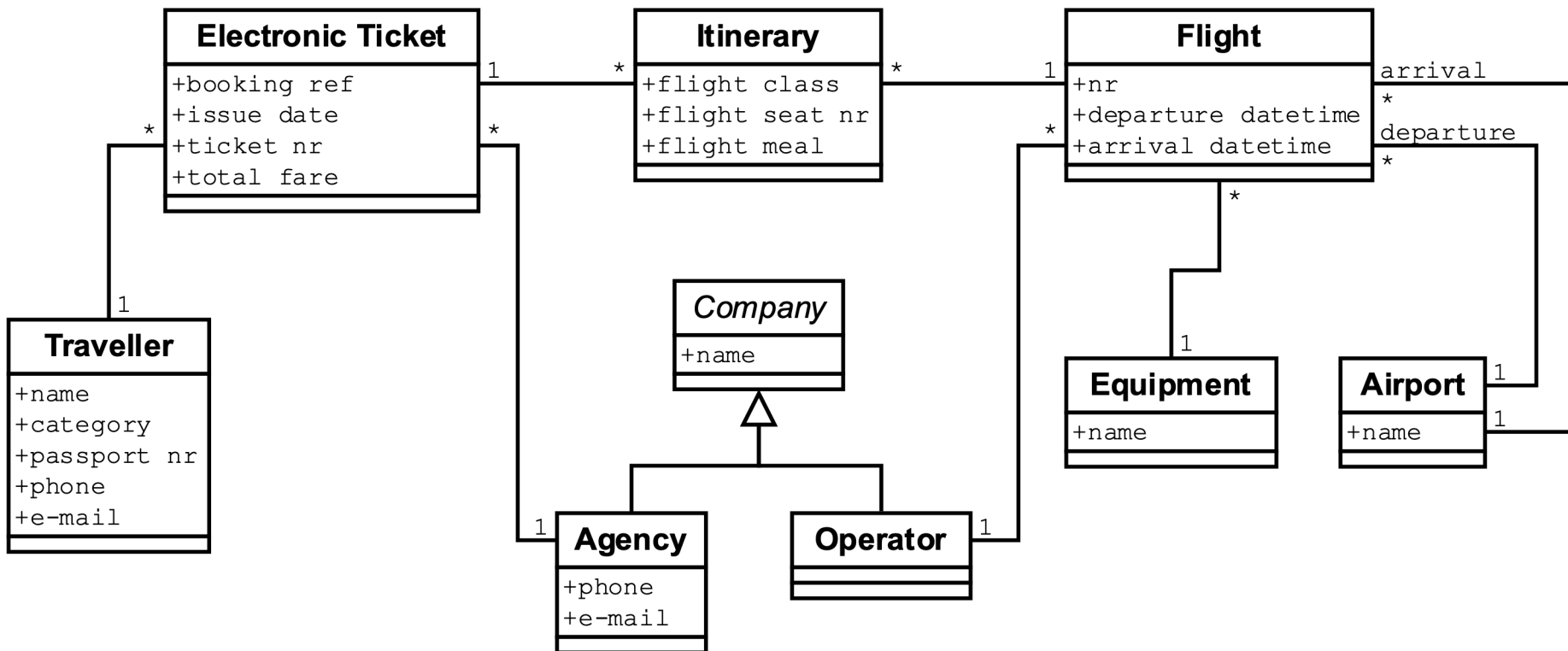
<p>data types level attribute data types INHERITANCE</p>	<p>1st OONF multivalued within objects</p>	<p>4th OONF sharing metadata in subclasses</p>	<p>5th OONF independent metadata in mixins</p>
<p>data values level attribute data values COMPOSING</p>		<p>2nd OONF sharing data values within objects</p>	<p>3rd OONF independent data values within objects</p>

4th OONF - shared data types

Rule 4. *A class is in the fourth object-oriented normal form (4thOONF) when it is in the 3rdOONF and when its objects do not contain an attribute or a group of attributes, whose attribute types are shared with another object. Shared attribute types must be extracted into new class linked as an inheritance superclass of all classes of objects, where they originally appeared. An object schema is in the 4thOONF when all of its classes are in the 4thOONF.*

Definition 4. Let us have two classes a, b for $k \geq 1$ (length of a collection of shared attribute types) as $datatypes(a) = [\dots, x_1, \dots, x_k, \dots]$ and $datatypes(b) = [\dots, y_1, \dots, y_k, \dots]$ having $\forall i \in (1, \dots, k): x_i \equiv y_i$. Then it is required to modify classes a, b and to create new class c as $datatypes(c) = [x_1, \dots, x_k] = [y_1, \dots, y_k]$ and $superclass(a) = c$ and $superclass(b) = c$. (otherwise $c < a$ and $c < b$).

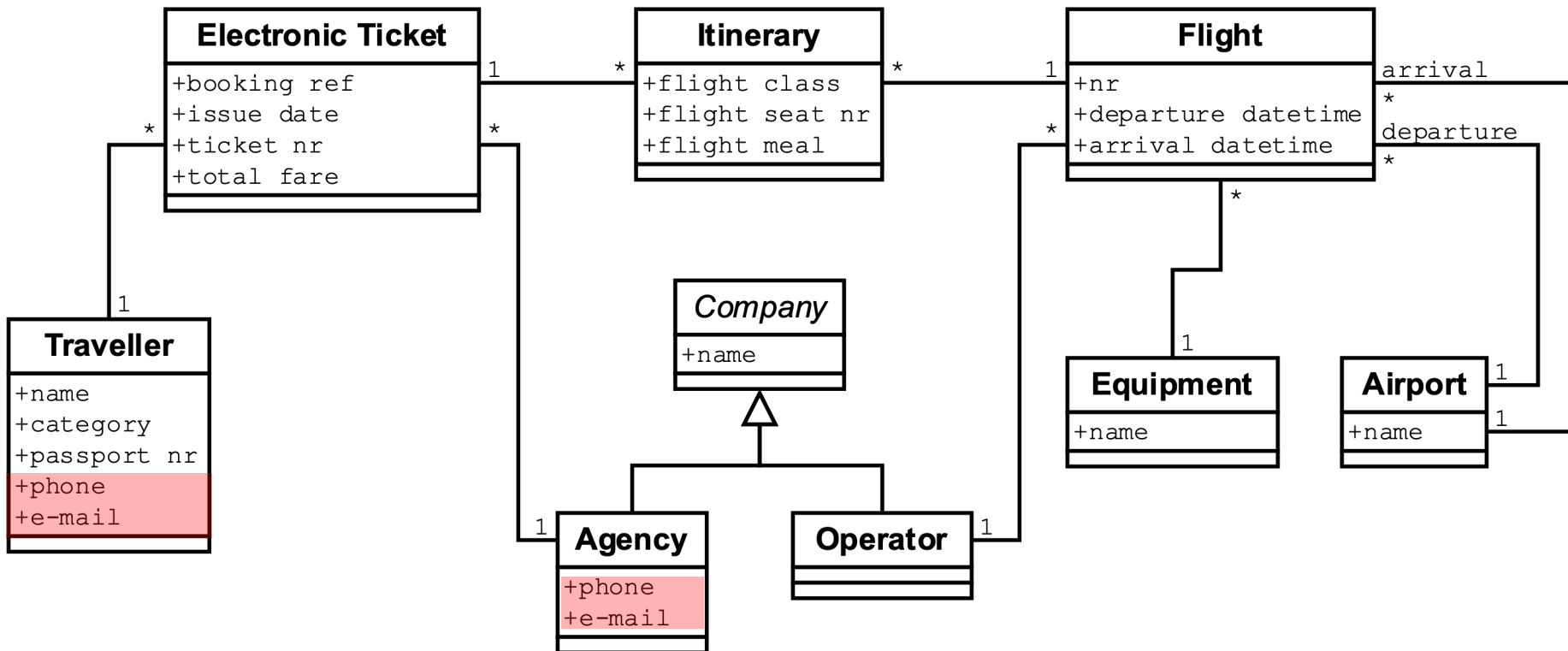
3rd OONF

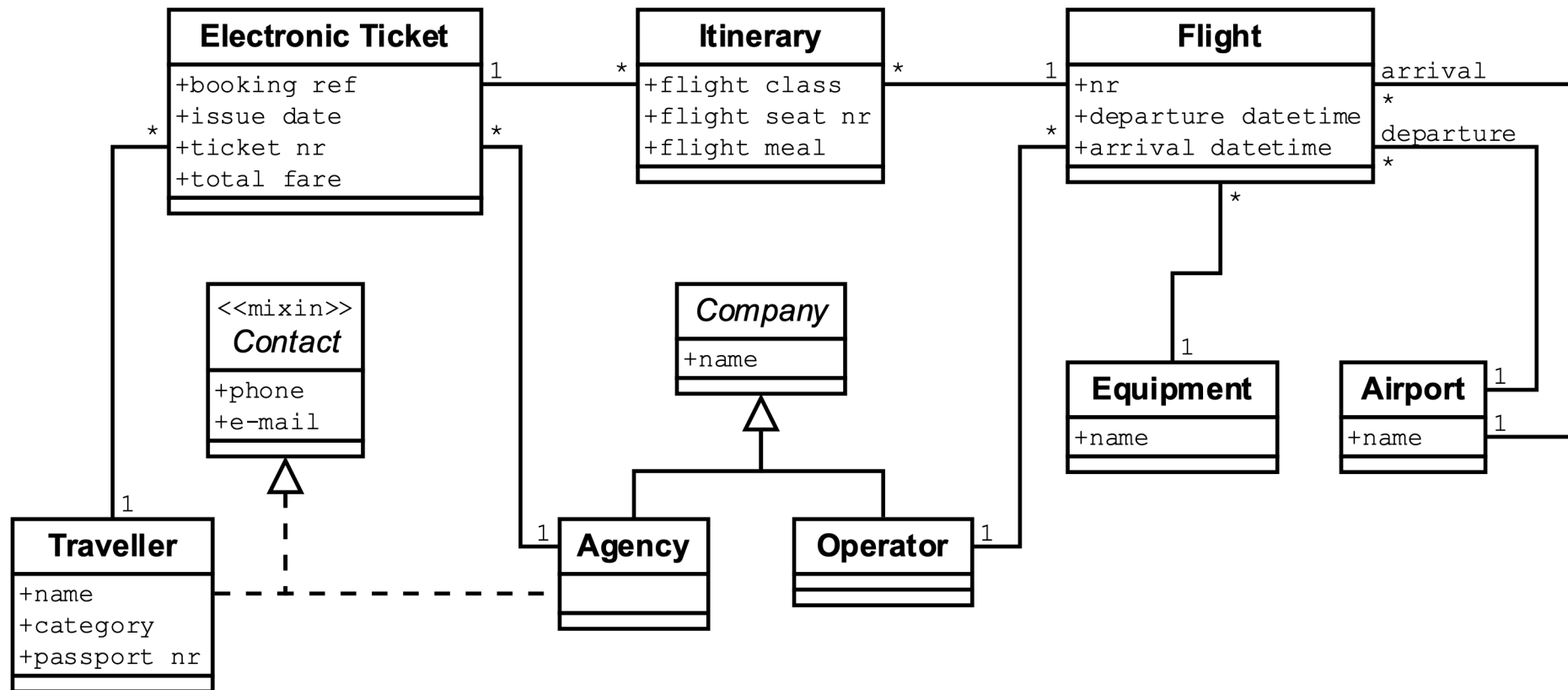
4th OONF

5th OONF - independent data types

Rule 5. *A class is in the fifth object-oriented normal form (5thOONF) when it is in the 4thOONF and when do not contain an attribute type or a group of attribute types, which has an independent interpretation in the modelled system. These attribute types must be extracted into a new mixin and in classes where they originally appeared, must be replaced by the link to this new mixin. An object schema is in the 5thOONF when all of its classes are in the 5thOONF.*

Definition 5. Let us have a class a for $k \geq 1$ (length of a collection of independent attribute types) having $datatypes(a) = [\dots, x_1, \dots, x_k, \dots]$, where $[x_1, \dots, x_k]$ is a collection of independent attribute types. Then it is required to create a new *mixin* b as $datatypes(b) = [x_1, \dots, x_k]$ and add *mixin* b to the class a .

4th OONF

5th OONF

conclusion

1. We added inheritance to the object-oriented normalisation rules.
2. We showed new conceptual connections (analogy, symmetry) between the object composing and the object inheritance.
3. We found a conceptual-modeling reason of inheritance-like programming constructs mixin (or trait).
4. We have confirmed the previously known idea that behavioural design patterns can be replaced by direct semantic constructs in some programming languages. Either our language allows mixins or we need to use a decorator design pattern, but at the conceptual level, it is the same thing.