

# Object-Role Modeling – wprowadzenie do notacji

Maciej Sobczak

## 1 Wstęp

*Object-Role Modeling (ORM)* to język do analizy koncepcyjnej systemu informacyjnego, ukierunkowany na modelowanie statycznych zależności pomiędzy danymi.

Początki notacji ORM to język NIAM opracowany w połowie lat '70 w Europie. Późniejsze prace prowadzone w Australii i USA doprowadziły do powstania całej rodziny języków (ale różniących się bardzo nieznacznie w porównaniu do różnic występujących w obrębie rodziny ER), nazywanych *Object-Role Modeling*. ORM, w postaci sformalizowanej i wspieranej przez popularne narzędzia CASE (np. Microsoft Visio) został opisany w [1] oraz w szeregu dokumentów dostępnych na [2].

Nazwa *Object-Role Modeling* pochodzi od postrzegania systemu jako zbioru obiektów (encji lub wartości), które odgrywają pewne role w relacjach z innymi obiektami. Słowo „rola” należy tutaj traktować w sposób statyczny – nie ma ono związku z niektórymi podejściami OO pozwalającymi na dynamiczną klasyfikację.

Ważną cechą notacji ORM jest fakt, że *relacja* jest jedyną konstrukcją służącą do przechowywania informacji. Nie istnieje pojęcie *atrybutu*, co odróżnia ORM od szeregu notacji ER oraz UML. Okazuje się, że brak atrybutów jako konstrukcji w języku nie jest ograniczeniem – wręcz przeciwnie. Pozwala to (a właściwie wymusza) na tworzenie modeli, które są zawsze konstrukcyjnie spójne i zwykle znacznie stabilniejsze od odpowiadających im diagramów ER czy UML w sytuacjach, kiedy w modelu danych pojawiają się zmiany.

ORM został stworzony z myślą o możliwości werbalizowania modeli danych do postaci zdań w języku naturalnym. Jest to bardzo ważna cecha, gdyż pozwala

na łatwą komunikację z osobami, które będąc ekspertami w danej dziedzinie nie muszą mieć wykształcenia informatycznego aby zrozumieć (i zweryfikować) dokument projektowy.

Zależność pomiędzy diagramami ORM i ich reprezentacjami w języku naturalnym jest tak silna, że transformacje w jedną i drugą stronę mogą być wykonywane automatycznie. Przedstawione tutaj werbalizacje pochodzą z programu Microsoft VisioModeler.

Dokument ten został przygotowany jako bardzo skrótowe wprowadzenie do języka ORM. W większości przypadków posługuję się terminologią angielską, gdyż wiele pojęć w ogóle nie ma polskich odpowiedników.

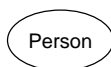
## 2 Podstawowe symbole

ORM wyróżnia dwa rodzaje obiektów: encje i wartości. Encje są obiektami odpowiadającymi przedmiotom ze świata rzeczywistego (w sensie modelowanej aplikacji) i zwykle wymagają określenia pewnego sposobu ich identyfikowania, podczas gdy wartości identyfikują same siebie i są traktowane tak, jak literały w językach programowania.

Warto pamiętać, że wszystkie obiekty, które pojawiają się na diagramie ORM wyznaczają *dziedziny semantyczne* modelu danych, czyli innymi słowy definiują typy danych. Dwa różne obiekty (dotyczy to zarówno encji jak i wartości) są zawsze niekompatybilne w tym sensie, że nie jest możliwe np. ich porównywanie.

Relacje pomiędzy obiektami odpowiadają nazwanym predykatom. Nie ma ograniczeń na ilość predykatów, co pozwala na użycie w diagramie relacji dowolnego rodzaju, łącznie z unarnymi.

### 2.1 Entity



Rysunek 1: Entity (encja)

Rysunek 1 przedstawia symbol encji. Istnieją pewne warianty tego symbolu, o których wspomnę później.

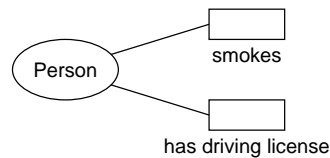
## 2.2 Value



Rysunek 2: Value (wartość)

Rysunek 2 przedstawia symbol wartości. Jeżeli wartość jest typu numerycznego i pozwalamy na jej użycie w wyrażeniach arytmetycznych, to często podkreśla się to przez dodanie symbolu + do nazwy typu wartości.

## 2.3 Unary predicate (relationship)



Rysunek 3: Unary relation (relacja unarna)

Rysunek 3 przedstawia dwie relacje unarne, opisane przez predykaty *smokes* oraz *has driving license*. Predykaty występują na diagramach jako prostokąty, połączone z obiektami za pomocą ciągłych linii.

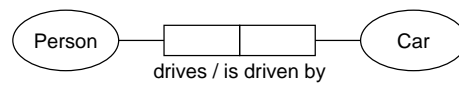
Powyższy diagram opisuje system, w którym możemy przechowywać dwie informacje na temat osób: czy dana osoba pali oraz czy posiada prawo jazdy. Predykaty unarne pozwalają na stwierdzenie pewnego faktu (pali/nie pali, ma/nie ma prawa jazdy) i na poziomie logicznym odpowiadają atrybutom typu **Boolean**.

Werbalizacja powyższych predykatów do zdań w języku naturalnym jest oczywista:

Person *smokes*  
Person *has driving license*

## 2.4 Binary predicate (relationship)

Rysunek 4 przedstawia relację binarną pomiędzy dwoma obiektami.

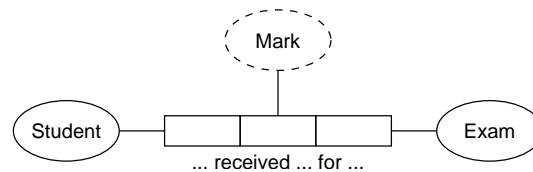


Rysunek 4: Binary relation (relacja binarna)

Preedykaty czyta się od lewej do prawej lub z góry na dół, o ile nie zostanie to określone inaczej (np. przez umieszczenie strzałek na diagramie, przy nazwie preedykatu). W przypadku preedykatów binarnych, często podaje się obydwa sposoby ich czytania. Tutaj jest to *drives* oraz *is driven by*, co pozwala na następującą werbalizację:

Person *drives* Car  
 Car *is driven by* Person

## 2.5 Higher arity predicates (relationships)



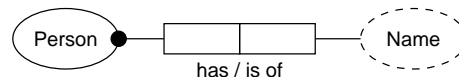
Rysunek 5: Ternary relation (relacja ternarna)

Rysunek 5 przedstawia relację ternarną, czyli relację, w której obiekty pełnią trzy różne role. Preedykaty dla relacji ternarnych i wyższych opisuje się jako zdania z „dziurami” na obiekty, przy czym ilość „dziur” odpowiada arności relacji. W powyższym przypadku preedykatem jest *... received ... for ...*, co razem z nazwami obiektów werbalizujemy jako:

Student *received* Mark *for* Exam

Teoretycznie, ORM nie ogranicza arności relacji, ale w praktyce większość relacji to relacje binarne. Relacje o wyższych arnościach zwykle daje się rozłożyć bez utraty informacji na mniejsze relacje (odpowiada to normalizacji relacji na poziomie logicznym).

## 2.6 Mandatory role



Rysunek 6: Mandatory role (rola obowiązkowa)

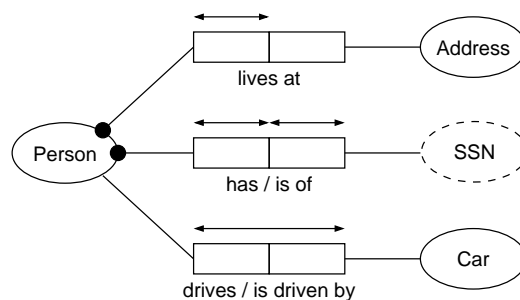
Rysunek 6 przedstawia relację binarną, której jedna z ról jest określona jako obowiązkowa dla obiektu, do którego jest przyłączona – opisane jest to przy użyciu dużej czarnej kropki na końcu połączenia. Oznacza to, że każdy obiekt musi wziąć udział w relacji, w odpowiedniej dla siebie roli. Werbalizacja całego diagramu jest następująca:

Person *has* Name / Name *is of* Person  
**each** Person *has some* Name

Przyjmuje się domyślnie, że jeżeli encja pełni tylko jedną rolę w jednej relacji na diagramie, to ta rola jest obowiązkowa i nie musi być już to specjalnie oznaczone. Jeżeli chcemy natomiast zaznaczyć, że dana encja może mieć instancje, które istnieją w bazie danych niezależnie od tego, czy biorą udział w jakiejś relacji (dotyczy to np. encji słownikowych lub inaczej referencyjnych), to o takiej encji mówimy, że jest *niezależna* (ang. *independent*) i do jej nazwy dołączamy znak „!”.

## 2.7 Uniqueness constraint

Rysunek 7 przedstawia trzy relacje binarne wraz z określeniem więzów unikatowych. Więzy takie pozwalają zdefiniować warunek dla każdego legalnego stanu bazy danych, oznaczający, że populacja danej relacji ma w danej roli (lub sekwencji ról) unikalne wartości. W przypadku pierwszej relacji chodzi o zastrzeżenie, że w każdej chwili populacja relacji będzie zawierać krotki, które nie powtarzają się w pierwszej kolumnie, czyli że dana instancja encji Person wystąpi co najwyżej raz w relacji. Nie ma takiego zastrzeżenia co do drugiej roli, czyli dana instancja encji Address może wystąpić kilka razy w relacji. Odpowiada to systemowi, gdzie osoba ma jeden adres, natomiast pod jednym adresem może mieszkać wiele osób, czyli jest to realizacja relacji  $1:n$ , z „1” po stronie Address. Pierwsza relacja binarna na rysunku 7 ma więc następującą, złożoną werbalizację (dołączono tutaj również werbalizacje wynikające z ról obowiązkowych):



Rysunek 7: Uniqueness constraint (więzy unikatowe)

Person *lives at* Address

**each** Person *lives at* **some** Address

**each** Person *lives at* **at most one** Address

Relacje binarne tego rodzaju (mandatory role + uniqueness constraint na jednej roli) nazywa się predykatami funkcyjnymi, gdyż istotnie działają jak funkcje w sensie matematycznym, przyporządkowując każdemu elementowi pewnego zbioru dokładnie jeden element drugiego zbioru. Na poziomie fizycznym takie relacje realizuje się jako pojedyncze kolumny (być może jako klucz obcy) w tabeli związanej z tą encją, przy której jest ograniczenie na obowiązkowość roli (czyli tutaj mielibyśmy kolumnę Address w tabeli Person).

Druga relacja binarna zawiera dwa proste więzy unikatowe, co oznacza, że zarówno Person, jak i SSN (ang. *social security number*) mogą pojawić się co najwyżej raz w relacji. To bardzo ważna właściwość oznaczająca iniekcję zbioru Person w zbiór SSN i jest stosowane do przedstawienia *schematu referencyjnego* dla danej encji. W tym przypadku chodzi o stwierdzenie, że osoba może być identyfikowana przez swój numer SSN. Ponieważ schemat referencyjny pojawia się praktycznie dla każdej encji (na poziomie fizycznym jako klucz główny), istnieje skrócona postać takiej relacji, pokazana na rysunku 8 – wersje a) i b) są równoważne.

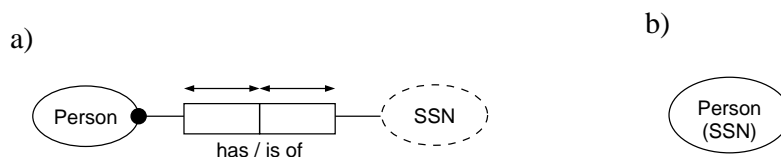
Oczywiście, werbalizacja dla drugiej relacji z rysunku 7 wygląda następująco:

Person *has* SSN / SSN *is of* Person

**each** Person *has* **some** SSN

**each** Person *has* **at most one** SSN

**each** SSN *is of* **at most one** Person



Rysunek 8: Reference scheme (schemat referencyjny)

Relacje, które nie biorą udziału w schemacie referencyjnym żadnego obiektu nazywamy *typami faktów*. Ich instancje (krotki w tych relacjach) są właściwymi danymi dotyczącymi obiektów w systemie.

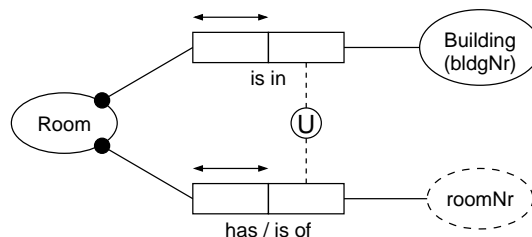
Trzecia relacja binarna z rysunku 7 posiada więzy unikatowe na obu rolach. Oznacza to, że kombinacja wartości dla tych ról musi być w relacji unikalna, odpowiadając relacji *m:n* na poziomie logicznym. Innymi słowy:

Person *drives* Car / Car *is driven by* Person

**it is possible that some Person *drives* more than one Car**

**and that some Car *is driven by* more than one Person**

## 2.8 External uniqueness constraint



Rysunek 9: External uniqueness constraint (zewnętrzne więzy unikatowe)

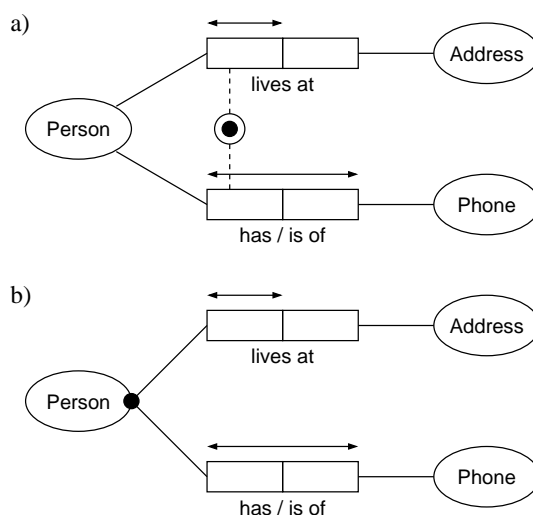
Rysunek 9 przedstawia diagram z użyciem zewnętrznych więzów unikatowych. Ich znaczenie jest takie, że pokój ma swój numer, ale ten numer nie określa pokoju w sposób unikalny. Pokój ma też przypisany numer budynku, ale on również nie jest unikalny. Jest natomiast unikalną kombinacją numeru pokoju i numeru budynku dla każdego pokoju. Można powiedzieć, że pokój jest identyfikowany przez parę (roomNr, bldgNr), co na poziomie fizycznym odpowiada tabeli, gdzie klucz główny jest złożony z dwóch kolumn. W tym przypadku można

zamiast litery U w kółku zastosować literę P (w kółku), co oznacza *explicite*, że więzy unikatowe stanowią główny schemat referencyjny (ang. *primary reference scheme*), co na poziomie fizycznym oznacza uznanie klucza kandydującego jako głównego.

Werbalizacja dla diagramu z rysunku 9 wygląda następująco:

Room *has* roomNr / roomNr *is of* Room  
**each** Room *has* **some** roomNr  
**each** Room *has* **at most one** roomNr  
Room *is in* Building  
**each** Room *is in* **some** Building  
**each** Room *is in* **at most one** Building  
**for each** roomNr  $r$  **and** Building  $b$   
**there is at most one** Room that  
*has* roomNr  $r$  **and** *is in* Building  $b$

## 2.9 Disjunctive mandatory role



Rysunek 10: Disjunctive mandatory role (role sumarycznie obowiązkowe)

Rysunek 10 przedstawia diagram z użyciem ról sumarycznie obowiązkowych. Intencją takiego zastrzeżenia jest określenie, że z danego zbioru ról obiekt pełni

co najmniej jedną z nich – żadna z ról nie jest obowiązkowa, ale w sumie udział obiektu w jednej z ról z całego zbioru jest obowiązkowy. Rysunek 10 pokazuje dwie alternatywne notacje. Dowolna liczba ról może być objęta przez takie więzy, pod warunkiem, że wszystkie role odnoszą się do tego samego obiektu. Znaczenie tych więzów na rysunku 10 jest takie, że system przechowuje dla każdej osoby albo adres albo telefon albo jedno i drugie. Na poziomie fizycznym predykaty funkcyjne z więzami ról sumarycznie obowiązkowych mogą być zrealizowane jako kolumny opcjonalne w tabeli z odpowiednim kodem sprawdzającym warunek, że obie kolumny nie mogą być NULL w tej samej chwili.

Weralizacja dla diagramu z rysunku 10 wygląda następująco:

Person *has* Phone / Phone *is of* Person

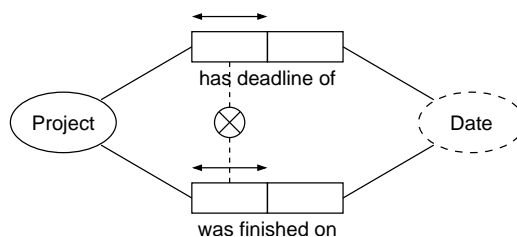
**it is possible that some Person has more than one Phone  
and that some Phone is of more than one Person**

Person *lives at* Address

**each Person lives at at most one Address**

**each Person lives at some Address or has some Phone**

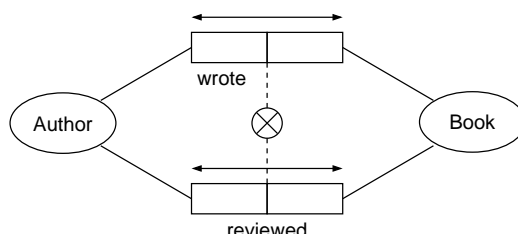
## 2.10 Exclusion



Rysunek 11: Exclusion (role wykluczające się)

Rysunek 11 przedstawia diagram z wykorzystaniem więzów wykluczania. Obiekt nie może pełnić więcej niż jednej roli w danym zbiorze ról. Znaczenie tych więzów w kontekście diagramu 11 jest takie, że jeśli dany projekt ma określony deadline, to na pewno nie jest dla niego znany czas zakończenia i odwrotnie – jeżeli znany jest czas zakończenia projektu, to nie przechowujemy informacji o dacie deadline.

Więzy wykluczania mogą dotyczyć pojedynczych ról (nawet wielu pojedynczych ról) lub *sekwencji* ról. Rysunek 12 przedstawia więzy wykluczania na dwóch parach ról.



Rysunek 12: Exclusion on role sequences (wykluczające się sekwencje ról)

Znaczenie diagramu 12 jest takie, że jeśli para (Author, Book) znajduje się w populacji górnej relacji, to nie może znajdować się w populacji relacji dolnej i odwrotnie. Innymi słowy – autor nie może recenzować książki, którą sam napisał (lub był współautorem, bo więzy unikatowe na obydwu relacjach oznaczają relacje  $m:n$ ).

Werbalizacja dla diagramu 11:

*Project has deadline of Date*

**each** Project *has deadline of* **at most one** Date

*Project was finished on Date*

**each** Project *was finished on* **at most one** Date

**no** Project **that** *has deadline of* **some** Date *was finished on* **some** Date

Werbalizacja dla diagramu 12:

*Author wrote Book*

**it is possible that some** Author *wrote* **more than one** Book  
**and that more than one** Author *wrote* **some** Book

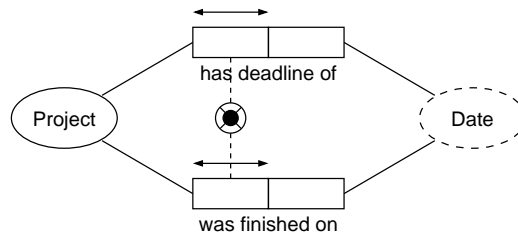
*Author reviewed Book*

**it is possible that some** Author *reviewed* **more than one** Book  
**and that more than one** Author *reviewed* **some** Book

**no** Author *wrote* **and** *reviewed* **the same** Book

Warto zwrócić uwagę, jak różnią się ostatnie linijki tych dwóch werbalizacji.

## 2.11 Exclusive mandatory (role obowiązkowe, wykluczające się)



Rysunek 13: Exclusive mandatory (role obowiązkowe, wykluczające się)

Rysunek 13 przedstawia złożenie więzów ról sumarycznie obowiązkowych z więzami ról wykluczających się. Znaczenie takich więzów jest takie, jak koncepcyjne złożenie operacji *or* (odpowiadające rolom sumarycznie obowiązkowym) i operacji wykluczania, dając w rezultacie operację *xor*. W notacji graficznej podkreśla się ten fakt (ortogonalność różnych więzów, które występują tutaj niezależnie) poprzez nałożenie na siebie symboli obowiązkowości (duża czarna kropka) i symbolu wykluczania (znak X). Znaczenie diagramu 13 jest takie, że dla każdego projektu znana jest albo data deadline albo data zakończenia, ale co najmniej jedna z nich i nie obie jednocześnie. Podobnie jak poprzednie więzy tak i te mogą dotyczyć wielu ról lub nawet wielu sekwencji ról.

Verbalizacja dla diagramu 13:

*Project has deadline of Date*

**each** *Project has deadline of at most one Date*

*Project was finished on Date*

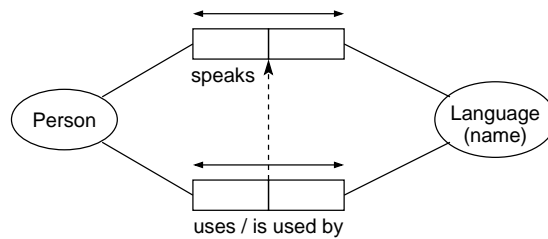
**each** *Project was finished on at most one Date*

**each** *Project has deadline of some Date or was finished on some Date*

**no** *Project that has deadline of some Date was finished on some Date*

## 2.12 Subset (podzbiór)

Rysunek 14 przedstawia więzy podzbiorów. Gwarantują one, że populacja danej relacji (a dokładnie jej rzut na wybrane role) jest zawsze podzbiorem innej relacji (a dokładnie rzutu na wybrane role). W przypadku diagramu 14 znaczenie tych



Rysunek 14: Subset (podzbiór, zawieranie się relacji)

więzów jest takie, że relacja dolna zawiera tylko takie krotki, które występują w relacji górnej. Dopuszcza się oczywiście taką możliwość, że relacja górna będzie zawierać krotki nieobecne w relacji dolnej. Innymi słowy, system zamodelowany na tym rysunku zakłada, że osoba używa tylko tych języków, które zna. Na poziomie fizycznym więzy podzbiorów często realizowane są jako klucz obcy (w tym przypadku dwu-kolumnowy) do relacji wskazanej przez kierunek strzałki.

Verbalizacja dla diagramu 14:

Person speaks Language

**it is possible that some Person *speaks* more than one Language and that more than one Person *speaks* some Language**

Person *uses* Language / Language *is used by* Person

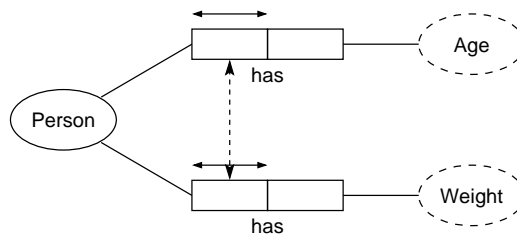
**it is possible that some Person *uses* more than one Language and that some Language *is used by* more than one Person**

**if Person *p* *uses* Language *l* then**

Person *p* *speaks* Language *l*

## 2.13 Equality (równość)

Rysunek 15 przedstawia więzy równości zbiorów. Ich znaczenie jest takie, jak złożenie dwóch więzów podzbiorów (*subset*), co widać również w symbolu graficznym. Jest to oczywiście zgodne z teorią zbiorów, gdzie zawieranie się zbiorów w obie strony jest równoznaczne z równością tych zbiorów. Znaczenie rysunku 15 jest takie, że wiek i wagę osoby przechowujemy zawsze razem, niedopuszczając do sytuacji, kiedy znana jest tylko jedna z tych wartości (ale pozwalając na sytuację, kiedy obie są nieznane).



Rysunek 15: Equality (równość zbiorów)

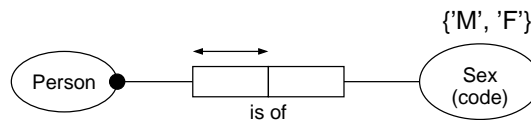
Werbalizacja dla diagramu 15:

Person *has* Age  
**each** Person *has at most one* Age

Person *has* Weight  
**each** Person *has at most one* Weight

Person *p has some* Weight **if and only if**  
 Person *p has some* Age

## 2.14 Value constraint (ograniczenie wartości)



Rysunek 16: Value constraint (ograniczenie wartości)

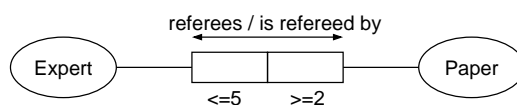
Rysunek 16 przedstawia więzy ograniczenia wartości obiektów. Więzy takie definiują dziedzinę (zbiór legalnych wartości) dla danego obiektu. W przypadku rysunku 16 chodzi o to, że osoba ma płeć określoną jako 'M' albo 'F' – żadna inna wartość nie jest tutaj dozwolona. Więzy ograniczenia wartości mogą być również określone jako zakres (np. { 'A' - 'Z' }) lub w sposób mieszany.

Werbalizacja dla diagramu 16:

Person *is of* Sex  
**each** Person *is of some* Sex

**each Person is of at most one Sex**  
**Sex(code) is an entity object type**  
**every Sex is identified by one distinct code**  
**the possible values of 'code' are: 'M' , 'F'**

## 2.15 Frequency constraint (ograniczenie częstości)



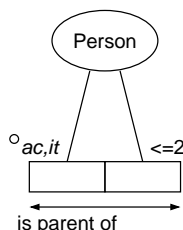
Rysunek 17: Frequency constraint (ograniczenie częstości)

Rysunek 17 prezentuje więzy ograniczające częstość występowania obiektu w danej roli. Ograniczenie takie może określać częstość jako stałą, zakres lub wyrażenie matematyczne (np.  $< 5$ ). Ostatnio rozszerzono notację ORM o możliwość określania częstości jako licznosci pewnego zbioru (np. #Car – jako liczba obiektów typu Car). Znaczenie rysunku 17 jest takie, że ekspert nie może referować więcej niż 5 artykułów oraz że artykuł musi być referowany przynajmniej przez 2 ekspertów. Ograniczenie częstości określa więc ile razy jedna wartość może (lub powinna) się pojawić w populacji relacji, na pozycji wyznaczonej przez rolę.

Warto zauważyć, że wartości częstości są określone przy rolach odpowiednich dla obiektu, którego więzy dotyczą – odwrotnie niż w notacjach ER i UML, gdzie ograniczenia częstościowe określa się po przeciwnej stronie relacji lub asocjacji.

Weralizacja dla diagramu 17:

Expert *referees* Paper / Paper *is refereed by* Expert  
**it is possible that some Expert referees more than one Paper**  
**and that some Paper is refereed by**  
**more than one Expert**  
**each Expert occurs at most 5 times or not at all**  
**each Paper occurs at least 2 times or not at all**



Rysunek 18: Ring constraints (ograniczenia dla relacji pierścieniowych)

## 2.16 Ring constraints (ograniczenia dla relacji pierścieniowych)

Rysunek 18 przedstawia relację, w której obiekt może odgrywać więcej niż jedną rolę – w tym przypadku osoba jest rodzicem osoby. Ograniczenie częstości po przeciwnej stronie predykatu *is parent of* oznacza, że co najwyżej dwie krotki mogą zawierać tą samą osobę w drugiej roli, czyli że jedna osoba ma co najwyżej dwoje rodziców. Relacje pierścieniowe mogą podlegać dodatkowym ograniczeniom – w tym przypadku zastrzegamy, że relacja *is parent of* jest acykliczna (*ac*) oraz nieprzechodnia (*it*).

Werbalizacja dla diagramu 18:

Person *is parent of* Person

**it is possible that some Person is parent of more than one Person and that more than one Person is parent of some Person**

**each Person p that occurs in Person is parent of Person p occurs there at most 2 times**

**if Person p1 is parent of Person p2 and Person p2 is parent of Person p3 then it cannot be that**

Person p1 is parent of Person p3

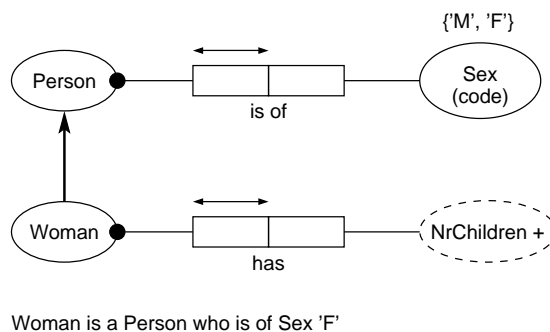
**a Person cannot cycle back to itself through one or more applications of this relationship**

W notacji ORM zdefiniowano następujące symbole dla ograniczeń w relacjach pierścieniowych:

- *ans* – antysymetryczna
- *as* – asymetryczna
- *ac* – acykliczna
- *ir* – niezwrrotna
- *it* – nieprzechodnia
- *sym* – symetryczna

Niektóre z tych ograniczeń mogą być stosowane razem.

## 2.17 Subtyping (dziedziczenie)



Rysunek 19: Subtyping (dziedziczenie)

Rysunek 19 pokazuje przykład dziedziczenia, czyli zdefiniowania podklasy lub podtypu. Relację dziedziczenia w ORM rozumie się wyłącznie w kategoriach teorii zbiorów (podobnie jak wszystkie inne pojęcia) i oznacza ona, że populacja pewnego typu (czyli zbiór jego instancji) – podtypu – jest zawsze podzbiorem populacji innego typu – nadtypu. Jeden typ może być uznany jako podtyp innego typu tylko wtedy, gdy w każdym legalnym stanie systemu każdy obiekt, który jest instancją podtypu jest również instancją nadtypu. Przyjmuje się, że podtyp „dziedziczy” udział we wszystkich rolach dotyczących swojego nadtypu. W przypadku rysunku 19 chodzi o stwierdzenie, że kobieta jest osobą, czyli że w każdym stanie bazy danych każda kobieta będzie miała odpowiadającą jej instancję typu osoba. Co więcej, ORM wymaga aby podtyp był formalnie zdefiniowany przy użyciu

informacji dostępnych na poziomie nadtypu. Diagram 19 określa, że osoba ma płeć (relacja funkcyjna) oraz że kobieta ma płeć (to jest relacja dziedziczona po osobie) i dodatkowo liczbę dzieci (znowu relacja funkcyjna).

Verbalizacja dla tego diagramu wygląda następująco:

**each** Woman **is a** Person **but not every** Person  
**is necessarily a** Woman  
Woman **is a subtype of** Person /  
Person **is a supertype of** Woman  
**subtype definition:** Woman **is a** Person **who is of** Sex 'F'

Sex(code) **is an entity object type**  
**every** Sex **is identified by one distinct code**  
**the possible values of 'code' are:** 'M' , 'F'

Person *is of* Sex  
**each** Person *is of some* Sex  
**each** Person *is of at most one* Sex

Woman *has* NrChildren  
**each** Woman *has some* NrChildren  
**each** Woman *has at most one* NrChildren

Definicja podtypu jest tutaj bardzo ważna:

Woman **is a** Person **who is of** Sex 'F'

W ten sposób definiujemy, jakie instancje typu osoba są uznane również za instancje typu kobieta. Co więcej, formalne definicje podtypów zwykle odnoszą się do informacji, które na poziomie nadtypu są w pewien sposób określone (np. przez ograniczenia wartości), co implikuje różne cechy „hierarchii” dziedziczenia. Na przykład dołożenie nowego podtypu mężczyzna do diagramu, z definicją:

Man **is a** Person **who is of** Sex 'M'

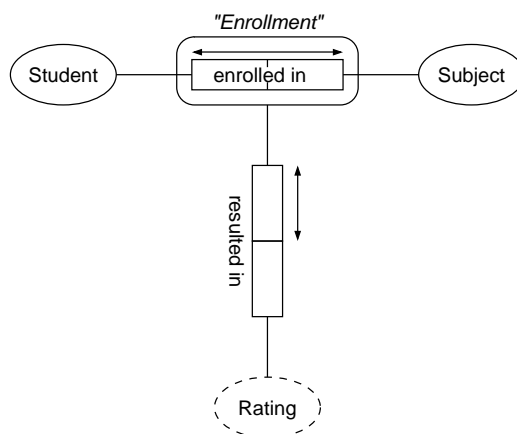
implikuje *rozłączność* i *kompletność* hierarchii dziedziczenia, czyli fakt, że nadtyp jest podzielony przez swoje podtypy na zasadzie partycji – nie ma osoby, która byłaby jednocześnie kobietą i mężczyzną (*rozłączność*) i nie ma osoby, która nie byłaby ani kobietą ani mężczyzną (*kompletność*). Cechy te wynikają z dwóch faktów:

1. płeć może mieć tylko dwie różne wartości i każda z nich determinuje jeden z podtypów
2. płeć jest obowiązkowa dla osoby

Oznacza to również, że użycie symbolu „obowiązkowe, wykluczające się” (jak na rysunku 13) pomiędzy grubymi strzałkami symbolizującymi dziedziczenie jest tutaj możliwe, ale nieobowiązkowe, bo wynika ono z definicji podtypów oraz więzów użytych na poziomie nadtypu. Podobnie, oryginalny diagram 19 (bez dodatkowego podtypu dla mężczyzny) oznacza, że mogą istnieć osoby, które nie są kobietami – wynika to z definicji podtypu kobieta oraz więzów na obiekcie płeć.

Istnieje możliwość zdefiniowania dziedziczenia z wieloma nadtypami (wielodziedziczenie) – jedyne ograniczenie jest takie, że relacja dziedziczenia musi być acykliczna.

## 2.18 Nesting, objectified relationships (zagnieżdżanie)



Rysunek 20: Nesting, objectified relationship (zagnieżdżenie)

Rysunek 20 pokazuje przykład zagnieżdżonej relacji, czyli relacji, która nabiera cech obiektu. Celem takiej operacji jest stworzenie możliwości użycia relacji w relacjach z innymi obiektami. W przykładzie z rysunku 20 chodzi o to, że relacja studenta z przedmiotem (*enrollment*) jest czymś, co w innym miejscu

systemu może być wykorzystane tak samo, jak gdyby był to normalny obiekt. W tym przypadku przechowujemy ocenę, którą student uzyskał za dany przedmiot. Warto zauważyć, że podobny efekt (w sensie przechowywanej informacji) mogliśmy uzyskać przy użyciu relacji ternarnej (Student, Subject, Rating), jednak w przypadku zagnieżdżania mamy możliwość przechowania faktu udziału studenta w przedmiocie *bez konieczności przechowywania oceny lub udział w większej liczbie relacji (osobne oceny z kolokwium, projektu, egzaminu, itp.)*.

Verbalizacja dla diagramu 20 wygląda następująco:

Student *enrolled in* Subject

**it is possible that some** Student *enrolled in*  
**more than one** Subject  
**and that more than one** Student *enrolled in*  
**some** Subject  
**this fact is nested as** 'Enrollment'

Enrollment *resulted in* Rating

**each** Enrollment *resulted in at most one* Rating

Notacja ORM pozwala jedynie za zagnieżdżanie relacji całkowicie pokrytych przez więzy unikatowe (jak w przykładzie) lub binarnych *1:1*, choć rozważa się możliwość rozluźnienia tej reguły dla innych przypadków.

## 2.19 Inne symbole

Przedstawione symbole nie wyczerpują pełnych możliwości notacji ORM. Istnieje jeszcze kilka symboli pokrywających bardziej zaawansowane aspekty modelowania (*join-constraint, relative closure, non-primary supertypes, collection constructors*), jednak bardzo rzadko występuje konieczność ich stosowania w praktyce. Pełny opis notacji ORM można znaleźć w [1].

## 3 Procedura projektowa

Każdy język służący do modelowania czegokolwiek posiada również swoją własną procedurę lub metodologię użycia.

Język ORM posiada procedurę nazwaną *Conceptual Schema Design Procedure*, czyli *Procedura Projektowania Konceptyjnego*. Została ona zdefiniowana jako siedem kroków:

1. Przedstaw znane przykłady danych w postaci elementarnych faktów.  
Jako elementarny fakt rozumiemy relację (wraz z jej więzami), której nie da się rozłożyć na mniejsze relacje bez utraty informacji. Bardziej formalnie, fakt nie jest elementarny jeżeli jakieś złączenie jego projekcji na rozłączne role daje zawsze tą samą relację. Definicja, która sprawdza się w praktyce mówi, że fakt jest elementarny, gdy wszystkie zależności funkcyjne pomiędzy rolami wychodzą wyłącznie z ról objętych więzami unikatowymi oraz gdy co najwyżej jedna rola w relacji nie jest pokryta przez jakieś więzy unikatowe.
2. Narysuj typy faktów i sprawdź populacje.  
Sprawdzenie populacji polega na wypełnieniu fikcyjnych „tabelek” odpowiadających typom faktów i ich weryfikacji.
3. Sprawdź, czy jakieś typy encji mogą być połączone oraz znajdź zależności arytmetyczne na diagramie.
4. Dodaj więzy unikatowe i sprawdź arność typów faktów (np. czy spełniają zasadę, że co najwyżej jedna rola w relacji może nie być objęta więzami unikatowymi – zasada ta wynika z definicji faktu elementarnego).
5. Dodaj więzy ról obowiązkowych i sprawdź zależności logiczne.
6. Dodaj ograniczenia wartości, więzy zbiorów i dziedziczenia.
7. Dodaj pozostałe więzy i wykonaj ostateczne sprawdzenie diagramu.

Procedura ta jest dokładnie opisana w literaturze.

## 4 Procedura Rmap

Ważną cechą notacji ORM jest fakt, że ma ona solidne podstawy matematyczne. Pozwala to na zdefiniowanie procedur przekształcających opis ORM w fizyczną implementację bazy danych. Procedura transformująca diagram ORM w relacyjną bazę danych nosi nazwę *Rmap* i jest zaimplementowana między innymi w programie Microsoft VisioModeler, z możliwością generowania opisów DDL dla popularnych serwerów baz danych.

Ścisłe przestrzeganie zaleceń procedury CSDP prowadzi do diagramów, które po zastosowaniu procedury *Rmap* gwarantują uzyskanie bazy w 5 postaci normalnej.

Ponieważ procedura Rmap jest określona bardzo dokładnie i nie pozwala na żadną swobodę w wyborze różnych opcji, to z danego diagramu ORM zawsze wynika tylko jedna implementacja na poziomie fizycznym. Jeżeli zachodzi potrzeba dokonania zmian na poziomie fizycznym (np. ze względów wydajnościowych należy użyć innego układu tabel), to wykonuje się ją przez odpowiednie transformacje diagramu ORM – tak, aby po ponownym zastosowaniu procedury Rmap uzyskać zamierzony efekt. Taki proces nazywa się *optymalizacją koncepcyjną* i można go przeprowadzić w oparciu o szereg praw dotyczących równoważności diagramów.

## Literatura

[1] Terry Halpin. *Information Modeling and Relational Databases*. Morgan-Kaufmann.

[2] <http://www.orm.net/>