# Model konceptualny
# jako wspólna platforma niejednorodnych modeli logicznych

Dariusz Put

Akademia Ekonomiczna w Krakowie

**Streszczenie**: W artykule omówiono założenia modelu konceptualnego, na podstawie, którego tworzone są projekty baz danych niezależne od modeli logicznych. Pokazano, jak model ten może być przekształcony do postaci logicznego modelu relacyjnego i obiektowego. Omówiono także zagadnienie przygotowania danych zapisanych w relacyjnych bazach danych dla obiektowo zorientowanych aplikacji oraz zwrócono uwagę na rolę, jaką w takim przekształceniu może odgrywać uniwersalny model konceptualny.

**Abstract**: The features of a conceptual model, which is used in designing databases independent of specific logical models, are discussed in the paper. It is described, how the conceptual model may be transformed to relational or object-oriented logical model. The paper discusses also the problem of preparing data stored in relational DBMS for object-oriented applications and describes the role the universal conceptual model plays in such a transformation.

## I. Wstęp

Już kilka lat po opublikowaniu w 1970 roku przez E. F. Codda [7] podstaw modelu relacyjnego zaczęły się pojawiać relacyjne systemy zarządzania bazami danych (SZBD). W modelu tym przyjęto założenie, że dane o faktach, czyli obiektach i zdarzeniach, przechowywane są w relacjach, które w pojawiających się kolejnych latach rozwiązaniach praktycznych reprezentowane były (i nadal są) w postaci tabel. Ponieważ odzwierciedlony w bazie danych fragment rzeczywistości jest zwykle złożony, rejestracja danych dotyczących jednego faktu wymaga niejednokrotnie dokonania wpisu w wielu tabelach. Naturalna złożoność faktów musi być przystosowana do założeń modelu relacyjnego i skonwertowana w taki sposób, aby mogła być zapisana w systemie składającym się z „płaskich" tabel.

W latach osiemdziesiątych XX wieku zaczęły powstawać obiektowe języki programowania, w których możliwe stało się tworzenie struktur danych na wyższym poziomie abstrakcji niż w językach strukturalnych, bardziej zbliżonym do modelowanej rzeczywistości. Istnienie możliwości reprezentacji obiektów w sposób odzwierciedlający ich naturalną złożoność ułatwiło tworzenie procedur obiektowych i operowanie na faktach występujących w opisywanym fragmencie rzeczywistości. Koncepcja obiektowych języków programowania była rozwijana w kolejnych latach – języki te stanowią podstawę dla zdecydowanej większości tworzonych obecnie aplikacji.

Metodologia obiektowa przyjęła się i jest powszechnie wykorzystywana w dziedzinie programowania. Jest oczywiste, że aplikacje muszą niejednokrotnie korzystać z danych zapisanych w systemach bazodanowych. Jednak w zakresie przechowywania danych, mimo iż podejmowane są prace nad stworzeniem modelu obiektowego, a nawet powstało szereg obiektowych SZBD[1], dominujące są nadal systemy relacyjne. Można wskazać, co najmniej dwie przyczyny takiego stanu:

1. Brak silnych podstaw teoretycznych dla modelu obiektowego.

Od momentu pojawienia się artykułu E. F. Codda cały czas trwają badania nad modelem relacyjnym. Pojawiają się nowe pomysły i rozwiązania. Język SQL, umożliwiający praktyczne implementacje baz danych w relacyjnych SZBD, był kilkakrotnie standaryzowany. Dzięki temu model ten ma silne podstawy teoretyczne. Nie można tego powiedzieć o modelu obiektowym. Nie opracowano dotychczas powszechnie akceptowanego standardu, ani nawet modelu nieformalnego, co oznacza, że producenci systemów implementują w nich własne pomysły i rozwiązania, przez co współdziałanie takich systemów między sobą a także z aplikacjami tworzonymi w językach obiektowych jest utrudnione. Problem ten trafnie skomentował C. J. Date [9] pisząc: „*Wiele pojęć ze świata obiektowego – a przynajmniej opublikowane definicje tych pojęć – jest mocno nieścisłych. Nie ma prawdziwej zgody, a jest dużo kontrowersji, nawet na najbardziej podstawowym poziomie. W szczególności, nie ma abstrakcyjnego, formalnie zdefiniowanego 'obiektowego modelu danych', ani nawet zgody, co do modelu nieformalnego w rezultacie tego stanu rzeczy definicje i wyjaśnienia przedstawione w tej części książki* [dotyczącej modelu obiektowego – przyp. aut.] *nie są powszechnie akceptowane i niekoniecznie odpowiadają sposobom działania realnych systemów obiektowych. Prawie każde wyjaśnienie czy każda definicja mogłyby być kwestionowane przez innego eksperta z tej dziedziny*".

2. Ograniczone możliwości konwersji danych z modelu relacyjnego do obiektowego.

Istniejące SZBD to w większości systemy relacyjne. Zaprojektowane i zaimplementowane bazy danych w znacznej większości działają, więc w tych systemach. Próba konwersji bazy danych do obiektowego SZBD jest przedsięwzięciem ryzykownym i zwykle nieefektywnym, gdyż wymaga przebudowy całego, niejednokrotnie bardzo złożonego i sprawnie działającego systemu oraz utworzenia projektu obiektowego

[1] Np. Illustra, $O_2$, GemStone, ObjectStore.

implementowanego następnie w nie posiadającym podstaw teoretycznych obiektowo zorientowanym SZBD. Konwersja systemu wymaga nie tylko zmiany sposobu przechowywania danych, ale także przebudowy lub nawet napisania na nowo aplikacji, które z tych danych korzystają.

Dane wykorzystywane w procedurach obiektowych są, więc zwykle pobierane z systemów relacyjnych. Wiąże się z tym szereg problemów i niedogodności – zagadnienie to jest określane w literaturze jako tzw. „niezgodność impedancji". Można wskazać kilka sposobów rozwiązania tego problemu:

– konwersja systemu przechowywania danych na obiektowy – co, jak wyżej starano się uzasadnić, jest trudne w realizacji, wiąże się bowiem z koniecznością przebudowy systemu, i oznacza zwykle konieczność poniesienia dużych nakładów związanych z wymianą modelu przechowywania danych, zakupem nowego SZBD oraz utworzeniem aplikacji, w tym także interfejsu użytkownika,
– operowanie na danych w postaci „płaskich" tabel – oznacza to, że możliwości modelu obiektowego nie będą w pełni wykorzystane, a w kodzie programu trzeba uwzględnić różnicę w typach i sposobach reprezentacji danych pomiędzy modelem relacyjnym (w którym przechowywane są dane) i obiektowym (w którym tworzone jest oprogramowanie),
– zastosowanie warstwy pośredniej – dane zwracane przez zapytanie są wybierane z relacyjnej bazy danych, w warstwie pośredniej są konwertowane na obiekty i w takiej postaci przekazywane do procedur napisanych w języku obiektowym.

Ostatnie z zaproponowanych rozwiązań wydaje się najefektywniejsze. Wiąże się jednak z koniecznością dokonania pewnej modyfikacji systemu i stworzenia mechanizmów automatycznie konwertujących dane obiektowe na ich relacyjną reprezentację i na odwrót. Potrzeba takiej konwersji musi być wpisana w projekt systemu.

Jednym z etapów cyklu życia bazy danych (por. rys. 1) jest proces projektowania. Najpierw tworzony jest projekt konceptualny, niezależny od rozwiązań charakterystycznych dla jakiegokolwiek modelu logicznego i SZBD. Następnie podejmowana jest decyzja dotycząca typu zastosowanego modelu logicznego oraz tworzony jest projekt logiczny oparty na wybranym modelu. Z kolei następuje wybór konkretnego SZBD zgodnego z wybranym modelem logicznym i tworzony jest projekt fizyczny uwzględniający specyfikę wybranego SZBD, który następnie jest implementowany w tym systemie.

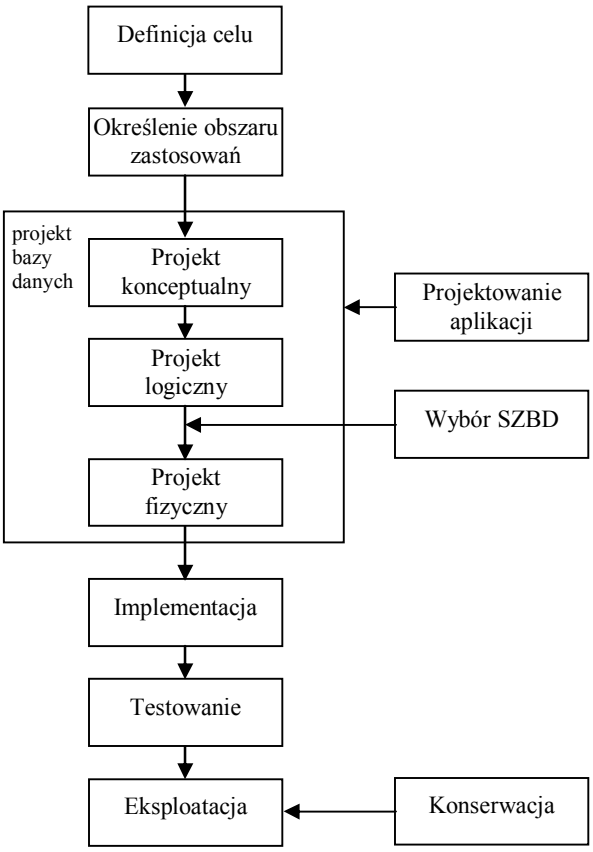
Wspólną platformę dla różnorodnych modeli logicznych, która może być wykorzystana do budowy warstwy pośredniej, może stanowić model konceptualny. W artykule omówiono założenia tego modelu, umożliwiającego zaprojektowanie sposobu przechowywania danych bez konieczności uwzględniania własności charakterystycznych dla modeli logicznych. Opisano także modele logiczne: obiektowy i relacyjny i pokazano na przykładzie praktycznym sposób przekształcenia projektu konceptualnego w logiczny, uwzględniający specyfikę wybranej metody przechowywania danych.

## II. Założenia modelu konceptualnego

Głównym celem konceptualnego modelowania bazy danych jest stworzenie projektu odzwierciedlającego analizowany fragment rzeczywistości, wolnego od szczegółów, które umiejscawiałyby go wśród modeli określonej klasy (obiektowych, relacyjnych lub innych) oraz niezależnego od platformy programowej. Efektem końcowym procesu projektowania konceptualnego jest identyfikacja zbioru faktów, o których dane mają być przechowywane w bazie danych oraz związków między tymi faktami. W procesie tym uczestniczy zarówno projektant, jak i przyszły użytkownik systemu. Na podstawie informacji przekazanych przez

przedstawicieli organizacji, dla której przygotowywana jest baza danych (przyszłego użytkownika) tworzony jest projekt zawierający trzy rodzaje elementów:

- fakty, czyli obiekty i zdarzenia o których dane mają być przechowywane w bazie danych,
- atrybuty charakteryzujące poszczególne fakty,
- typy powiązań między faktami.




**Rys. 1.** Cykl życia aplikacji bazy danych

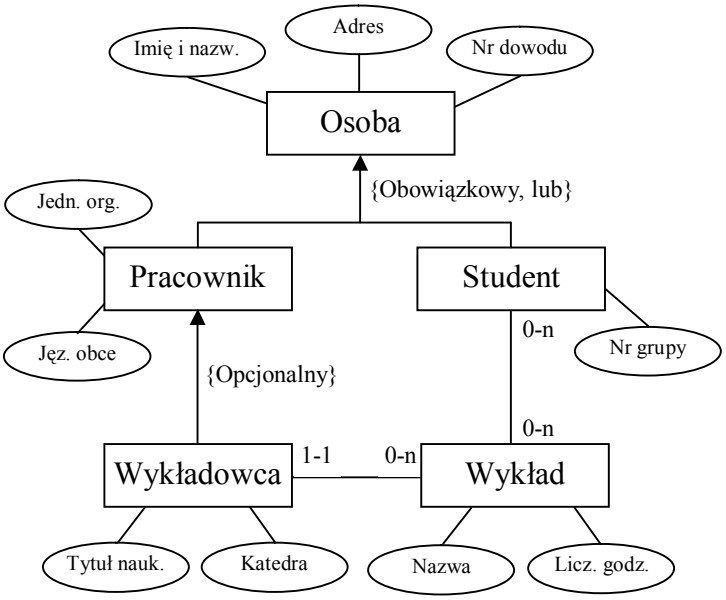
(źródło: opracowanie własne na podstawie C. Begg, T. Connolly [2], s. 272).

Projekt jest najczęściej wyrażony w formie graficznej w postaci diagramu związków encji oraz uzupełniony o opis słowny, w którym informacje zawarte w projekcie graficznym są szczegółowo scharakteryzowane. Na diagramie fakty są oznaczane zwykle w postaci prostokątów, atrybuty występują jako elipsy, a związki między faktami są oznaczone liniami łączącymi prostokąty oraz symbolami umieszczonych obok tych linii, opisującymi typ związku (por. rys. 2).

W modelu konceptualnym atrybuty mogą być zarówno proste, jak i złożone: wielowartościowe i segmentowe. Atrybuty złożone są eliminowane w konkretnym modelu logicznym, a sposób ich reprezentacji jest zależny od typu przyjętego modelu (obiektowy lub relacyjny).

Na rysunku 2 zaprezentowano przykładowy projekt oparty na modelu konceptualnym. Wyróżniono pięć faktów (*Osoba*, *Pracownik*, *Student*, *Wykładowca*, *Wykład*) charakteryzowanych przez atrybuty, wśród których można wyróżnić zarówno atrybuty proste (atomowe) – takich jest większość, jak i złożone. Atrybutami segmentowymi są *Imię i nazwisko* oraz *Adres* charakteryzujące obiekt *Osoba*. Jedynym atrybutem wielowartościowym jest *Języki obce* opisujący obiekt *Pracownik*. Obiekty *Pracownik* i *Student* pochodzą od obiektu *Osoba* (wskazuje na to linia w postaci strzałki skierowanej

grotem w stronę faktu nadrzędnego). Charakter związku określony jako *{Obowiązkowy, lub}* oznacza, że każdy obiekt *Osoba* musi być *Pracownikiem* albo (nigdy i) *Studentem*. Innymi słowy, fakt zarejestrowania w bazie danych osoby musi być uzupełniony o wpis określający, czy osoba ta jest studentem, czy pracownikiem. Związek między *Pracownikiem* a *Wykładowcą* jest *{Opcjonalny}*, przy czym obiektem nadrzędnym jest *Pracownik*. Oznacza to, że można rejestrować pracowników bez konieczności zapisywania wykładowców. Umieszczenie w bazie danych o wykładowcy wymaga natomiast uprzedniego zarejestrowania go jako pracownika (każdy wykładowca jest pracownikiem, ale nie każdy pracownik to wykładowca). *Wykładowcy* prowadzą *Wykłady*, charakter związku wskazuje, że ten sam wykładowca może prowadzić wiele wykładów, a każdy wykład jest prowadzony dokładnie przez jednego wykładowcę. Między obiektami *Student* i *Wykład* istnieje, wielowartościowy związek dwukierunkowy: student może uczęszczać na wiele wykładów, w jednym wykładzie




może uczestniczyć wielu studentów.

**Rys. 2**. Przykładowy projekt konceptualny (źródło: opracowanie własne).

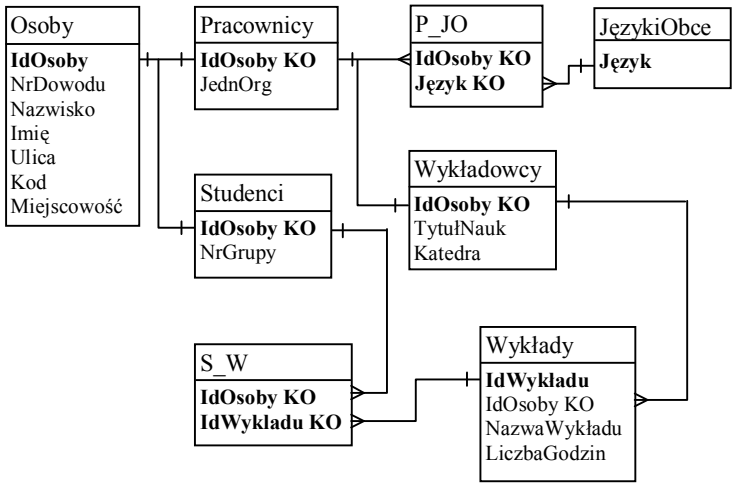
## III. Reprezentacja faktów w logicznym modelu relacyjnym

Wybór modelu relacyjnego za podstawę metody przechowywania danych oznacza, iż należy przekształcić uniwersalny projekt konceptualny na projekt, w którym dane o faktach są przechowywane w tabelach, a atrybuty, zgodnie z założeniami modelu relacyjnego, są atomowe, co niekiedy oznacza konieczność utworzenia dodatkowych tabel i połączeń między nimi. Projekt logiczny wyrażony w postaci diagramu stanowi w dalszej fazie procesu podstawę do utworzenia projektu fizycznego i jego implementacji w wybranym relacyjnym SZBD.

W modelu relacyjnym dane mogą być przechowywane, w zależności od złożoności faktu, w jednej lub kilku tabelach, przy czym, jeśli wykorzystywanych jest kilka tabel, są one zwykle połączone relacjami. Przekształcenie faktów występujących w projekcie konceptualnym na tabele w logicznym projekcie relacyjnym wymaga przeprowadzenia analizy uwzględniającej rodzaj powiązań występujących między faktami oraz złożoność charakteryzujących je atrybutów. Dokonując przekształcenia należy zwrócić uwagę na następujące aspekty:

1. Fakty są jednorodne, wszystkie opisywane przez tę samą liczbę atrybutów i wszystkie atrybuty są atomowe. W takiej sytuacji jednemu faktowi odpowiada jedna tabela

w modelu relacyjnym, każdemu atrybutowi jedna kolumna. Dopisanie do bazy danych nowego faktu wymagać będzie dodania jednego rekordu do jednej tabeli.
2. Fakty są opisywane przez różną liczbę atrybutów, co w praktyce oznacza, że dla jednych wymagane jest zapisanie większej liczby danych niż dla innych. W takiej sytuacji dane są przechowywane w dwóch tabelach: głównej zawierającej dane o wszystkich faktach oraz tabeli-podzbiorze przechowującej dodatkowe dane o faktach opisywanych przez większą liczbę atrybutów. Tabele są połączone relacją 1-1. Jeśli typ związku jest obowiązkowy, należy rozważyć rozwiązanie oparte tylko na jednej tabeli zawierającej wszystkie dane opisujące fakty występujące w tym związku.
3. Fakty zawierają pola wielowartościowe. Usunięcie takich pól wymaga utworzenia dodatkowej tabeli. Dane są umieszczane w dwóch tabelach połączonych relacją 1-n. Dla zachowania więzów integralności konieczne jest utworzenie dodatkowej tabeli zawierającej wszystkie wartości, które może przyjmować eliminowane pole wielowartościowe. W konsekwencji między tabelą faktów a tabelą słownikową istnieje relacja n-m.
4. Fakty są połączone zależnością 1-n. W tabeli podrzędnej umieszczone jest pole (klucz obcy) służące do utworzenia związku.
5. Fakty są połączone współzależnie, istniejący związek jest typu n-m. Relacja jest tworzona poprzez dodatkową tabelę (łączącą) zawierającą klucze podstawowe połączonych tabel.
6. Fakty są połączone współzależnie a tabela łącząca zawiera dodatkowe pola charakteryzujące parametry połączenia.




**Rys. 3**. Relacyjny projekt logiczny zaprojektowany na podstawie projektu konceptualnego z rys. 2 (źródło: opracowanie własne).

Na rysunku 3 pokazano projekt logiczny relacyjnej bazy danych utworzony na podstawie projektu konceptualnego (rys. 2). Klucze podstawowe wytłuszczono, klucze obce oznaczono jako KO. Tabele *Pracownicy* i *Studenci* są połączone z tabelą *Osoby* relacjami 1-1, w których tabela *Osoby* jest nadrzędna, a tabele *Pracownicy* i *Studenci* podrzędnymi. Klucz obcy relacji to w obydwu przypadkach pole *IdOsoby*. Rejestracja pracownika lub studenta wymaga wpisania do bazy danych dwóch rekordów: jednego w tabeli *Osoby* oraz drugiego odpowiednio w tabeli *Pracownicy* lub *Studenci*[2]. Tabele *Pracownicy* i *Wykładowcy* także są połączone relacją 1-1. W relacji tej tabela *Pracownicy* jest nadrzędna. Oznacza to, że aby zarejestrować wykładowcę, należy najpierw wpisać dane o pracowniku. Jest to zgodne

[2] Problem ten można także rozwiązać rezygnując z tabeli *Osoby* i przenosząc wszystkie znajdujące się w niej atrybuty zarówno do tabeli *Studenci*, jak i *Pracownicy*.

z założeniami modelu konceptualnego – każdy wykładowca jest pracownikiem. Zgodnie z modelem konceptualnym związek między *Wykładowcami* a *Wykładami* jest typu 1-n. Kluczem obcym odpowiadającej mu relacji jest pole *IdOsoby* znajdujące się w tabeli *Wykłady*. Między tabelami *Studenci* i *Wykłady* istnieje relacja n-m, więc w modelu relacyjnym wymaga to utworzenia tabeli łączącej (*S_W*) zawierającej klucze podstawowe obydwu tabel biorących udział w relacji oraz dwóch relacji 1-n, w których tabelą związaną (podrzędną) jest tabela łącząca. Tabela łącząca ma wielopolowy (dwupolowy) klucz podstawowy.

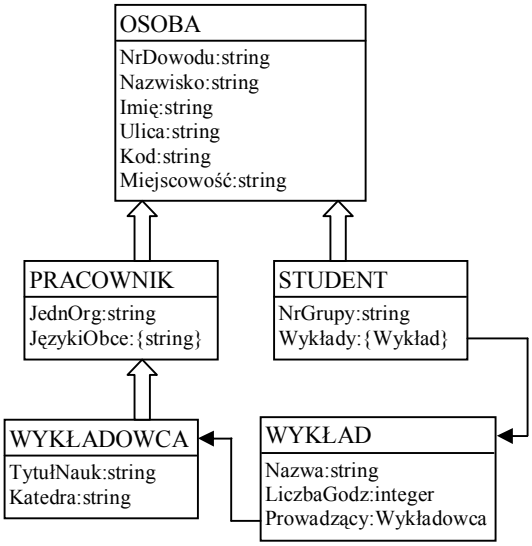
W modelu relacyjnym nie mogą występować pola segmentowe – zostały one usunięte i zastąpione atrybutami atomowymi (*Nazwisko i imię* polami *Nazwisko* oraz *Imię*, *Adres* polami *Ulica*, *Kod*, *Miejscowość*). Jedyne pole wielowartościowe występujące w modelu konceptualnym zostało wyeliminowane poprzez utworzenie dodatkowej tabeli zawierającej dopuszczalne języki obce (*JęzykiObce*) oraz relacji n-m z tabelą *Pracownicy*.

Ostatecznie pięć faktów zapisanych w modelu konceptualnym wymaga w tym przypadku utworzenia ośmiu tabel: pięciu odpowiadających tym faktom, dwóch łączących, co było związane z koniecznością utworzenia relacji n-m, oraz jednej tabeli słownikowej (*JęzykiObce*).

## IV. Reprezentacja faktów w logicznym modelu obiektowym

W modelu obiektowym do reprezentacji faktów wykorzystywane są klasy, a każdy obiekt jest instancją jednej z istniejących klas. Model obiektowy jest bardziej zbliżony do konceptualnego niż relacyjny, pozwala, bowiem na modelowanie rzeczywistości na wyższym poziomie abstrakcji – klasy można zdefiniować w sposób bardziej adekwatny do projektowanej rzeczywistości. Każdemu faktowi uwzględnionemu w projekcie konceptualnym odpowiada zwykle jedna klasa w projekcie obiektowym. Podczas konwersji projektu konceptualnego na obiektowy wykorzystywane są następujące własności modelu obiektowego:

1. Typ krotkowy – każdy obiekt jest reprezentacją krotkową.
2. Dziedziczenie – podklasa dziedziczy atrybuty i metody nadklasy.
3. Atrybuty, których typem jest identyfikator istniejącego obiektu klasy wskazanej w definicji atrybutu.
4. Typ zbiorowy – wartością atrybutu może być zbiór identyfikatorów istniejących obiektów lub zbiór wartości określonego typu.




**Rys. 4**. Obiektowy projekt logiczny zaprojektowany na podstawie projektu konceptualnego z rys. 2 (źródło: opracowanie własne).

Na rysunku 4 zaprezentowano projekt bazy danych oparty na obiektowym modelu logicznym, utworzony na podstawie projektu konceptualnego (rys. 2). Wyróżniono pięć klas. Klasa *Osoba* jest nadklasą w stosunku do dwóch innych: *Student* i *Pracownik*. Obiekty klas *Student* i *Pracownik* dziedziczą, więc atrybuty i zachowanie obiektu klasy *Osoba*. Podobnie klasa *Wykładowca* jest podklasą klasy *Pracownik*, a pośrednio także podklasą klasy *Osoba*. Aby zarejestrować nowego wykładowcę należy utworzyć obiekt klasy *Wykładowca*, który oprócz atrybutów zdefiniowanych w tej klasie będzie zawierał wszystkie atrybuty (i metody) obiektów *Pracownik* oraz *Osoba*. Klasa *Student* zawiera atrybut *Wykłady*, który jest typu zbiorowego. Dla każdego obiektu klasy *Student* atrybut ten może zawierać zbiór identyfikatorów istniejących obiektów klasy *Wykład*. Identyfikatorów tych będzie tyle, na ile wykładów dany student zostanie zapisany. Identyfikator tego samego wykładu pojawi się w zbiorze *Wykłady* dla wszystkich obiektów klasy *Student*, które zostały związane z tym wykładem, czyli dla wszystkich studentów, którzy zapisali się na dany wykład. Każdy wykład ma prowadzącego-wykładowcę. Identyfikator wykładowcy jest rejestrowany jako atrybut *Prowadzący* obiektu klasy *Wykład*. Wykład może mieć najwyżej jednego wykładowcę (co wynika z modelu konceptualnego), a więc atrybut *Prowadzący* obiektu *Wykład* jest skalarem. Wykładowca może prowadzić wiele wykładów – identyfikator tego samego wykładowcy może się pojawić jako atrybut *Prowadzący* w różnych obiektach klasy *Wykład*. Znajomość przez pracownika wielu języków obcych jest rejestrowana w atrybucie *JęzykiObce*, który jest typu zbiorowego, a każdy element tego zbioru jest łańcuchem znakowym.

## V. Wnioski

W artykule opisano założenia uniwersalnego modelu konceptualnego tworzonego na początku procesu projektowania bazy danych, którego główną cechą jest niezależność od modelu logicznego uwzględniającego specyficzne rozwiązania charakterystyczne dla danej klasy modeli. Zaprezentowano także, i omówiono na przykładzie, sposób przekształcenia projektu konceptualnego na logiczny projekt relacyjny i obiektowy, zwracając uwagę na te rozwiązania stosowane w poszczególnych typach modeli, które umożliwiają takie przekształcenie.

Projekt konceptualny bazy danych, będący efektem końcowym procesu projektowania, w którym czynny udział bierze przyszły użytkownik, stanowi platformę, na podstawie, której można stworzyć projekt logiczny uwzględniający specyfikę przyjętego systemu, w którym baza danych będzie implementowana. Uniwersalność modelu konceptualnego i jego niezależność od projektu logicznego pozwalają na zaprojektowanie bazy danych bez konieczności uwzględniania szczegółów konkretnego typu modelu.

Przytoczone przykłady pokazują, że model obiektowy pozwala na projektowanie bazy danych na wyższym poziomie abstrakcji niż relacyjny. Klasy umożliwiają bardziej adekwatne do rzeczywistości reprezentowanie faktów, niż tabele. Projektowanie obiektowe jest więc bardziej naturalne i nie wymaga przeprowadzania tak wielu przekształceń projektu konceptualnego, jak relacyjne. Jednak ze względów historycznych oraz z powodu braku podstaw teoretycznych dla modelu obiektowego, systemy relacyjne obejmują obecnie większą liczbę praktycznych implementacji, niż obiektowe.

Wykorzystanie wspólnego modelu niezależnego od szczegółów implementacji ułatwia konwersję danych przechowywanych w modelach relacyjnych na postać obiektową i na odwrót. Wymaga to podjęcia dodatkowych działań i utworzenia mechanizmów umożliwiających taką konwersję. Można to zrobić w fazie eksploatacji systemu, a operacja ta nie wymaga modyfikacji istniejących w nim rozwiązań. Pozwoli to na dopasowanie danych przechowywanych w modelach relacyjnych do obiektowych języków programowania bez konieczności przebudowy całego systemu.

**Literatura:**


[1] Bancilhon F., Delobel C., Kanellakis P., *Building an Object-Oriented Database System, The Story of O2*, Morgan Kaufmann, 1992

[2] Begg C., Connolly T., *Systemy baz danych. Praktyczne metody projektowania, implementacji i zarządzania*, tom 1 i 2, RM, Warszawa, 2004

[3] Beeri C., *Formal Models for Object-Oriented Databases*, Proc. DOOD Conference, 370–395, 1989

[4] Beynon-Davies P., *Systemy baz danych*, WNT, Warszawa, 2003

[5] Cattel R. G. G., *Object Data Management*, Addison-Wesley, 1994

[6] Cluet S., *Designing OQL: Allowing Objects to be Queried*, Information Systems, 23(5), 279–305, 1998

[7] Codd E., *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM (CACM), Vol. 13, 377–387, 1970

[8] Date C. J., *Relational Database: Selected Writings*, Addison-Wesley, 1986

[9] Date C. J., *Wprowadzenie do systemów baz danych*, WNT, Warszawa, 2000

[10] Lausen G., Vossen G., *Obiektowe bazy danych*, WNT, Warszawa, 2000

[11] Melton J., Simon A., *Understanding the New SQL: A Complete Guide*, Morgan Kauffman, 1993

[12] Subieta K., *Object-Oriented Standards: Can ODMG OQL be Extended to a Programming Language?*, [w:] „Cooperative Databases and Applications", World Scientific, 459–468, 1997

[13] Subieta K., Jodłowski A., Habela P., Płodzien J., *Conceptual Modelling of Business Applications with Dynamic Object Roles*, [w:] „Technologies Supporting Business Solutions", Nova Science Books and Journals, Nowy Jork, 57–84, 2003

[14] Vermeer M. W. W., Apers P. M. G., *Object-Oriented Views of Relational Databases Incorporating Behaviour*, Proc. DASFAA Conference, 26–35, 1995