



# Relacyjne Bazy Danych

Andrzej M. Borzyszkowski  
PJATK/ Gdańsk

materiały dostępne elektronicznie  
<http://szuflandia.pjwstk.edu.pl/~amb>

© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

## Krok 1: odwzorowanie zwykłych encji

- Każdy typ encji w diagramie ER otrzymuje swoją relację
  - np. *Klient*, *Zamówienie*, *Towar*, *Student*, *Nauczyciel*, itp.
- Atrybutami relacji są wszystkie proste atrybuty encji
  - np. *imię*, *nazwisko*, *pesel*, *nr\_zamówienia*, *nazwa towaru*...
  - atrybuty złożone rozpadają się na swoje składowe – nie ma atrybutu *adres*, są atrybuty *miasto*, *ulica*, *kod*
  - atrybuty pochodne, np. *wiek*, w ogóle nie są atrybutami relacji, będą one mogły być wyliczane na bieżąco
  - atrybuty wielokrotne, np. *wykształcenie*, będą obsługiwane inaczej
- Kluczem głównym będzie wybrany klucz z diagramu
  - być może, ale nieczęsto, będzie składał się z wielu atrybutów
  - inne klucze zostaną kluczami alternatywnymi

© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

3

## Odwzorowanie modelu encji i związków w model relacyjny

© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

## Przykład

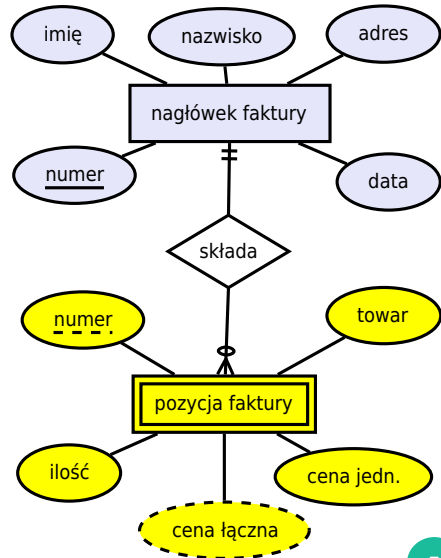
```
create table klient (  
    nr            serial      ,  
    tytuł        char(4)     ,  
    imię         varchar(16) ,  
    nazwisko     varchar(32)  not null,  
    kod_pocztowy char(6)      not null,  
    miasto       varchar(32) ,  
    ulica_dom    varchar(64) ,  
    telefon      varchar(11) ,  
    CONSTRAINT klient_nr_pk PRIMARY KEY(nr) );
```

© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

4

## Krok 2: odwzorowanie słabych encji

- Słabe encje, to encje podporządkowane swoim właścicielom
  - nie mają sensu istnienia bez encji właścicielskiej
  - nie mają swojego klucza głównego
  - np. kolejne pozycje faktury
  - możliwe adresy klientów, wykształcenie studentów, dzieci pracowników, itd.



© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

5

## Krok 2: odwzorowanie słabych encji, c.d.

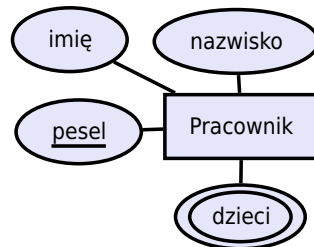
- Każdy typ słabej encji otrzymuje swoją relację wraz z atrybutami prostymi
  - dodatkowo, jednym z atrybutów będzie klucz główny encji właścicielskiej jako klucz obcy relacji
- Kluczem głównym dla słabej encji będzie zestaw: klucz główny encji właścicielskiej plus klucz jej słaby
  - np. *numer faktury* plus *numer kolejny* pozycji faktury
  - albo *numer studenta* plus *typ szkoły* dla studenta (student posiada jedno świadectwo szkoły podstawowej, jedno gimnazjum, jedno szkoły średniej)
  - *pesel* pracownika plus *imię* dziecka (pracownik ma dzieci o różnych imionach)

© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

6

## Słabe encje a atrybuty wielokrotne

- W szczególnie prostych przypadkach zamiast słabych encji można stosować atrybut wielokrotny
  - np. pracownik i lista jego dzieci
  - odpowiada do słabej encji z jednym tylko atrybutem *nazwa* (chyba, że atrybut ten rozkłada się na więcej)

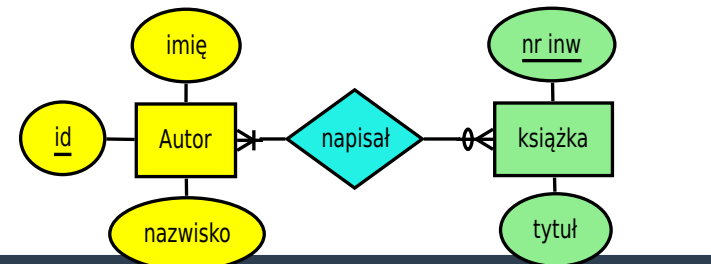


© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

7

## Słabe encje a atrybuty wielokrotne, c.d.

- W innych przypadkach wartości atrybutu mogą powtarzać się dla różnych encji
  - wówczas atrybut wielokrotny nie odpowiada słabej encji
  - raczej jest to silna encja, a krotność oznacza związek wieloznaczny i powinien być odpowiednio potraktowany



© Andrzej M. Borzyszkowski  
Bazy Danych

8

## Krok 3: odwzorowanie związków 1:N

- Założenie: dane są dwa typy encji *S* oraz *T*, pomiędzy nimi związek jednoznaczny
  - tzn. dla jednej encji typu *S* przypisanych jest wiele encji typu *T*
- W relacji dla typu encji po stronie „wiele”, *T*, dodajemy klucz obcy wskazujący na klucz główny relacji po stronie „jeden”, *S*
  - jeśli sam związek miał atrybuty, to dołączamy je do *T*
- Jest to najczęstszy związek w projektach
  - *Klient* i jego *Zamówienia* (klucz obcy dla zamówienia)
  - *Zamówienie* i jego *Pozycje* (klucz obcy dla pozycji)
  - *Towar* i *Pozycje*, w których występuje (klucz obcy dla pozycji)
  - *Nauczyciel* i nauczane *Przedmioty* (klucz obcy dla przedmiotu)
- Klucz musi być określony (NOT NULL) jeśli po stronie „jeden” nie dopuszczamy zera, czyli jeden oznacza *dokładnie jeden*

© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

9

## Krok 4: odwzorowanie związków 1:1

- Założenie: dane są dwa typy encji *S* oraz *T*, pomiędzy nimi związek jednojednoznaczny
- Rozwiązanie pierwsze – klucz obcy, który jest też kluczem kandydującym
  - klucz obcy zapewnia związek 1:N, dodatkowy warunek kluczowości zapewnia, że  $N=1$
  - jeśli każde *S* ma przyporządkowane *T*, to można zaprojektować klucz obcy po stronie *S*
  - na ogół jeden z typów ma pełen udział w związku, a drugi niekoniecznie
  - np. *Przedmiot* ma przypisany *Termin*, ale pozostały wolne terminy
  - a może *Termin* jest zarezerwowany dla *Przedmiotu*
  - podobnie *Zapas* ma przypisany *Towar*, którego dotyczy

© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

11

## Tabele, SQL

```
create table zamowienie (  
  Nr          serial,          typ automatycznego numerowania  
  klient_nr   int              not null,    atrybut musi  
  data_zlozenia date          not null,    być określony  
  data_wyslki  date,  
  koszt_wyslki numeric(7,2),          nazwa atrybutu  
  CONSTRAINT  zamowienie_nr_pk PRIMARY KEY(nr),  
              nazwa warunku integralności, najczęściej  
              w postaci rozwiniętej: nazwa tabeli_atrybutu_pk  
              nazwa atrybutu będącego kluczem głównym  
  CONSTRAINT klient_fk FOREIGN KEY(klient_nr)  
              nazwa atrybutu będącego kluczem obcym  
              REFERENCES klient(nr), nazwa tabeli  
              i atrybutu w tej tabeli wskazywanego przez klucz obcy  
              ON UPDATE CASCADE ON DELETE CASCADE,  
              określa zachowanie systemu w razie naruszenia  
              integralności referencyjnej  
);
```

© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

10

## Tabele, przykład c.d.

```
create table towar (  
  nr          serial ,  
  opis        varchar(64) ,      not null,  
  koszt       numeric(7,2)      not null,  
  cena        numeric(7,2),  
  CONSTRAINT  towar_nr_pk PRIMARY KEY(nr)  
);  
create table zapas (  
  towar_nr    integer not null,  
  ilosc       integer not null,  
  CONSTRAINT  towar_nr_fk FOREIGN KEY(towar_nr)  
              REFERENCES towar(nr)  
              ON UPDATE NO ACTION ON DELETE NO ACTION  
              jest domyślnym zachowaniem systemu w razie naruszenia  
              integralności referencyjnej  
  CONSTRAINT  zapas_towar_nr_pk PRIMARY KEY(towar_nr)  
);  
• klucz obcy i jednocześnie główny realizuje związek 1:1
```

© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

12

## Odwzorowanie związków 1:1, c.d.

- Klucz obcy, który jest jednoznaczny można zastosować po stronie typu z niepełnym udziałem jeśli dopuszczalne są wartości NULL dla klucza obcego
  - np. *Termin* ma przypisany *Przedmiot*, chyba, że pozostaje wolny
- Można stosować klucze obce dla obu relacji
  - ale prowadzi to do nadmiarowości informacji i trzeba sięgać specjalnych rozwiązań (wyzwalacze) by to obsłużyć
  - nawet jeśli sytuacja jest symetryczna, to jeden z kluczy musi dopuszczać NULL (dla tej encji, która jest wstawiana wcześniej)

© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

13

## Tabele, przykład c.d.

```
CREATE TABLE przedmiot_termin (  
  kod          serial      PRIMARY KEY,  
  rodzaj       varchar(20)  not null,  
  nazwa        varchar(50)  not null,  
  godziny      int          not null,  
  -- teraz atrybuty terminu  
  dzien_tyg    int          ,  
  godzina      int          ,  
  sala         varchar(5)   ,  
  -- nr legitymacji nauczyciela prowadzącego  
  nr_leg       char(7)      REFERENCES nauczyciel  
  ON UPDATE SET NULL ON DELETE SET NULL, określa zachowanie systemu w razie naruszenia integralności referencyjnej  
  -- atrybuty encji termin muszą być unikalne  
  CONSTRAINT przedmiot_un UNIQUE(dzien_tyg, godzina, sala)  
);
```

- związek 1:1 zachodzi ponieważ *obie* encje, *przedmiot* i *termin* występują najwyżej jeden raz
- klucz obcy może mieć wartość NULL, brak odniesienia

© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

15

## Odwzorowanie związków 1:1, drugie rozwiązanie

- Scalenie dwu relacji
  - zamiast odrębnych relacji dla każdego z typów encji projektujemy jedną relację zawierającą atrybuty obu encji
  - jeśli jeden z udziałów typów w związku jest niepełny, to na pewno musimy dopuszczać nieokreśloność atrybutów
- Przykład: relacja *Przedmiot-Termin*, której typową krotką będzie pełna informacja o przedmiocie i o jego terminie
  - jeśli dopuszczamy przedmioty (jeszcze) niezaplanowane, to atrybuty terminu będą nieokreślone

© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

14

## Krok 5: odwzorowanie związków M:N

- Założenie: dane są dwa typy encji *S* oraz *T*, pomiędzy nimi związek wieloznaczny
- Rozwiązanie: dla związku tworzona jest nowa relacja
  - atrybutami relacji są klucze obce wskazujące na klucze główne w *S* oraz *T*, oba klucze obce muszą być określone (NOT NULL)
  - oraz atrybuty związku, jeśli takie występowały
- Przykład: *Student <Zalicza> Przedmiot* powoduje zaprojektowanie relacji *Zalicza* z dwoma kluczami obcymi
  - i być może atrybutami *ocena*, *data egzaminu*, itp.
- Kluczem głównym w nowej relacji jest zestaw kluczy obcych

© Andrzej M. Borzyszkowski  
Relacyjne Bazy Danych

16

## Tabele, przykład c.d.

```
create table pozycja
(
  zamowienie_nr integer not null,
  towar_nr       integer not null,
  ilosc          integer not null,
  CONSTRAINT pozycja_pk
    PRIMARY KEY(zamowienie_nr, towar_nr),
  CONSTRAINT pozycja_zamowienie_nr_fk
    FOREIGN KEY(zamowienie_nr)
      REFERENCES zamowienie(nr)
      ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT pozycja_towar_nr_fk
    FOREIGN KEY(towar_nr)
      REFERENCES towar(nr)
);
```

- tabela z **dwoma** kluczami obcymi realizuje związek wieloznaczny (dwuargumentowy)

© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

17

## Krok 6: odwzorowanie związków niebinarnych

- Rozwiązanie „nowa relacja” jest dobre również dla związków o większej liczbie zaangażowanych typów encji
  - np. *Klient* kupuje *Polisę* u *Agent*a
  - Lekarz* wykonuje zabieg u *Pacjenta* w asyście *Pielęgniarki* w danym *Gabiniecie*/*Terminie* używając *Sprzętu*
- Kluczem głównym jest często zestaw wszystkich kluczy obcych
  - ale czasami pewne zestawy są w naturalny sposób unikalne
  - np. w relacji *Zabieg* klucz obcy wskazujące gabinet i termin razem z kluczem na lekarza jest unikalny
  - podobnie jak w połączeniu z pielęgniarką czy pacjentem
  - jeden z zestawów może być kluczem głównym, pozostałe są kandydujące

© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

19

## Relacja dla związku

- Rozwiązanie „nowa relacja dla związku” jest zawsze skuteczne
  - jeśli związek jest typu N:M, to są dwa klucze obce
  - jeśli związek jest typu 1:N, to klucz obcy po stronie 1: będzie kluczem kandydującym (UNIQUE/PRIMARY KEY)
  - jeśli związek jest typu 1:1, to każdy z kluczy obcych będzie kluczem kandydującym
- Rozwiązanie jest szczególnie polecane, gdy związek ma niewiele elementów (tzn. udziały obu typów są mocno niepełne)
  - np. projekt studencki może być wykonywany pojedynczo, a czasami przez dwie osoby
  - zamiast projektować klucz obcy dla *Studenta* „druga osoba” lepiej zaprojektować osobną relację z dwoma kluczami
  - nawet jeśli normą mają być pary, to nowa relacja może być łatwiejszym rozwiązaniem niż zmiana istniejącego projektu

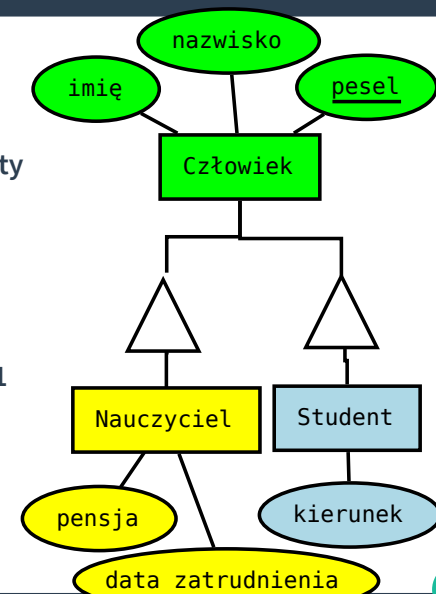
© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

18

## Związek „jest”

- Projekt przewiduje wspólną generalizację kilku typów
  - np. *Student* oraz *Nauczyciel* mają wspólne niektóre atrybuty
  - mogą być uogólnieni do *Człowiek*
  - i jednocześnie zachować atrybuty charakterystyczne
- Generalizacja oznacza związek 1:1
- Generalizacja może być
  - kompletna lub nie
  - rozłączna lub nie



© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

20

## Związek „jest”, c.d.

- Rozwiązanie 1: Jedna relacja dla ogólnego typu oraz relacje dla podtypów z kluczem obcym będącym kluczem kandydującym
  - można zapisać dane spoza wszystkich konkretnych klas
  - można dopuścić, że encja należy do kilku klas
- Rozwiązanie 2: Odrębne relacje z kompletem atrybutów
  - *Student* i *Nauczyciel* mają imię, nazwisko, pesel i dalsze atrybuty
  - nie można zapisać danych osób spoza tych klas
  - można dopuścić należenie do kilku klas (kosztem kopiowania danych)

© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

21

## Podsumowanie

- |                            |   |
|----------------------------|---|
| • Typ zwykłych encji       | • Relacja z atrybutami  |
| • Encja podporządkowana    | • Relacja z kluczem obcym wskazującym na właściciela  |
| • Związek jednoznaczny     | • Klucz obcy po stronie wiele   |
| • Związek 1:1              | • Klucz obcy = klucz <ul style="list-style-type: none"><li>– lub scalenie dwu relacji</li></ul> |
| • Związek wieloznaczny     | • Relacja z 2 kluczami obcymi   |
| • Związek $n$ -składnikowy | • Relacja z $n$ kluczami obcymi   |
| • Atrybut prosty           | • Atrybut   |
| • Atrybut złożony          | • Kilka atrybutów   |
| • Atrybut wielowartościowy | • Relacja z kluczem obcym wskazującym na właściciela  |

© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

23

## Związek „jest”, c.d.

- Rozwiązanie 3: Jedna relacja ze wszystkimi atrybutami i dodatkowo atrybutem wskazującym na typ pochodzenia (dyskryminator)
  - każda encja musi należeć najwyżej do jednej klasy
  - można wymusić, że dokładnej do jednej klasy
  - być może będzie dużo wartości niekreślonych
  - ale niektóre atrybuty mogą być wspólne (np. nr legitymacji)
- Rozwiązanie 4: Jedna relacja ze wszystkimi atrybutami oraz z flagami wskazującymi, czy encja należy do tej klasy
  - encje mogą należeć do wielu klas jednocześnie lub do żadnej

© Andrzej M. Borzyszkowski

Relacyjne Bazy Danych

22