



# Bazy Danych

Andrzej M. Borzyszkowski

Instytut Informatyki  
Uniwersytetu Gdańskiego

materiały dostępne elektronicznie  
<http://inf.ug.edu.pl/~amb>

Bazy Danych © Andrzej M. Borzyszkowski

## Język SQL, cz.1, definiowanie danych (*data definition language*)

Bazy Danych © Andrzej M. Borzyszkowski

2

### Standard SQL

- Standard nieformalnie nazywany SQL/92
  - pełna nazwa: Międzynarodowy Standardowy Język Baz Danych SQL (1992)
  - skrót od *Structured Query Language*
- Istniejące implementacje nie implementują w pełni powyższego standardu
  - ale rozszerzają niektóre aspekty standardu, czyli *nadzbior podzbioru*
- Język deklaracyjny - użytkownik deklaruje swoje potrzeby, optymalizator przekształca zapytanie na ciąg instrukcji
- Zawiera w sobie język definiowania danych i język manipulowania danymi
  - i dodatkowo język zarządzania użytkownikami, określania zabezpieczeń, sterowania transakcjami, ...

Bazy Danych © Andrzej M. Borzyszkowski

3

### SQL, kilka uwag ogólnych

- Relacja nazywana jest tabelą (*table*), może zawierać powtórzenia i oczywiście ma ustaloną kolejność
- Projekt bazy danych składa się głównie z zestawu tabel, są one zgrupowane w *schemacie*
- Standard wymaga by wielkość liter w nazwach nie grała roli
  - zasadniczo wszystkie słowa są konwertowane na duże litery
  - napisy w cudzysłowach traktowane są dosłownie z rozróżnieniem wielkości liter
- Standard nie określa sposobu kończenia zapytania,
  - w większości SZBD jest to średnik
- Każdy element musi mieć nazwę, nawet gdy nie mamy zamiaru odwoływać się do niego

Bazy Danych © Andrzej M. Borzyszkowski

4

# Typy wbudowane

- Jest wiele typów wbudowanych, najważniejsze z nich:
  - **CHAR(\_)**, **VARCHAR(\_)**
  - **INTEGER**, **SMALLINT**
  - **DATE**, **TIME**, **TIMESTAMP**, obsługa czasu i daty
  - **BOOLEAN**, wartości np. 't', TRUE, '1', 'y', 'yes', ( [SQL/99](#) )
  - **NUMERIC(,\_)**, np. NUMERIC (7,2), 7 cyfr, w tym 2 po przecinku
  - **FLOAT(\_)**, np. FLOAT(15), 15 cyfr znaczących
  - **BIT**, **VARBIT**, wartości np. B'10011101'
  - **MONEY**, to samo co NUMERIC (9,2)

© Andrzej M. Borzyszkowski

Bazy Danych

5

# Definiowanie tabeli

- **CREATE TABLE *nazwa\_tabeli***  
***lista-( definicja\_kolumny [ wartość\_domyślna ]***  
**| [ *definicja\_klucza\_kandydującego* ]**  
**| [ *definicja\_klucza\_obcego* ]**  
**| [ *definicja\_warunku\_poprawności* ] ) ;**
- ***definicja\_kolumny ::= nazwa\_kolumny nazwa\_dziedziny***
- ***nazwa\_dziedziny ::= typ\_wbudowany | nazwa\_zdefiniowana***  
 **::= "to jest"**  
**| "albo"**  
**[ ] "alternatywnie"**
- Wartość domyślna może zmienić wartość podaną w definicji dziedziny, brak definicji wartości domyślnej oznacza **NULL**
- Można żądać, by atrybut był zawsze określony: **NOT NULL**

© Andrzej M. Borzyszkowski

Bazy Danych

6

# Definiowanie tabel, klucze

- ***definicja\_klucza\_kandydującego ::=***  
 ***UNIQUE ( lista\_kolumn ) |***  
 ***PRIMARY KEY ( lista\_kolumn )***
  - lista kolumn w obu przypadkach jest niepusta
  - najwyżej jeden klucz może być określony jako główny (PRIMARY KEY)
  - jeśli występuje klucz główny, to wszystkie atrybuty tego klucza zyskują warunek poprawności NOT NULL
  - klucz alternatywny dopuszcza wartości NULL,
    - PostgreSQL - nie naruszają one warunku
    - SQL92 (np. MS SQL Server) - naruszają warunek
- Warunki poprawności można opcjonalnie nazwać:  
***CONSTRAINT nazwa definicja\_klucza\_kand.***

© Andrzej M. Borzyszkowski

Bazy Danych

7

# Definiowanie tabel, klucze obce

- ***definicja\_klucza\_obcego ::=***  
 ***FOREIGN KEY ( lista\_kolumn )***  
 ***REFERENCES tabela\_bazowa [ ( lista\_kolumn ) ]***  
 ***[ ON DELETE opcja ]***  
 ***[ ON UPDATE opcja ]***
  - nie jest wymagane podanie listy kolumn, jeśli klucz obcy odwołuje się do klucza o tej samej nazwie
- ***opcja ::= NO ACTION | CASCADE | SET NULL | SET DEFAULT***
- Warunki poprawności można opcjonalnie nazwać:  
***CONSTRAINT nazwa definicja\_klucza\_obcego***

© Andrzej M. Borzyszkowski

Bazy Danych

8

# Definiowanie tabel, warunki poprawności

- **definicja\_warunku\_poprawności ::=**  
**CHECK ( wyrażenie\_warunkowe )**
  - **wyrażenie\_warunkowe** może być dowolnie skomplikowane, nie musi ograniczać się do danej tabeli, musi być określone dla każdego wiersza tabeli
  - więzy poprawności są spełnione, jeśli powyższe **wyrażenie\_warunkowe** ma wartość "true" dla każdego wiersza tabeli
  - system zarządzania bazą danych nie zezwoli na wprowadzenie danych czy aktualizację danych takie, że więzy poprawności nie są spełnione
  - kolejność sprawdzania warunków jest nieokreślona
- Warunki poprawności można opcjonalnie nazwać:  
**CONSTRAINT nazwa definicja\_warunku\_poprawności**

© Andrzej M. Borzyszkowski

Bazy Danych

9

# Zmiana definicji tabeli podstawowej

- **ALTER TABLE nazwa\_tabeli operacja;**
- Przykład
  - ALTER TABLE klient**  
**ADD COLUMN rabat INT**  
**DEFAULT 0;**
  - **operacja** może oznaczać
    - dodanie/usunięcie/zmiana nazwy kolumny,
    - zmiana dotychczasowej wartości domyślnej w kolumnie,
    - dodanie/usunięcie warunku poprawności
  - ALTER TABLE klient**  
**ALTER COLUMN telefon DROP NOT NULL**

© Andrzej M. Borzyszkowski

Bazy Danych

10

# Usuwanie tabeli podstawowej

- **DROP TABLE nazwa\_tabeli [ RESTRICT | CASCADE ];**
  - jeżeli wybrano **RESTRICT** i tabela podstawowa występuje w jakiegokolwiek definicji perspektywy, to instrukcja **DROP TABLE** nie powiedzie się
  - jeżeli wybrano **CASCADE**, to instrukcja **DROP TABLE** powiedzie się i usunie daną tabelę wraz ze wszystkimi perspektywami bazującymi na tej tabeli oraz więzami poprawności

© Andrzej M. Borzyszkowski

Bazy Danych

11

# Odwzorowanie modelu encji i związków w model relacyjny

© Andrzej M. Borzyszkowski

Bazy Danych

12

## 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

Bazy Danych

13

## Przykład

```
create table klient (  
  nr          integer      ,  
  tytul       char(4)      ,  
  imie        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) ) ;
```

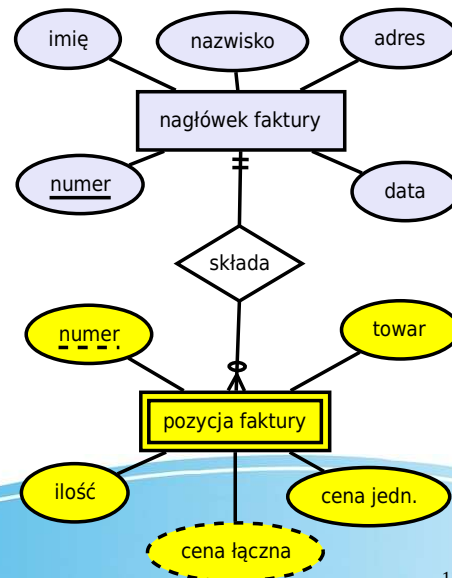
© Andrzej M. Borzyszkowski

Bazy Danych

14

## 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

Bazy Danych

15

## 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

Bazy Danych

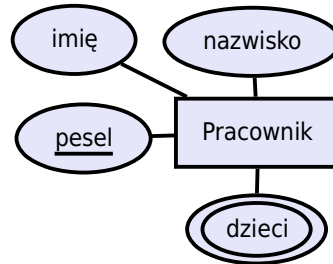
16



# 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)



- Metoda stosowana dla słabych encji ma tu zastosowanie
  - relacja z kluczem obcym wskazującym na encję, z której pochodzi atrybut wielokrotny
  - kluczem głównym jest klucz obcy plus atrybut *nazwa*

17

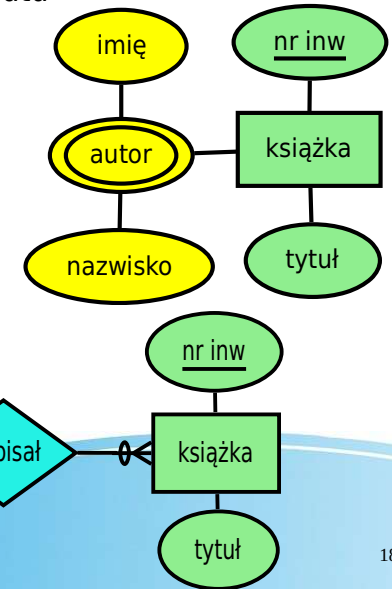
© Andrzej M. Borzyszkowski

Bazy Danych

# 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 ew. krotność oznacza związek wieloznaczny i powinien być odpowiednio potraktowany



18

© Andrzej M. Borzyszkowski

Bazy Danych

## Krok 3: odwzorowanie związków 1:N

- Założenie: dane są dwa typy encji *S* oraz *T*, pomiędzy nimi związek jednoznaczny
  - dokładniej, 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)

19

© Andrzej M. Borzyszkowski

Bazy Danych

## Krok 3: odwzorowanie związków 1:N c.d.

- 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 (kolejny klucz obcy dla pozycji)
  - Nauczyciel* i nauczane *Przedmioty* (klucz obcy dla przedmiotu)
  - Czytelnik* i czytane *Książki* (klucz obcy dla książki, razem z datą wypożyczenia)
- Klucz musi być określony (NOT NULL) jeśli po stronie jeden oznacza *dokładnie jeden* (nie dopuszczamy zera)

20

© Andrzej M. Borzyszkowski

Bazy Danych

# Tabele, SQL

```
create table zamowienie (
    nr integer,
    klient_nr integer not null,
    data_zlozenia date not null,
    data_wyslki date,
    koszt_wyslki numeric(7,2),
    CONSTRAINT zamowienie_nr_pk PRIMARY KEY(nr),
    CONSTRAINT klient_fk FOREIGN KEY(klient_nr)
    REFERENCES klient(nr),
    ON UPDATE NO ACTION ON DELETE NO ACTION
);
```

*nazwa tabeli*

*nazwa atrybutu*

*nazwa warunku integralności, najczęściej w postaci rozwiniętej: nazwa tabeli\_atributu\_pk*

*nazwa atrybutu będącego kluczem głównym*

*nazwa atrybutu będącego kluczem obcym*

*nazwa tabeli i atrybutu w tej tabeli wskazywanego przez klucz obcy*

*jest domyślnym zachowaniem systemu w razie naruszenia integralności referencyjnej, można ten tekst opuścić*

© Andrzej M. Borzyszkowski

Bazy Danych

21

## Tabele, przykład c.d.

```
create table towar (
    nr integer,
    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 CASCADE ON DELETE CASCADE,
    CONSTRAINT zapas_towar_nr_pk PRIMARY KEY(towar_nr)
);
```

• klucz obcy i **jednocześnie** główny realizuje związek 1:1

© Andrzej M. Borzyszkowski

Bazy Danych

23

## 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

Bazy Danych

22

## 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 nieobsadzony
- 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

Bazy Danych

24

# 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

Bazy Danych

25

# Tabele, przykład c.d.

```
CREATE TABLE przedmiot_termin (  
  kod          integer          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
- może istnieć przedmiot bez przypisanego terminu
- ale nie może istnieć termin bez przypisanego przedmiotu

© Andrzej M. Borzyszkowski

Bazy Danych

26

# 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*
  - 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

Bazy Danych

27

# 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

Bazy Danych

28

## Relacja dla związku

- Rozwiązanie „nowa relacja dla związku” jest zawsze skuteczne (raczej nie zawsze wskazane)
  - 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

© Andrzej M. Borzyszkowski

Bazy Danych

29

## Relacja dla związku c.d.

- 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* „drugi autor” lepiej zaprojektować osobną relację z dwoma kluczami
- nawet jeśli normą mają być pary, to nowa relacja może być łatwiejszym rozwiązaniem niż zmiana istniejącego schematu

© Andrzej M. Borzyszkowski

Bazy Danych

30

## Krok 6: odwzorowanie związków niebinarnych

- Rozwiązanie „nowa relacja” jest dobre również dla związków o większej liczbie zaangażowanych typów encji
  - np. *Klient* kupuje *Polisę* u *Agenta*
  - *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

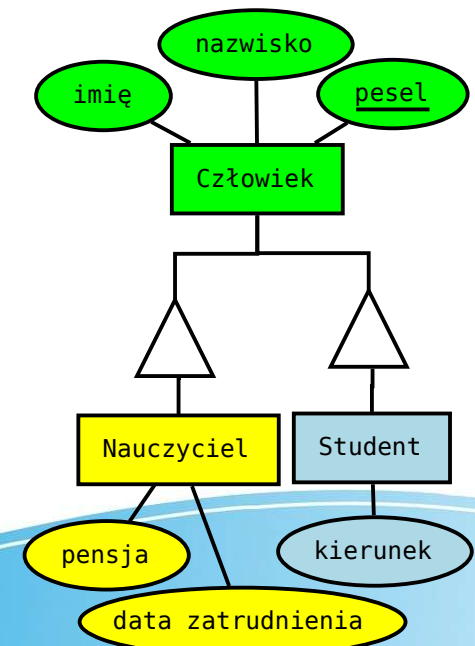
© Andrzej M. Borzyszkowski

Bazy Danych

31

## 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 jest naturalnie związkiem 1:1
- Generalizacja może być
  - kompletna lub nie
  - rozłączna lub nie



© Andrzej M. Borzyszkowski

Bazy Danych

32



## 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 (generalizacja niekompletna)
  - można dopuścić, że encja należy do kilku klas (generalizacja nierozłączna)
- 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 (generalizacja kompletna)
  - można dopuścić należenie do kilku klas (kosztem kopiowania danych)

33

© Andrzej M. Borzyszkowski

Bazy Danych

## 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 (generalizacja rozłączna)
  - 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 (generalizacja niekompletna i nierozłączna)

34

© Andrzej M. Borzyszkowski

Bazy Danych

## Podsumowanie

- Typ zwykłych encji
- Encja podporządkowana
- Związek jednoznaczny
- Związek 1:1
- Związek wieloznaczny
- Związek  $n$ -składnikowy
- Atrybut prosty
- Atrybut złożony
- Atrybut wielowartościowy
- Relacja z atrybutami
- Relacja z kluczem obcym wskazującym na właściciela
- Klucz obcy po stronie wiele
- Klucz obcy = klucz
  - lub scalenie dwu relacji
- Relacja z 2 kluczami obcymi
- Relacja z  $n$  kluczami obcymi
- Atrybut
- Kilka atrybutów
- Relacja z kluczem obcym wskazującym na właściciela

35

© Andrzej M. Borzyszkowski

Bazy Danych