# BAZY DANYCH WPROWADZENIE

Wykład 1 – 2

Prowadzący: dr Paweł Drozda

# Informacje Ogólne

- Konsultacje

  Środa 11:30 – 13:00

  pokój E 0/5

- Zaliczenie ćwiczeń
  - Projekt na zadany temat (grupy 2-osobowe)
- Egzamin
  - praktyczny

# Program Wykładu

- Wprowadzenie
- Relacyjny model danych
- Modelowanie baz danych (diagramy związków encji)
- Przekształcanie modelu związków encji do modelu relacyjnego
- Normalizacja
- Język baz danych SQL
- Fizyczna organizacja danych
- Transakcje
- Zarządzanie uprawnieniami

# Literatura

- J. Ullman, J. Widom „Podstawowy wykład z systemów baz danych”
- http://wazniak.mimuw.edu.pl
- Theriault, Carmichael „Oracle DBA”
- Pribyl, „Oracle PL/SQL. Wprowadzenie”
- Dokumentacja, Tutoriale Oracle

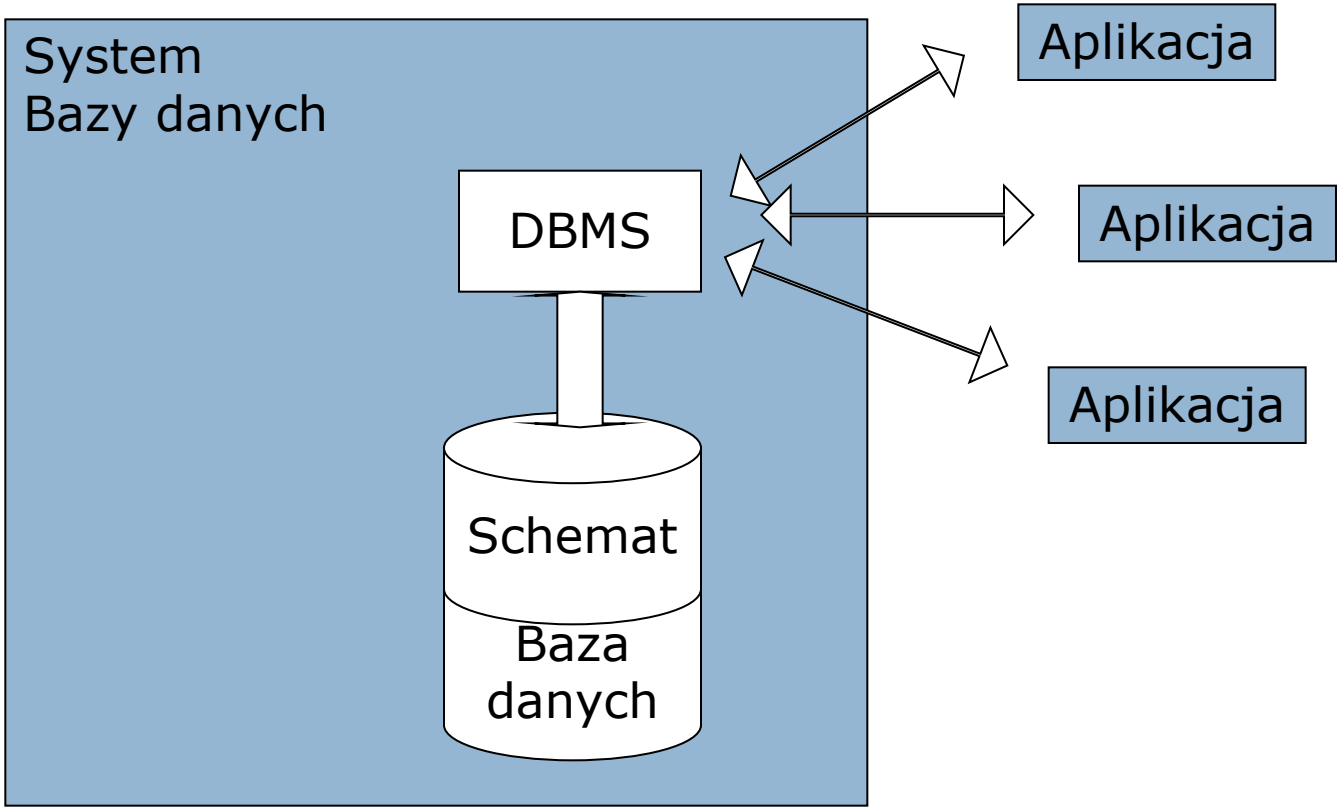
# Plan Wykładu

- Podstawowe pojęcia
- System zarządzania bazami danych (DBMS)
- Właściwości baz danych
- Funkcje baz danych
- Modele danych

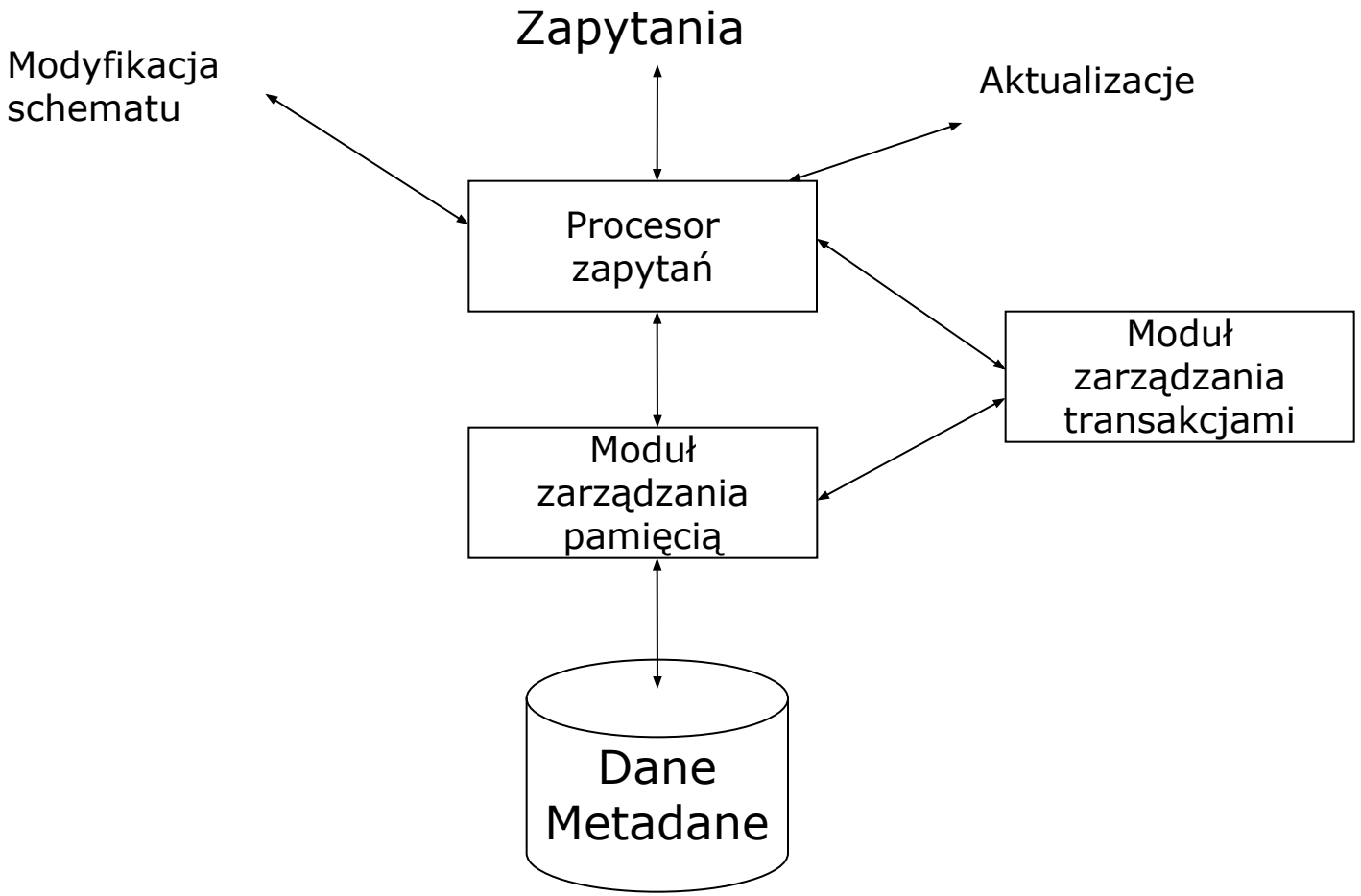
# Podstawowe pojęcia

- **Baza danych** – zbiór informacji opisujący wybrany fragment rzeczywistości. Np. Dla sklepu: dane dotyczące sprzedawanych towarów w sklepie, klientów sklepu, pracowników, zamówień
- **Schemat baz danych** – określa w jaka powinna być struktura danych oraz w jaki sposób dane są powiązane
- **System zarządzania bazą danych (DBMS)** – zbiór narzędzi pozwalający na dostęp oraz na zarządzanie jedną lub wieloma bazami danych
- **System baz danych** – baza danych + DBMS
- **Model danych** – zbiór ogólnych zasad posługiwania się danymi

# System Baz Danych



# System zarządzania bazą danych



# Właściwości bazy danych (1)

- Współdzielenie danych – wielu użytkowników tej samej bazy
- Integracja danych – baza nie powinna mieć powtarzających się bądź zbędnych danych
- Integralność danych – dokładne odzwierciedlenie obszaru analizy
- Trwałość danych – dane przechowywane przez pewien czas

# Właściwości bazy danych (2)

- Bezpieczeństwo danych – dostęp do bazy lub jej części przez upoważnionych użytkowników
- Abstrakcja danych – dane opisują tylko istotne aspekty obiektów świata rzeczywistego
- Niezależność danych – dane niezależnie od aplikacji wykorzystujących te dane

# Modele Danych

- Dla każdego modelu należy określić
  - Definicja danych
  - Operowanie danymi
  - Integralność danych

# Relacyjny Model Danych

# Definicja danych

- **Relacja –** dwuwymiarowa tabela, jedyna struktura danych w modelu relacyjnym
- Każda relacja posiada **atrybuty –** kolumny. Opisują dane umieszczane w relacji
- **Schemat relacji –** nazwa relacji wraz z atrybutami

  **Przykład schematu:** Miasto (id, nazwa, id_regionu)
- **Krotki –** wiersze relacji zawierające dane. Każdy atrybut ma swój odpowiednik w krotce

# Definicja danych – klucze główne

- każda relacja musi posiadać klucz główny
- jedna lub więcej kolumn identyfikujących jednoznacznie każdy wiersz tabeli
- Klucz kandydujący – atrybut lub zbiór atrybutów identyfikujących wiersze tabeli (musi być jednoznaczny i nie zawierać wartości *null*)
- Klucz główny wybierany spośród kluczy kandydujących

# Definicja danych

- Dziedzina – zbiór wszystkich możliwych wystąpień atrybutu (np. ocena_z_egzaminu wartości od 2 do 5 – dziedzina 2-5) – każdy atrybut posiada dziedzinę
- Klucz obcy – kolumna bądź kolumny będące kluczem głównym w innej tabeli, sposób łączenia tabel (np. numer studenta w tabeli Studenci i w tabeli Zaliczenia)
- Wartość *null* – nieznana informacja (np. brak numeru telefonu)

# Operowanie danymi

- Algebra relacyjna – zbiór sześciu operatorów do wyszukiwania danych (selekcja, rzut, złączenie, suma, przecięcie, różnica)
- Operacje dynamiczne na relacjach
  - INSERT – wstawianie
  - DELETE – usuwanie
  - UPDATE – modyfikowanie

# Integralność danych

- Integralność danych zapewnia dokładne odbicie rzeczywistości w bazie danych
- W modelu relacyjnym istnieją dwa rodzaje integralności wewnętrznej
  - integralność encji
  - integralność referencyjna

# Integralność encji

- Dotyczy kluczy głównych
  - Każda relacja musi mieć klucz główny
  - Klucz główny musi być jednoznaczny i nie może zawierać wartości *null* (co skutkuje jednoznacznością krotek w relacji)
  - Przykład

*kluczem głównym w tej relacji może być Nr_prac, nazwisko lub imię*

PRACOWNICY

| Nr_prac | Nazwisko | Imię |
|---|---|---|
| 1 | Golał | Jan |
| 2 | Resko | Paweł |
| 3 | Janik | Tadeusz |
| 4 | Ferel | Michał |

# Integralność referencyjna

- Dotyczy kluczy obcych
  - dwie możliwości (w zależności od konkretnej bazy danych)

  - Wartość klucza obcego musi odwoływać się do wartości klucza głównego w tabeli w bazie danych

  - Wartość klucza obcego może być *null*

    - Wymuszenie istnienia odniesienia każdego wiersza – parametr *not null*

# Integralność referencyjna

- Przykład

| Nr_prac | Nazwisko | Imię |
|---|---|---|
| 1 | Golał | Jan |
| 6 | Resko | Paweł |
| 3 | Janik | Tadeusz |
| 4 | Ferel | Michał |

| IdPrzed | Przedmiot | Prowadzący |
|---|---|---|
| 2 | Bazy danych | null |
| 5 | Filozofia | 4 |
| 6 | Analiza matematyczna | 3 |
| 3 | Statystyka | 3 |

Integralność referencyjna zachowana, jeśli są dopuszczane wartości *null* klucza obcego (klucze obce mogą należeć do zbioru {1,3,4,6})

# Zachowanie integralności referencyjnej

- Określenie więzów propagacji – określają co ma się stać z tabelą przy modyfikacji powiązanej tabeli
  - Ograniczone usuwanie – usunięcie krotki z kluczem głównym możliwe w momencie, gdy klucz główny nie ma wystąpień jako klucz obcy

    Dla poprzedniego przykładu – z tabeli pracownicy można usunąć pracowników o numerach 1 i 6. Pozostali mogą zostać usunięci dopiero w momencie gdy zostaną usunięte odpowiednie krotki w powiązanej tabeli

# Zachowanie integralności referencyjnej

- Kaskadowe usuwanie

  Przy usunięciu wiersza z kluczem głównym zostają usunięte wszystkie wiersze z tym kluczem z relacji powiązanej

  Jeśli usuniemy z tabeli PRACOWNICY pracownika o numerze 3 – zostaną usunięte Przedmioty o numerach 6 i 3 z tabeli PRZEDMIOTY
- Wstaw *null* – przy usunięciu krotki z kluczem głównym zostają wstawione wartości *null* zamiast klucza obcego
- Wstaw *default* – przy usuwaniu wstawia wartość domyślną

# Integralność dodatkowa

- Definiowana przez użytkownika – specyficzna dla każdej bazy danych
  - Przykład

    Możemy wymusić, że każdy pracownik musi prowadzić jakieś zajęcia

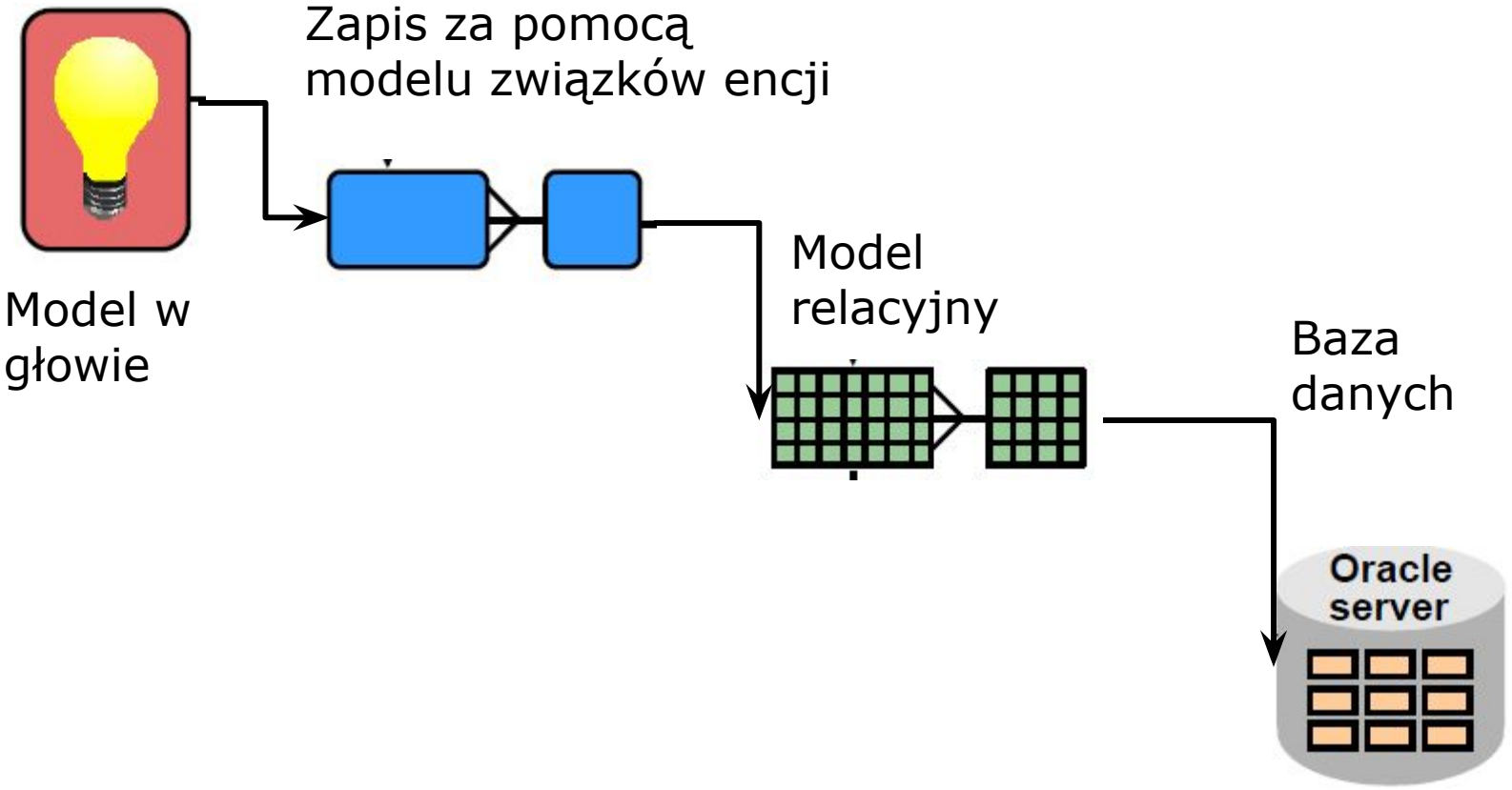
    CONSTRAINT (Project PRACOWNICY(Nr_prac)) – (Project PRZEDMIOTY(Prowadzący)) is empty

# MODELOWANIE
# MODEL ZWIĄZKÓW ENCJI

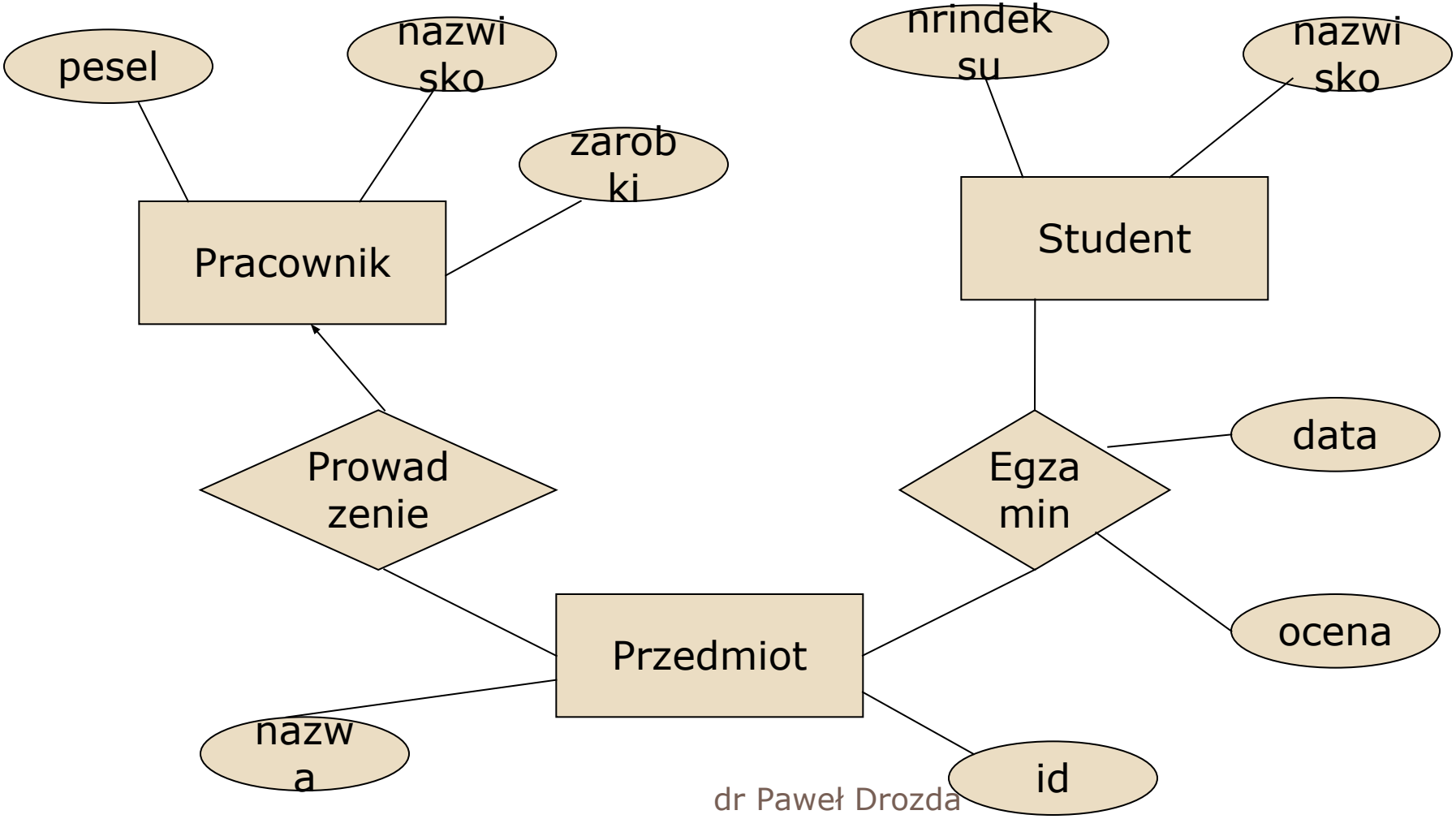
# Modelowanie

- Odwzorowanie obiektów rzeczywistych w systemie informatycznym
- Dwa typy modeli:
  - Konceptualny
    - Model związków encji
    - Model UML
  - Implementacyjny
    - Relacyjny
    - Obiektowy
    - Obiektowo-relacyjny

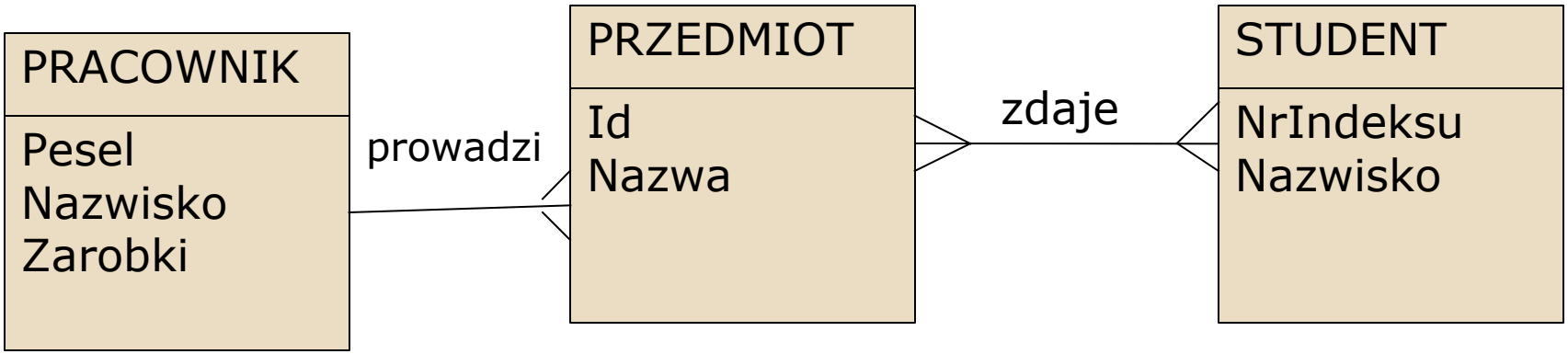
# Etapy tworzenia bazy danych



# Model związków encji – przykład – notacja Chena



# Przykład – notacja Barkera

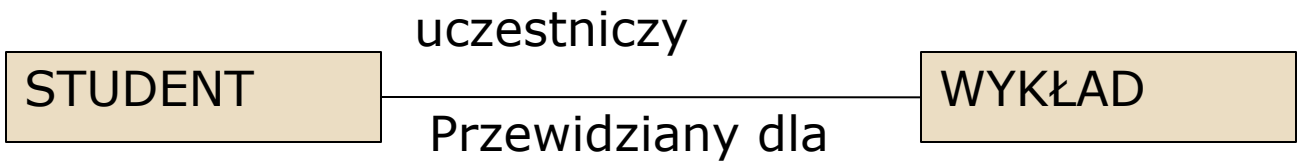


# Reguły modelowania encji

- Unikalność nazw
- Atrybuty
- Związki między encjami
- Obiekt reprezentowany tylko przez jedną encję
- Nazwa – rzeczownik w liczbie pojedynczej

# Związki encji

- Opisują połączenia pomiędzy encjami
- Powiązane dwie lub więcej encji
- Przykład:

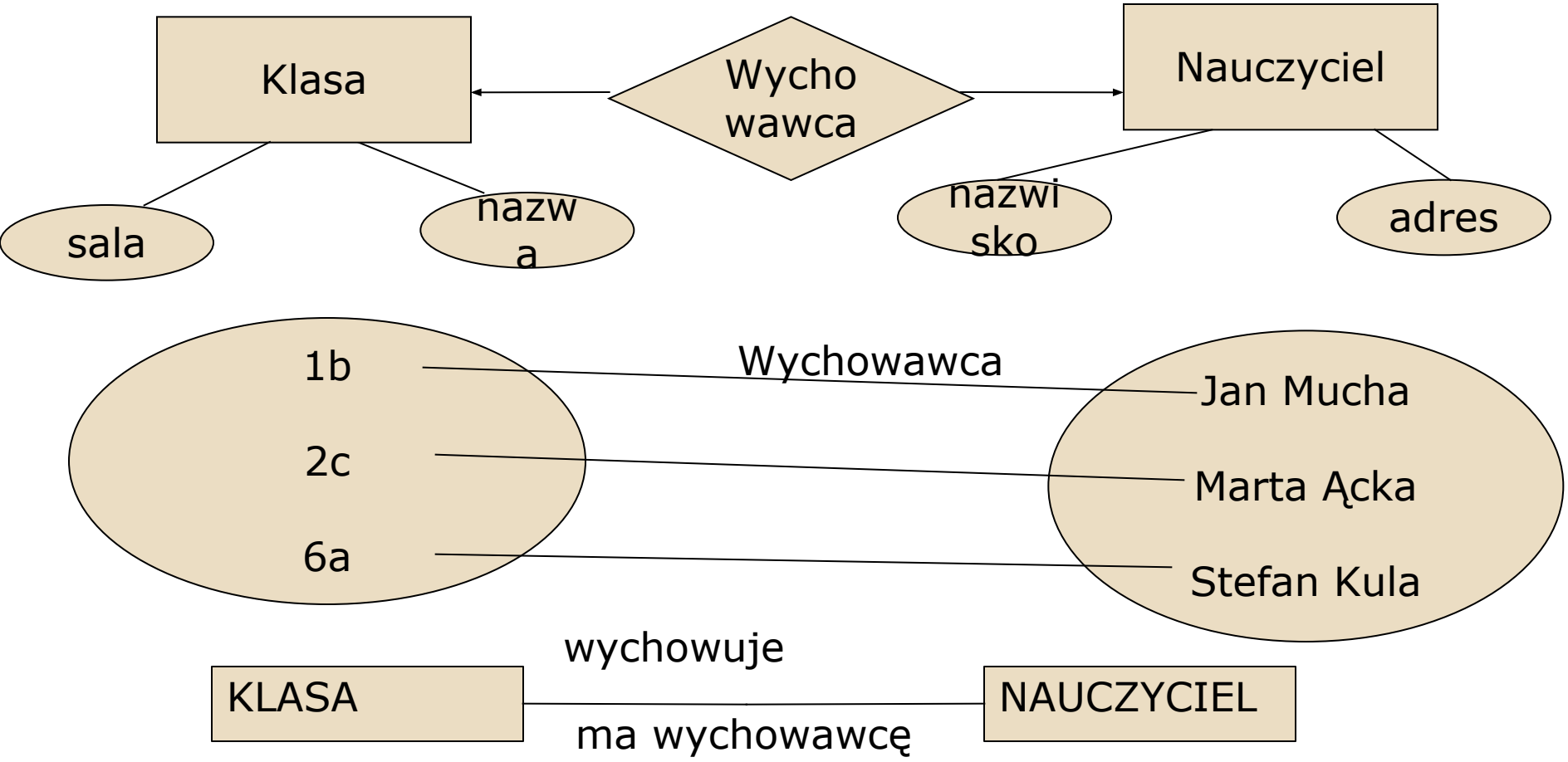



Pytania:
W ilu wykładach uczestniczy student, dla ilu studentów przewidziany jest wykład, czy wykład musi być przewidziany dla studenta, czy student musi uczestniczyć w wykładzie

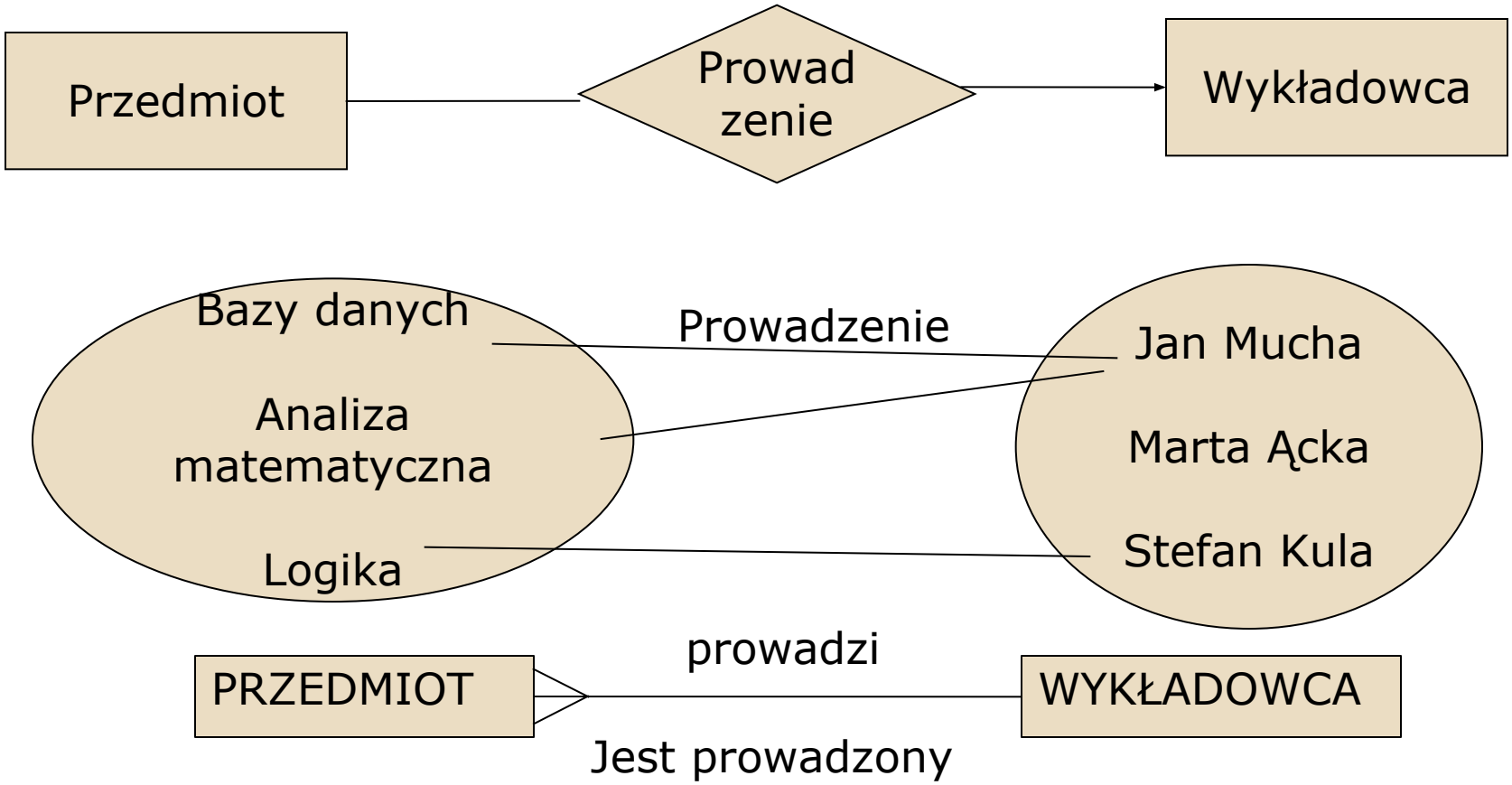
# Cechy związku

- Liczebność (unarny - rekursywny, binarny, tetrarny, n-arny)
- Istnienie (opcjonalny, obowiązkowy)
- Karynalność
  - 1:1 – jeden do jednego
  - 1:M – jeden do wielu
  - N:M – wiele do wielu

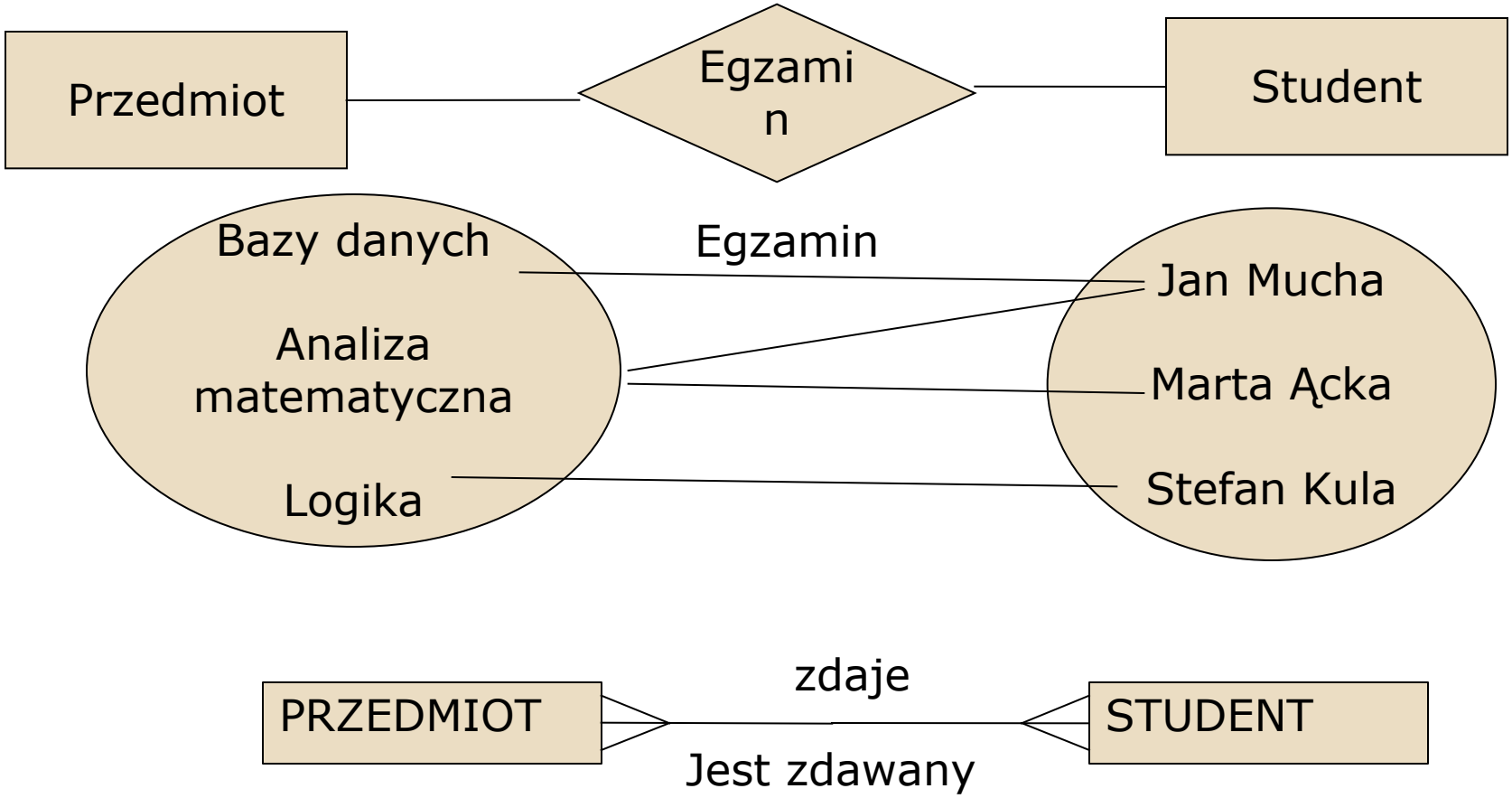
# Związek 1:1 - Przykład



# Związek 1:m - Przykład

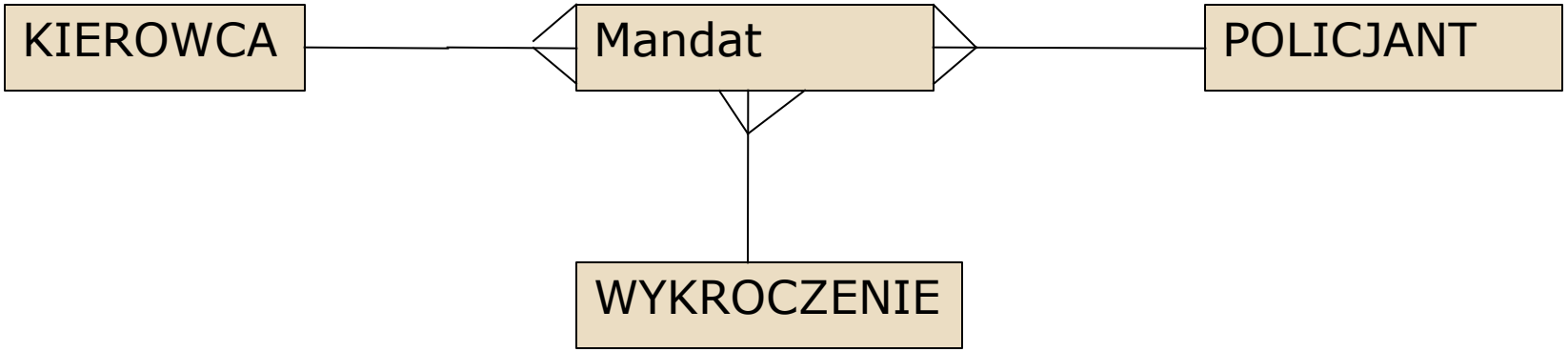


# Związek m:n - Przykład

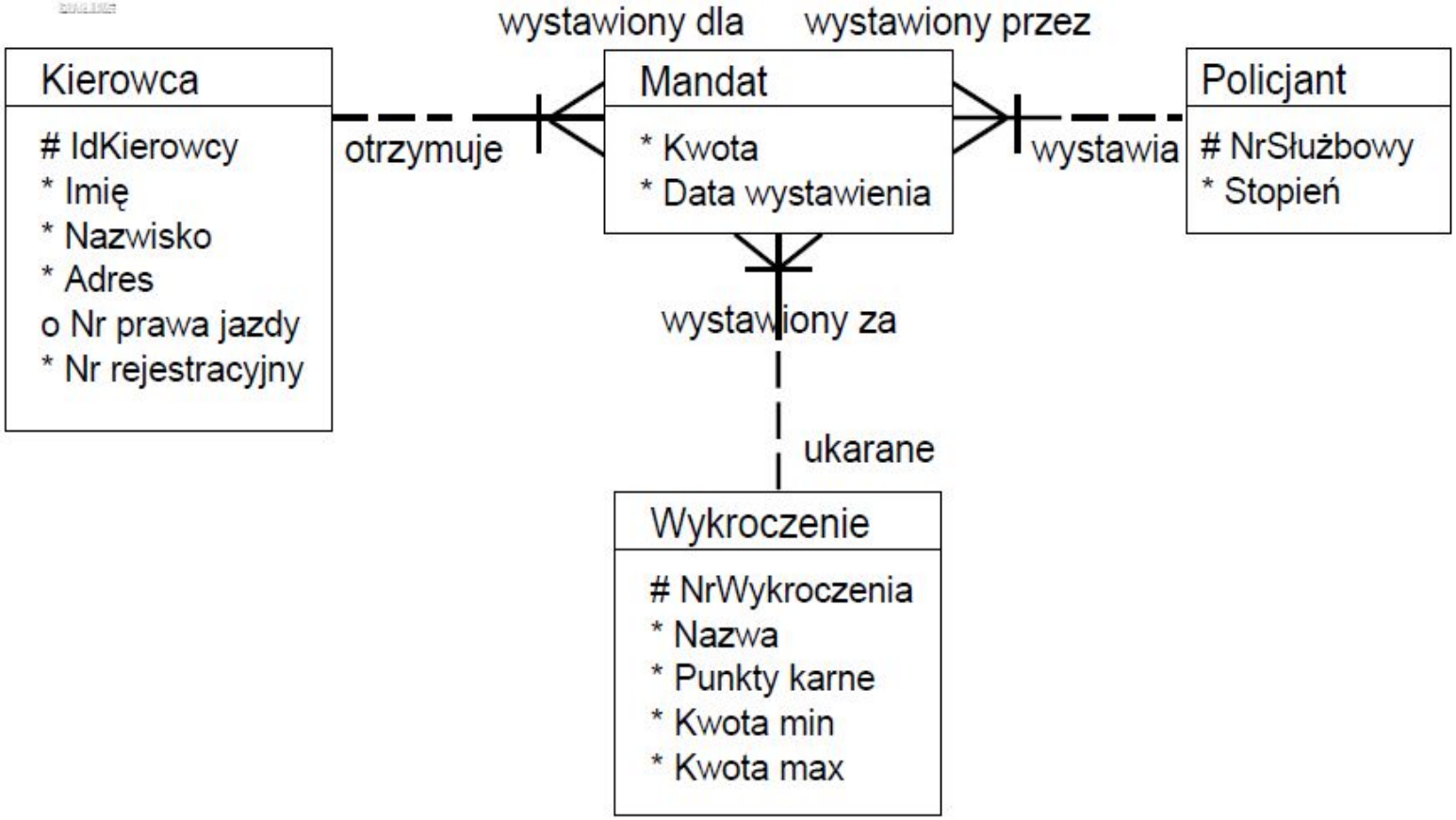


# Związki wieloargumentowe – notacja Berkera

- Gdy związek wieloargumentowy – zamienia się w encję



# Rozszerzenie – poprzedni przykład

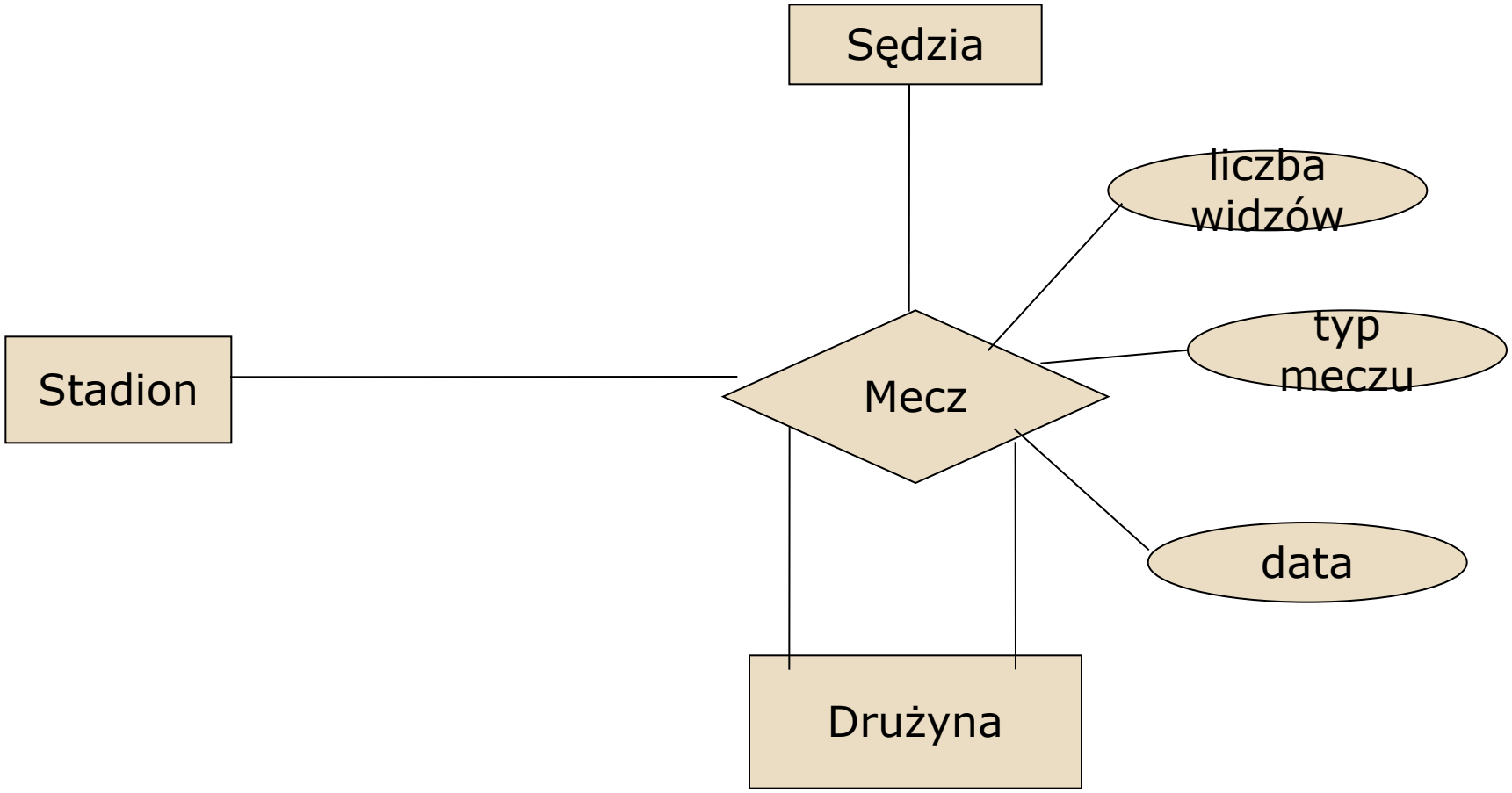



Ze strony ważniak

# Atrybuty związków

- Gdy związek posiada specyficzne cechy
- Można stworzyć encję dla związku z atrybutami odnoszącymi się do związku

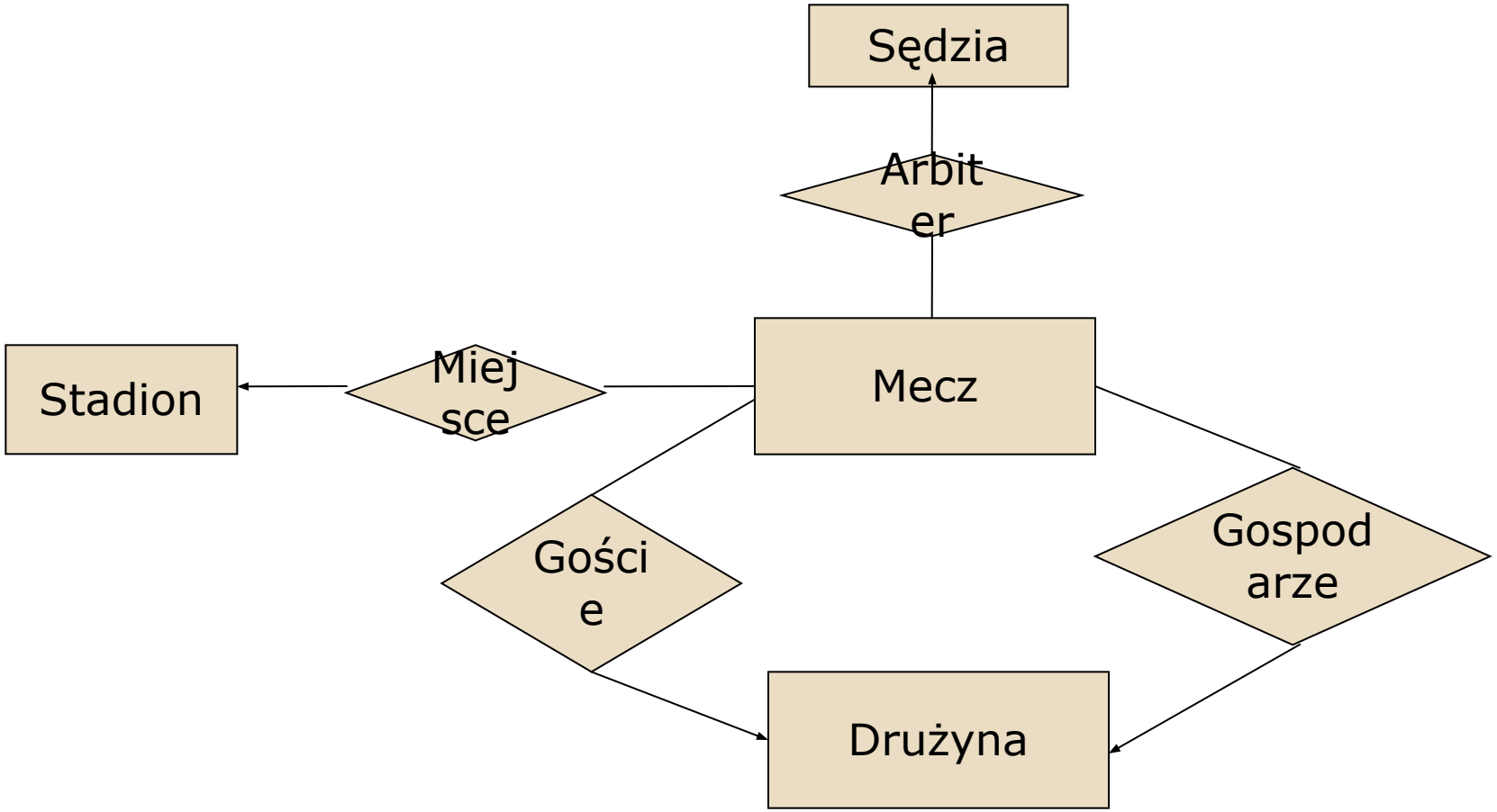
# Atrybuty związku - przykład



# Zamiana związków wielo- argumentowych na binarne

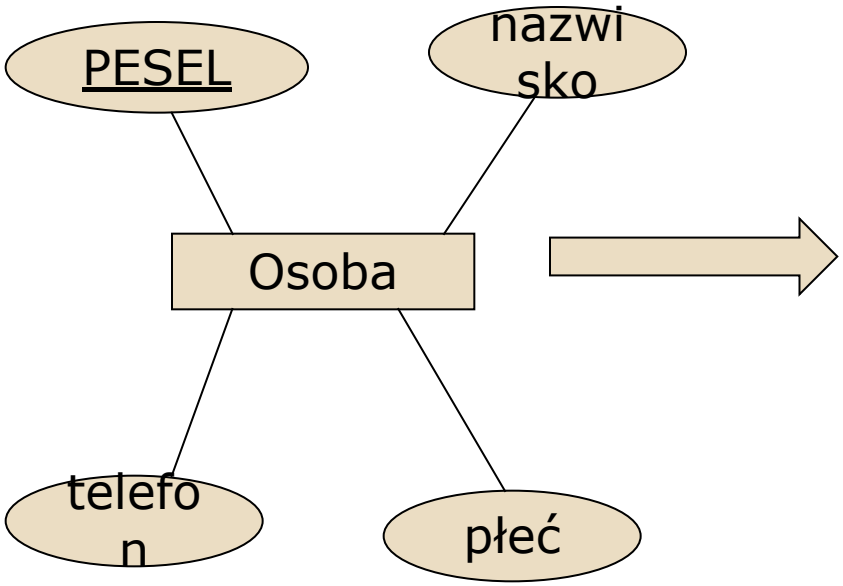
- Zamiana związku na encję
- Każda encja związku wieloargumentowego wchodzi w związek binarny jeden do wielu z nową encją

# Zamiana związków wielo- argumentowych na binarne -przykład



# Związki encji => projekty relacyjne

- Encja (nie słaba) przekształcana do relacji z tą samą nazwą oraz tym samym zbiorem atrybutów

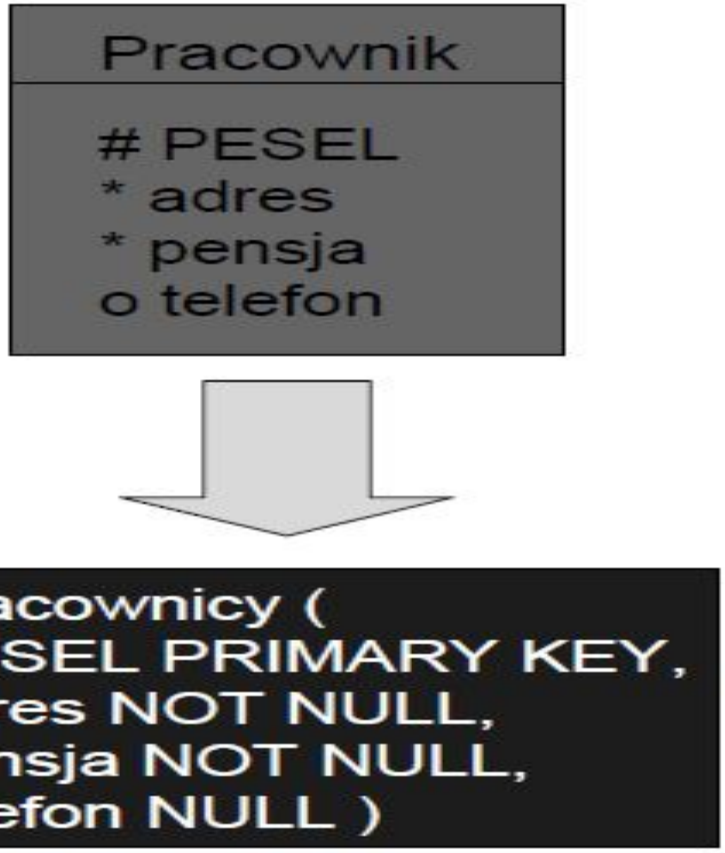



| Osoba | | | |
|---|---|---|---|
| PESEL | nazwisko | telefon | płeć |

# Reguły przekształcania

- Encja  Relacja
- Atrybut encji  Atrybut relacji
- Typ danych atrybutu encji  Typ danych atrybutu relacji
- Identyfikator  klucz podstawowy
- Obowiązkowość atrybutu  NOT NULL
- Opcjonalność  NULL
- Pozostałe ograniczenia atrybutów encji  ograniczenia integralnościowe relacji

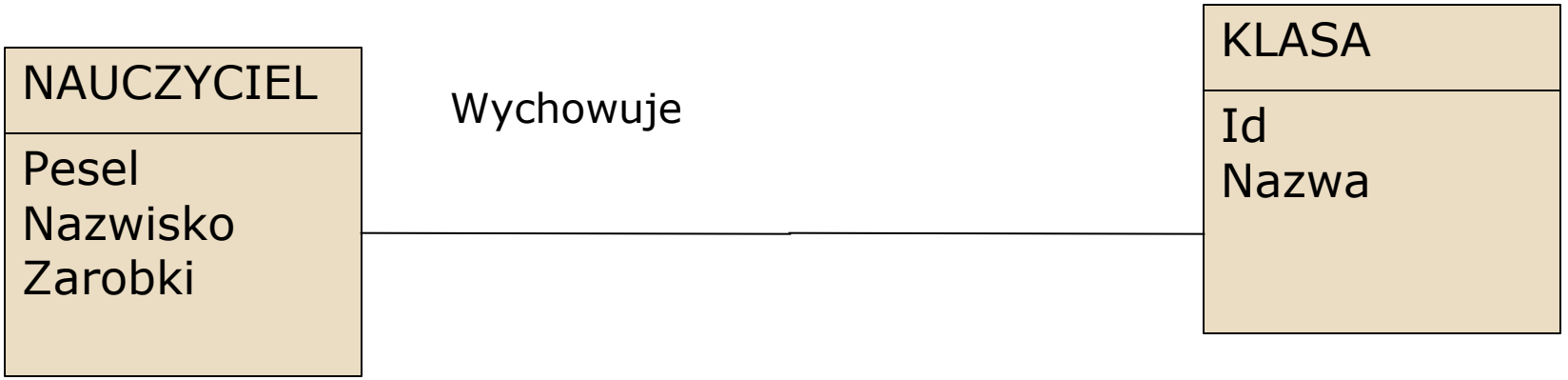
# Przykład


Pracownik
# PESEL
* adres
* pensja
o telefon
Pracownicy (
PESEL PRIMARY KEY,
adres NOT NULL,
pensja NOT NULL,
telefon NULL )

# Przekształcanie związków

- 1:1 – klucz obcy w wybranej tabeli
- 1:M – klucz obcy w tabeli po stronie wiele
- N:M – nowa tabela

# Związek binarny 1:1




- Dodany klucz obcy po stronie związku obowiązkowego

# Związek binarny 1:1


- Dodany klucz obcy po stronie mniejszej tabeli

# NORMALIZACJA

# Po co normalizować? (1)

| Student | | | | |
|---|---|---|---|---|
| *nrindeksu* | nazwisko | adres | *przedmiot* | ocena |
| 127000 | Maliniak | Świerkowa 6 | Analiza | 2 |
| 127000 | Maliniak | Świerkowa 6 | Algebra | 3 |
| 127000 | Maliniak | Świerkowa 6 | Bazy danych | 3 |
| 127000 | Maliniak | Świerkowa 6 | W-F | 5 |
| 123123 | Kowal | Akacjowa 1 | Algebra | 4 |
| 123123 | Kowal | Akacjowa 1 | Bazy danych | 5 |
| 123123 | Kowal | Akacjowa 1 | W-F | 3 |
| 666555 | Nowak | Różana 4/78 | PTO | 3 |
| 666555 | Nowak | Różana 4/78 | Sieci | 4 |

# Po co normalizować? (2)

- *nrindeksu, przedmiot* – pole unikalne
- Problemy (anomalie):
  - Redundancja
  - Przy wprowadzaniu danych
  - Przy usuwaniu danych
  - Przy aktualizacji
- Rozwiązanie – rozkład relacji na relacje **Student** oraz **Egzamin**

# Po co normalizować? (3)

Rozwiązanie:

| Student | | |
|---|---|---|
| *nrindeksu* | nazwisko | adres |
| 127000 | Maliniak | Świerkowa 6 |
| 123123 | Kowal | Akacjowa 1 |
| 666555 | Nowak | Różana 4/78 |

| Egzamin | | |
|---|---|---|
| *nrindeksu* | *przedmiot* | ocena |
| 127000 | Analiza | 2 |
| 127000 | Algebra | 3 |
| 127000 | Bazy danych | 3 |
| 127000 | W-F | 5 |
| 123123 | Algebra | 4 |
| 123123 | Bazy danych | 5 |
| 123123 | W-F | 3 |
| 666555 | PTO | 3 |
| 666555 | Sieci | 4 |

# Po co normalizować? (4)

- adres i nazwisko – tylko w jednej krotce (rozwiązanie redeundancji)
- przy wstawianiu nowego studenta – niekoniecznie przedmiot i ocena (rozwiązanie problemu wstawiania)
- usunięcie egzaminu nie usuwa studenta (rozwiązanie problemu usuwania)
- aktualizacja adresu, nazwiska – tylko raz (rozwiązanie problemu aktualizacji)

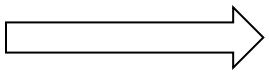
# Normalizacja

- Dekompozycja relacji, aż do osiągnięcia pożądanych cech schematu – różnych dla każdej postaci normalnej
- Własności normalizacji:
  - zachowania atrybutów
  - zachowania informacji
  - zachowania zależności

# Pierwsza postać normalna – 1NF (1)

- Definicja – relacja jest w pierwszej postaci normalnej wtw gdy każdy atrybut jest zależny funkcyjnie od klucza relacji

| Oceny | |
|---|---|
| *Student* | Ocena |
| Nowak | 2;4;3,5 |
| Kowal | 2;5 |



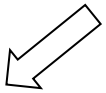
| Oceny w 1NF | |
|---|---|
| *Student* | *Ocena* |
| Nowak | 2 |
| Nowak | 4 |
| Nowak | 3,5 |
| Kowal | 2 |
| Kowal | 5 |

# Druga postać normalna - 2NF

- Definicja – relacja jest w drugiej postaci normalnej wtw gdy jest w pierwszej postaci normalnej oraz każdy atrybut niekluczowy jest w pełni funkcyjnie zależny od klucza głównego (zależy o całego klucza, a nie od jego części)

# Przekształcenie do 2NF - przykład

| Zaliczenie w 1NF | | | |
|---|---|---|---|
| *nrindeksu* | *przedmiot* | Nazwisko | ocena |
| 12345 | Analiza | Kowal | 3,5 |
| 12345 | Algebra | Kowal | 3,5 |
| 54321 | Bazy Danych | Nowak | 5 |
| 54321 | Algebra | Nowak | 3,5 |



| Student w 2NF | |
|---|---|
| *nrindeksu* | Nazwisko |
| 12345 | Kowal |
| 54321 | Nowak |

| Zaliczenie w 2NF | | |
|---|---|---|
| *nrindeksu* | *przedmiot* | ocena |
| 12345 | Analiza | 3,5 |
| 12345 | Algebra | 3,5 |
| 54321 | Bazy Danych | 5 |
| 54321 | Algebra | 3,5 |

# Trzecia postać normalna – 3NF

- Definicja – relacja jest w trzeciej postaci normalnej wtw gdy jest w drugiej postaci normalnej oraz gdy każdy niekluczowy atrybut relacji jest bezpośrednio zależny od klucza relacji

# Przekształcenie do 3NF - przykład

| Przedmioty w 2 NF | | |
|---|---|---|
| *przedmiot* | pesel | Nazwisko |
| Analiza | 78071103350 | Kowal |
| Algebra | 78071103350 | Kowal |
| Bazy Danych | 68121103312 | Nowak |

| Przedmioty w 3NF | |
|---|---|
| *przedmiot* | pesel |
| Analiza | 78071103350 |
| Algebra | 78071103350 |
| Bazy Danych | 68121103312 |

| Wykładowcy w 3NF | |
|---|---|
| *pesel* | Nazwisko |
| 78071103350 | Kowal |
| 78071103350 | Kowal |
| 68121103312 | Nowak |

# Postać normalna Boyce'a - Codda

- Definicja – relacja jest w postaci normalnej Boyce'a – Codda wtw gdy dla każdej zależności nietrywialnej $A_1,\ldots,A_n \rightarrow B$ zbiór $\{A_1,\ldots,A_n\}$ jest nadkluczem tej relacji

# BCNF - dekompozycja

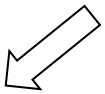
- Odnalezienie nietrywialnej zależności funkcyjnej:

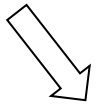
  $A_1A_2...A_n \rightarrow B_1B_2...B_n$, która narusza BCNF – tzn. $A_1A_2...A_n$ nie jest nadkluczem
- Dodanie do prawej strony wszystkich atrybutów zależnych funkcyjnie od $A_1A_2...A_n$ – w ten sposób powstaje nowa relacja
- Druga relacja będzie się składała z atrybutów $A_1A_2...A_n$ oraz z pozostałych (poza $B_1B_2...B_n$) atrybutów relacji

# BCNF – dekompozycja - przykład

| Zaliczenie | | | |
|---|---|---|---|
| *nrindeksu* | *przedmiot* | Nazwisko | ocena |

Nrindeksu, przedmiot  ocena nrindeksu  nazwisko





| Student w BCNF | |
|---|---|
| *nrindeksu* | Nazwisko |

| Zaliczenie BCNF | | |
|---|---|---|
| *nrindeksu* | *przedmiot* | ocena |

# SQL – STRUCTURED QUERY LANGUAGE

# Zadania SQL

- definiowanie danych
- definiowanie perspektyw
- przetwarzanie danych (interaktywne i programowe)
- definiowanie reguł integralności danych
- autoryzacja
- określanie początku transakcji, potwierdzenie i wycofywanie transakcji

# Grupy instrukcji

- Język definicji danych (DDL): CREATE, ALTER, DROP
- Język manipulowania danymi (DML): SELECT, INSERT, UPDATE i DELETE
- Instrukcje Sterowania Danymi: GRANT i REVOKE

# Data Definition Language (DDL)

- Tworzenie tabel, baz danych, itd. CREATE
- Modyfikacja schematu bazy danych – ALTER
- Usuwanie tabel, baz danych itd. - DROP

# Polecenie CREATE

- Tworzenie bazy danych

  CREATE database nazwa_bazy;

  Przykład: CREATE database restauracja;

- Tworzenie tabeli

  CREATE table nazwa(pole1 typ_danych1 ograniczenia1, pole2 typ_danych2 ograniczenia2, …, poleN typ_danychN ograniczeniaN, ograniczeniaOgólne);

# Różne typy danych

- Przykład

CREATE table Osoby(id_osoby Smallint auto_increment, Nazwisko Varchar(23), data_ur Date, czas_maratonu Time, rok_rozp YEAR(4), zarobki Decimal(7,2))

# Ograniczenia(1)

- NOT NULL – wymusza wpisanie wartości dla danego pola
- UNIQUE – wartości w danym polu nie mogą się powtarzać
- CHECK (warunek) – nakłada warunek na relację
- DEFAULT wartość – domyślnie wartość
- Przykład

```
CREATE table Pracownicy(id_prac Smallint(3) zerofill auto_increment,
    Nazwisko Varchar(25) Unique, zarobki Decimal(7,2) Default 1500,
    Check(zarobki>0))
```

# Ograniczenia(2)

- PRIMARY KEY – definicja klucza głównego; może być definiowany przy polu które jest kluczem, bądź na koniec relacji – gdy więcej pól niż jedno
- FOREIGN KEY (nazwa_pola) REFERENCES nazwa_tabeli(nazwa_pola1) – klucz obcy

# Ograniczenia(3)

- Przykład definicji kluczy

  1) CREATE table Pracownicy(id_prac Smallint(3) auto_increment PRIMARY KEY, Nazwisko Varchar(25) Unique, zarobki Decimal(7,2));

  2) CREATE table Projekty(nr_projektu Smallint Primary Key, nazwa char(20),

  kierownik Smallint, Foreign key(kierownik) References Pracownicy(id_prac));

# Ograniczenia(4)

- Wymuszanie więzi integralności
  - usuwanie

a) FOREIGN KEY(pole1) REFERENCES tabela(pole2) ON DELETE SET NULL

b) FOREIGN KEY(pole1) REFERENCES tabela(pole2) ON DELETE CASCADE

c) FOREIGN KEY(pole1) REFERENCES tabela(pole2) ON DELETE SET DEFAULT

d) FOREIGN KEY(pole1) REFERENCES tabela(pole2) ON DELETE RESTRICT

# Modyfikacja schematu relacji

- ALTER TABLE – dodawanie, usuwanie atrybutów oraz ograniczeń integralnościowych, modyfikacja definicji atrybutu

  Przykład:

  ALTER TABLE Pracownicy ADD Primary Key(Id_prac);

# ALTER TABLE - dodawanie

- Dodawanie kolumny

  ALTER TABLE nazwa_tabeli ADD COLUMN pole typ_pola;

  Przykład:

  ALTER TABLE Pracownicy ADD COLUMN stanowisko VARCHAR(20) AFTER NAZWISKO;

- Dodawanie ograniczenia

  ALTER TABLE nazwa ADD CONSTRAINT nazwa i rodzaj ograniczenia (PRIMARY KEY, FOREIGN KEY, CHECK, itd.)

# ALTER TABLE - usuwanie

- Usuwanie kolumny

  ALTER TABLE nazwa DROP COLUMN pole

- Usuwanie ograniczenia

  ALTER TABLE nazwa DROP CONSTRAINT
  nazwa_ograniczenia;

  Przykład

  ALTER TABLE Pracownicy DROP CONSTRAINT Klucz;

# ALTER TABLE - modyfikowanie

- Tylko do atrybutów

  ALTER TABLE nazwa MODIFY pole typ ograniczenia;

  Przykład

  ALTER TABLE Pracownicy MODIFY Nazwisko Char(30) not null;

- Zmiana nazwy i typu atrybutu

  ALTER TABLE nazwa CHANGE starepole nowepole typ ograniczenia

# Modyfikacje baz danych

- Trzy typy instrukcji
  - Wstawianie – INSERT INTO
  - Usuwanie – DELETE FROM
  - Aktualizacje - UPDATE

# Wstawianie (1)

- INSERT INTO tabela VALUES (wart1, wart2, …, wartn); - polecenie wstawia do tabeli wartości war1, …, warn
- Ilość wartości = ilość atrybutów relacji
- Kolejność wartości odpowiada definicji tabeli

# Wstawianie – przykład 1

Tabela studenci(nrIndeksu, nazwisko, imię, adres, rok studiów)

INSERT INTO Studenci values(123456, 'Kowal', 'Stefan', 'Akacjowa 4 Łódź', 5);

# Wstawianie (2)

- INSERT INTO tabela(pole1, pole2, …,polek) VALUES (wart1, wart2, …, wartk);
- Do pole1 wstawiana wart1 itd.
- Liczba pól nie musi być równa liczbie atrybutów relacji

# Wstawianie

- Wstawiane wiersze jako wynik zapytania

- Przykład:

```
INSERT INTO Studenci (Imie, Nazwisko, rok) SELECT
   imię, nazwisko,1 from Kandydaci;
```

# Usuwanie

- DELETE FROM tabela [WHERE warunek];

Bez warunku – usuwa wszystkie krotki z tabeli

- Przykłady:

  DELETE FROM Studenci;

  DELETE FROM Studenci WHERE rok=5;

# Aktualizacja

- UPDATE tabela SET nowe wartości [WHERE warunek];
- Nowe wartości w postaci atrybut=wartość
- Zmodyfikowane zostaną krotki spełniające warunek
- Przykład:

```
UPDATE Studenci SET rok=rok+1 WHERE rok<5;
UPDATE Pracownicy SET placa=placa+300 where
    stanowisko<>'Dyrektor';
```

# Wyszukiwanie

- Wybieranie interesujących informacji z jednej lub wielu relacji
- Najprostsza postać:

  SELECT * FROM tabela – zwraca wszystkie krotki z tabeli
- Po SELECT są wymieniane wybrane atrybuty (* - oznacza wszystkie)
- Po FROM wymieniane są relacje, których dotyczy zapytanie

# Wyszukiwanie – selekcja

- SELECT * FROM Tabela WHERE warunek;
- Zwracane wszystkie krotki spełniające warunek
- Przykład:

PRACOWNICY

| Id | Imię | Nazwisko | PESEL | Pensja |
|---|---|---|---|---|
| 1 | Jan | Topa | 68010333546 | 1400 |
| 2 | Monika | Stachura | 78022212121 | 3400 |
| 3 | Michał | Posek | 87010234567 | 5400 |
| 4 | Jan | Mara | 84081222000 | 2000 |

# Wyszukiwanie - selekcja

SELECT * FROM Pracownicy WHERE pensja>3000;

| Id | Imię | Nazwisko | PESEL | Pensja |
|---|---|---|---|---|
| 2 | Monika | Stachura | 78022212121 | 3400 |
| 3 | Michał | Posek | 87010234567 | 5400 |

SELECT * FROM Pracownicy WHERE Nazwisko LIKE '%ra%' AND Pensja BETWEEN 1000 AND 2500;

| Id | Imię | Nazwisko | PESEL | Pensja |
|---|---|---|---|---|
| 4 | Jan | Mara | 84081222000 | 2000 |

# Wyszukiwanie - selekcja

- Porównywanie wartości za pomocą operatorów =,<>,<,>,>=,<=, !=
- Operacje arytmetyczne – podobnie jak na liczbach
- Operatory logiczne AND, OR i NOT
- Operatory LIKE, BETWEEN AND, IN

# Wyszukiwanie - selekcja

- Przykład

SELECT * FROM Pracownicy WHERE

(imie NOT LIKE '%M%' OR imie IN ('Jan','Monika')) AND id>=3;

| Id | Imię | Nazwisko | PESEL | Pensja |
|---|---|---|---|---|
| 4 | Jan | Mara | 84081222000 | 2000 |
| 5 | Anna | Rożek | 76012900128 | 2500 |

# Wyszukiwanie – projekcja

- SELECT pole1, pole2,…,polen FROM Tabela;
- Wyświetla wybrane atrybuty dla poszczególnych krotek
- Przykład :

SELECT imie, nazwisko

FROM Pracownicy;

| Imię | Nazwisko |
|---|---|
| Jan | Topa |
| Monika | Stachura |
| Michał | Posek |
| Jan | Mara |
| Anna | Rożek |

# Wyszukiwanie – aliasy, wyrażenia

- CONCAT(wyr1,wyr2,…,wyrN) łączy pola w jedno
- Wyr1 AS Wyr2 – jako nagłówek atrybutu Wyr2

Przykład:

```
SELECT CONCAT(imie, ' ',nazwisko) FROM Pracownicy
   WHERE pensja>5000;
```

| CONCAT(imie, ' ',nazwisko) |
| --- |
| Michał Posek |

# Wyszukiwanie – aliasy, wyraŻenia

- Przykład

SELECT CONCAT(imie, ‘ ‘,nazwisko) AS Osoba, pensja/20 AS Dniówka FROM Pracownicy WHERE Id!=3 AND pensja>3000;

| Osoba | Dniówka |
|---|---|
| Monika Stachura | 170 |

# Wyszukiwanie – porządek wyświetlania, usuwanie duplikatów

- DISTINCT – różne wartości atrybutów

- ORDER BY pole1 [ASC/DESC], … – ustawia kolejność wyświetlania wyników rosnąco lub malejąco według kolejno wymienionych pól – domyślne ustawienie na rosnąco

# Wyświetlanie - przykład

SELECT DISTINCT Imie FROM Pracownicy;

| Imię |
|---|
| Jan |
| Monika |
| Michał |
| Anna |

SELECT * FROM PRACOWNICY ORDER BY imie, placa DESC;

| Id | Imię | Nazwisko | PESEL | Pensja |
|---|---|---|---|---|
| 5 | Anna | Rożek | 76012900128 | 2500 |
| 4 | Jan | Mara | 84081222000 | 2000 |
| 1 | Jan | Topa | 68010333546 | 1400 |
| 3 | Michał | Posek | 87010234567 | 5400 |
| 2 | Monika | Stachura | 78022212121 | 3400 |

# Funkcje agregujące

- Każda funkcja działa na zbiorach powstałych poprzez grupowanie względem jakiegoś wyrażenia
- Dla każdego zbioru zwraca jedną wartość

  Zadanie „Znaleźć średnią ocen dla każdego studenta”

# Funkcje agregujące, grupowanie

- AVG – zwraca średnią
- COUNT – zlicza liczbę wystąpień
- MIN – zwraca wartość minimalną
- MAX – zwraca wartość maksymalną
- SUM – zwraca sumę
- GROUP BY pole – determinuje według którego pola następuje grupowanie
- HAVING warunek – ogranicza grupy to tych których wszystkie krotki spełniają nałożony warunek

# Funkcje agregujące przykład

- SELECT NrIndeksu, AVG(Ocena) AS Średnia FROM Egzamin GROUP BY NrIndeksu;
- SELECT NrIndeksu, COUNT(Przedmioty) AS 'Ilosc zdawanych' FROM Egzamin GROUP BY NrIndeksu ORDER BY NrIndeksu DESC;

| NrIndeksu | Ilosc zdawanych |
|---|---|
| 66666 | 2 |
| 54321 | 3 |
| 12345 | 4 |

# Łączenie relacji – połączenia wewnętrzne

- Potrzebne informacje z więcej niż jednej tabeli
- Rodzaje połączeń
  - CROSS JOIN – iloczyn kartezjański
  - JOIN ON operator równości – połączenie równościowe
  - NATURAL JOIN, JOIN USING – połączenie naturalne
  - JOIN ON dowolny operator – połączenie nierównościowe

# Połączenie równościowe

- Bierze pod uwagę krotki, które spełniają wyrażenie po ON
- Przykład:

```
SELECT Tytuł, Ilość FROM Książki JOIN Zamówienia ON Książki.id =
    Zamówienia.IdKsiazki
```

| Tytuł | Ilość |
|---|---|
| Lalka | 2 |
| Szwejk | 4 |

# Połączenie naturalne

- Bierze pod uwagę krotki mające tę samą nazwę w obu relacjach
- Przykład

SELECT Tytuł, Ilość FROM Książki NATURAL JOIN Zamówienia;

| Tytuł | Ilość |
|---|---|
| Lalka | 2 |
| Potop | 4 |

# Połączenia zewnętrzne

- Zwracane wszystkie krotki z wybranej relacji
- LEFT – zwraca wszystkie wystąpienia relacji po lewej stronie połączenia
- RIGHT - zwraca wszystkie wystąpienia relacji po prawej stronie połączenia
- FULL – zwraca wszystkie wystąpienia obu relacji
- SELECT atrybuty FROM tabela1 LEFT|RIGHT|FULL OUTER JOIN tabela2 on warunek| using (atrybut);

# Połączenia zewnętrzne - przykład

Książki

| Id | Tytuł | Cena | Wydawca |
|---|---|---|---|
| 1 | Lalka | 47 | PWN |
| 2 | Potop | 34 | PTE |
| 3 | Szwejk | 70 | PTE |

Zamówienia

| Id | IdKsiązki | Ilość | Data |
|---|---|---|---|
| 1 | 1 | 2 | 08-03-01 |
| 2 | 3 | 4 | 08-02-22 |

# Przykład cd

SELECT Tytuł, Cena, Ilość FROM Książki LEFT OUTER JOIN Zamówienia USING (Id);

| Id | Tytuł | Cena | IdKsiązki | Ilość | Data |
|---|---|---|---|---|---|
| 1 | Lalka | 47 | 1 | 2 | 08-03-01 |
| 2 | Potop | 34 | 3 | 4 | 08-02-22 |
| 3 | Szwejk | 70 | NULL | NULL | NULL |

| Tytuł | Cena | Ilość |
|---|---|---|
| Lalka | 47 | 2 |
| Potop | 34 | 4 |
| Szwejk | 70 | NULL |

SELECT Tytuł, Cena, Ilość FROM Książki LEFT OUTER JOIN Zamówienia ON Książki.Id = Zamówienia.IdKsiążki;

| Tytuł | Cena | Ilość |
|---|---|---|
| Lalka | 47 | 2 |
| Potop | 34 | NULL |
| Szwejk | 70 | 4 |

# Połączenia zwrotne

- Łączenie tabeli samej ze sobą
- Przykład:

```
SELECT p.imie || ' ' || p.nazwisko as pracownik,
   s.nazwisko as szef FROM Pracownicy p JOIN
   Pracownicy s on p.id = s.id_szefa;
```

Zapytanie dla każdego pracownika zwróci nazwisko szefa

# Połączenia zwrotne - przykład

Pracownicy p

| id | imie | nazwisko | id_szefa |
|---|---|---|---|
| 1 | Jacek | Barcik | 2 |
| 2 | Anna | Baran | NULL |
| 3 | Tomasz | Kwiecień | 2 |

Pracownicy s

| id | imie | nazwisko | id_szefa |
|---|---|---|---|
| 1 | Jacek | Barcik | 2 |
| 2 | Anna | Baran | NULL |
| 3 | Tomasz | Kwiecień | 2 |

| id | imie | nazwisko | id_szefa | imie | nazwisko | id_szefa |
|---|---|---|---|---|---|---|
| 1 | Jacek | Barcik | 2 | Anna | Baran | NULL |
| 2 | Tomasz | Kwiecień | 2 | Anna | Baran | NULL |

# Łączenie wielu relacji

- Połączenie relacji z wcześniej połączonymi relacjami
- Przykład:

```
SELECT k.nazwisko as Klient, t.nazwa as Produkt, t.cena *
z.ilosc as Suma FROM (Klienci k JOIN Zamowienie z ON
k.id_klienta = z.id_klienta) JOIN Towary t ON z.id_towaru =
t.id_towaru;
```

# Wiele relacji - przykład

Zamówienia z

| id | Id_towaru | Id_klienta | ilosc |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 2 | 1 | 3 | 4 |
| 3 | 2 | 3 | 1 |

Towary t

| Id_towaru | nazwa | cena |
|---|---|---|
| 1 | Pączek | 1.3 |
| 2 | Chleb | 1.8 |
| 3 | Masło | 4.5 |

Klienci k

| Id_klienta | nazwisko |
|---|---|
| 1 | Barcik |
| 2 | Baran |
| 3 | Kwiecień |

| Klient | Produkt | Suma |
|---|---|---|
| Barcik | Pączek | 2.6 |
| Kwiecień | Pączek | 5.2 |
| Kwiecień | Chleb | 1.8 |

# Podzapytania

- Można stosować dla klauzuli:
  - WHERE
  - HAVING
  - FROM
- Taka sama postać jak zwykłe zapytanie – ujęte w nawiasy
- Podzapytanie jako prawy argument predykatów
  - =, <, <=, >, >=, <>, IN, NOT IN

# Podzapytania

- Wierszowe

  SELECT * FROM pracownik WHERE zarobki = (SELECT MAX(zarobki ) FROM pracownik);

- Tablicowe

  SELECT * FROM pracownik WHERE id_pracownika NOT IN (SELECT prowadzacy FROM przedmioty);

  SELECT * FROM student WHERE nazwisko LIKE '%a%' AND nrindeksu IN (SELECT student FROM oceny WHERE ocena=5);

  INSERT INTO student(imie, nazwisko, adres, rok, telefon) SELECT imie, nazwisko, adres, 1, 997 from kandydaci;

# Podzapytania – kwantyfikatory (1)

- ALL – dla wszystkich elementów podzapytania warunek musi być spełniony

```
SELECT imie, nazwisko FROM pracownik WHERE zarobki > ALL (SELECT
    zarobki FROM pracownik WHERE stanowisko = 'adiunkt');
```

- ANY(SOME) – co najmniej dla jednego elementu podzapytania warunek musi być spełniony

```
SELECT imie, nazwisko FROM pracownik WHERE zarobki > ANY (SELECT
    zarobki FROM pracownik WHERE stanowisko = 'adiunkt');
```

# Podzapytania – kwantyfikatory (2)

- EXISTS – kwantyfikator egzystencjalny „istnieje”

```
SELECT nazwisko FROM pracownik WHERE EXISTS (SELECT ‘x’ FROM przedmioty
    WHERE przedmioty.prowadzacy = pracownik.id_pracownika);
```

- NOT EXISTS – kwantyfikator uniwersalny z negacją „dla każdego nieprawda że”

```
SELECT nazwisko FROM pracownik WHERE NOT EXISTS (SELECT ‘x’ FROM
    przedmioty WHERE przedmioty.prowadzacy = pracownik.id_pracownika);
```

# Podzapytania – tworzenie tabel

- Po FROM

  SELECT a.stanowisko, 100*a.liczbaprac/b.liczbaprac as ‘procPracowników’, 100*a.zarob/b.zarob as ‘procZarobkow’ FROM (SELECT stanowisko, COUNT(*) AS liczbaprac, SUM(zarobki) as zarob FROM pracownik GROUP BY stanowisko) a, (SELECT COUNT(*) AS liczbaprac, SUM(zarobki) AS zarob FROM pracownik) b;

- Tworzenie tabeli (po AS)

  CREATE TABLE nowa (Imie varchar(30), Nazwisko varchar(30)) AS SELECT imie, nazwisko FROM pracownik WHERE zarobki >4000;

# Perspektywy (1)

- Nazwana tabela
- Nie może istnieć samodzielnie – dane pobiera z tabel bazowych (stworzonych przez CREATE TABLE) lub innych perspektyw
- W MySQL może posłużyć do zapamiętywania wykonywanych zapytań
- Gdy dane są aktualizowane w tabeli bazowej – odzwierciedlenie w perspektywie
- Gdy struktura tabeli bazowych się zmienia – brak odzwierciedlenia w perspektywie

# Perspektywy (2)

- Określają widok na bazę danych dla pewnych grup użytkowników
- Możliwe usuwanie, dodawanie, aktualizacja danych w perspektywie – dane w tabeli bazowej również zmieniana

# Perspektywy – SQL (1)

- Tworzenie – składnia:

  CREATE [OR REPLACE] VIEW nazwa AS zapytanie;

- Przykład:

  CREATE OR REPLACE VIEW Pierwszy AS SELECT nazwisko FROM Student
  WHERE rok=1;

- Usuwanie – składnia:

  DROP VIEW nazwa [RESTRICT/ CASCADE];

- Opcja sprawdzania (WITH CHECK OPTION) – sprawdza czy warunek podany w perspektywie nie zostaje zmieniony przez modyfikację bądź dodanie nowej krotki

# Perspektywy – SQL (2)

- Przykład

CREATE VIEW bogacze AS SELECT * FROM Pracownik WHERE zarobki > 4000 WITH CHECK OPTION;

INSERT INTO bogacze(nazwisko, imie , zarobki) VALUES (‘Biedak’, ‘Jan’, 2000); - takie zapytanie zwróci błąd

INSERT INTO bogacze(nazwisko, imie , zarobki) VALUES (‘Bogaty’, ‘Stefan’, 5000); - krotka zostanie dodana

# INDEKSY

# Indeks - wprowadzenie

- Problem – jak efektywnie wyszukiwać rekordów z zadanego zakresu wartości wybranego pola?
- Stworzenie pliku zdefiniowanego na atrybucie po którym dokonywane jest wyszukanie
- Zawartość pliku – rekordy odpowiadające wartościom pierwszych rekordów w poszczególnych blokach pliku danych

  <pierwszy klucz w bloku, wskaźnik do bloku>

# Indeks

- Stworzony plik nazywamy **indeksem**
- Cechy indeksu:
  - Przyśpiesza dostęp do danych
  - Zakładany na atrybutach relacji (atrybuty indeksowe)
  - Rekord indeksu zawiera dwa pola:
    - Klucz (odnosi się do atrybutu indeksowego)
    - Wskaźnik do bloku mającego ten sam klucz

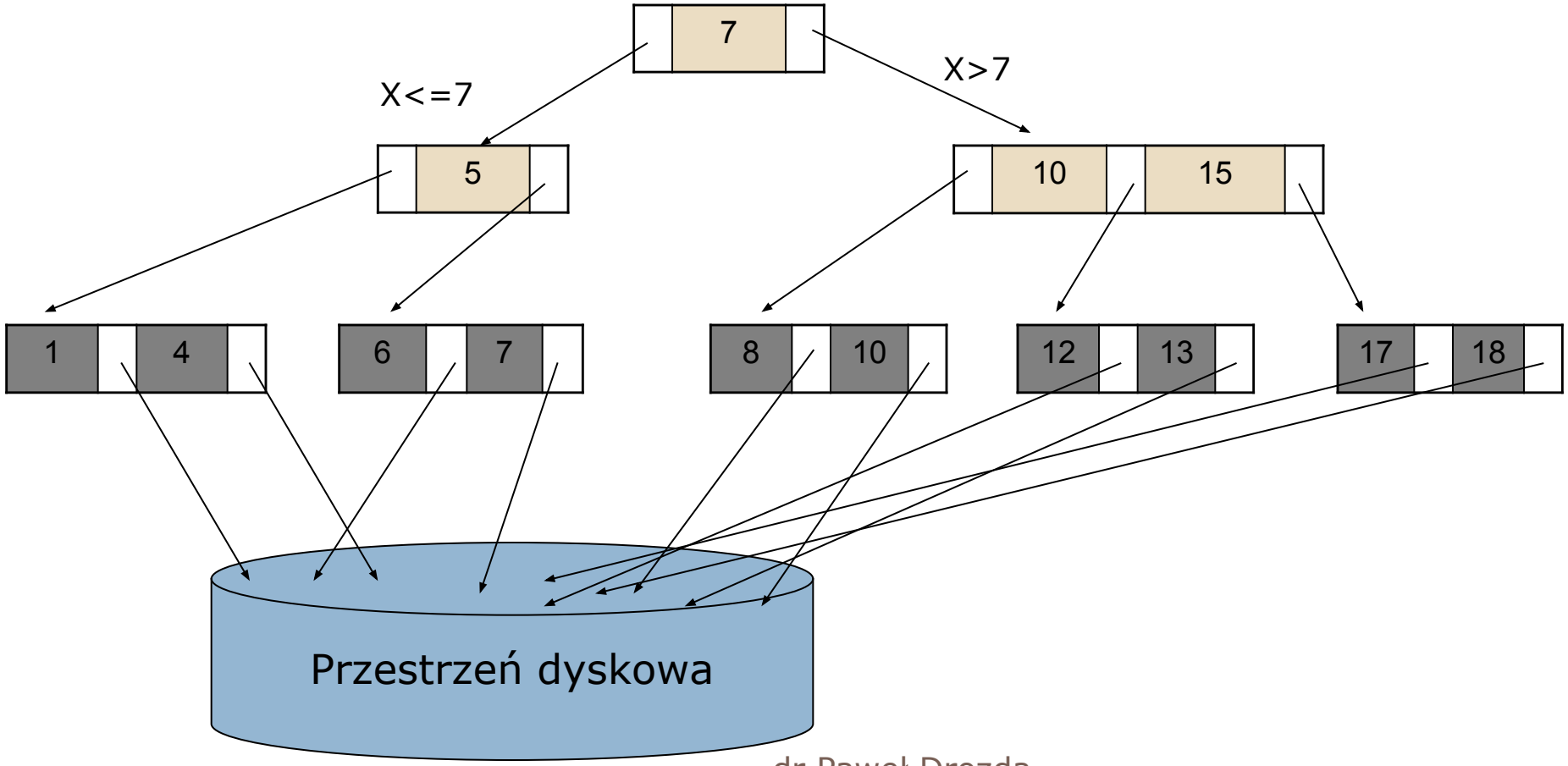
# Indeks - SQL

- CREATE INDEX nazwaindeksu ON nazwatabeli(pole1,pole2,…,polen);
- tworzenie indeksu z pól od 1 do n dla tabeli nazwatabeli

# Indeks – $B^+$ drzewo

- Zrównoważona struktura drzewiasta – każdy liść na tym samym poziomie
  - Węzły wspomagają wyszukiwanie
  - Liście wskazują na rekordy danych
  - Liście stanowią listę dwukierunkową
- Wstawianie i usuwanie rekordów pozostawiają indeks zrównoważony
- Wyszukanie rekordu – przejście od korzenia do liścia (długość ścieżki od korzenia do liścia – wysokość drzewa indeksu)

# Struktura indeksu $B^+$ drzewa



# Rząd indeksu – koszt wyszukania

- bez indeksu:

  Średnio liczba bloków danych/2=1500

- z $B^+$ drzewem: wysokość drzewa + 1=4

  odczytanie korzenia,

  odczytanie węzła,

  odczytanie liścia,

  odczytanie bloku rekordów zawierającego szukany rekord

# TRANSAKCJE

# Przykład wprowadzający

- Rezerwacja biletów lotniczych na lot X w firmie A przez pasażera Y za kwotę Z
  - Awaria

    po dokonaniu zapłaty (przed wystawieniem biletu) – częściowo wykonane operacje
  - Dwie osoby w tym samym czasie rezerwują ostatni bilet na dany lot
- Rozwiązanie - **transakcja**

# Transakcje

- *Sekwencja logicznie powiązanych operacji na bazie danych. Przeprowadza bazę z jednego stanu spójnego w inny stan spójny*
- Dozwolone operacje:
  - Odczyt, zapis danych
  - Zakończenie transakcji
  - Akceptację lub wycofanie transakcji
- Rozwiązuje problemy awaryjności, wielodostępności i rozproszenia

# Własności transakcji ACID (1)

- Atomowość (Atomicity)

  Wykonanie całej transakcji albo niewykonanie Żadnej operacji składowej (odzwierciedlenie Świata rzeczywistego)

- Spójność (Consistency)

  Transakcja nie narusza spójności (w czasie wykonywania transakcji baza może być przejściowo niespójna)

# Własności transakcji ACID (2)

- Izolacja (Isolation)

  Transakcje wykonywane jednocześnie nie wpływają na siebie

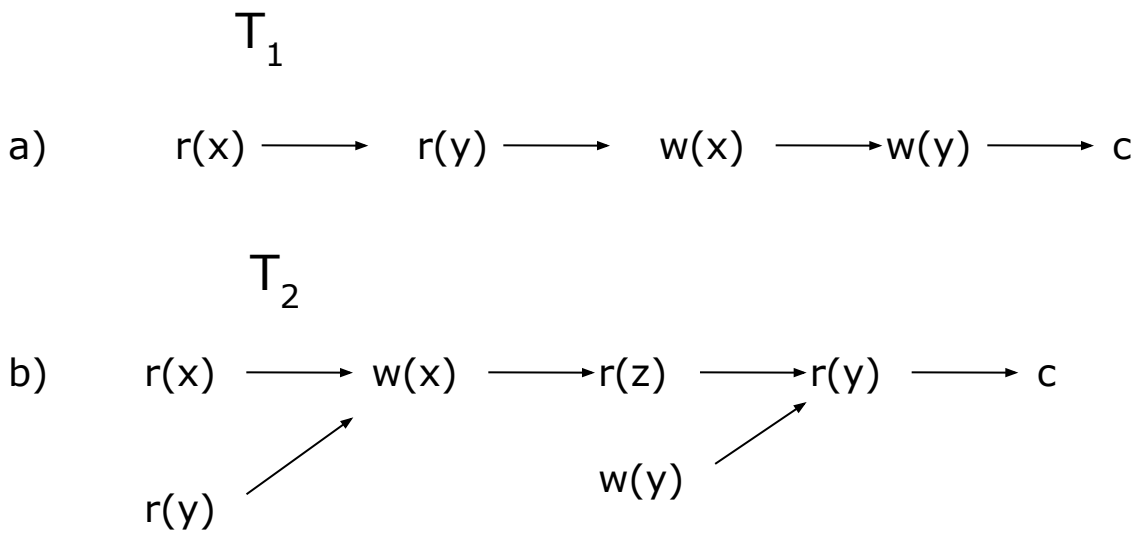
- Trwałość (Durability)

  Po zakończeniu transakcji zaktualizowane dane nie mogą zostać w żaden sposób utracone

# Reprezentacja transakcji

- Operacje transakcji:
  - Zapis – w(x)
  - Odczyt – r(x)
  - Zatwierdzenie – c
  - wycofanie – a
- Reprezentacja za pomocą grafu G(V,A):
  - V – węzły odpowiadające operacjom transakcji
  - A – krawędzie reprezentujące porządek na zbiorze operacji

# Reprezentacja transakcji - przykład



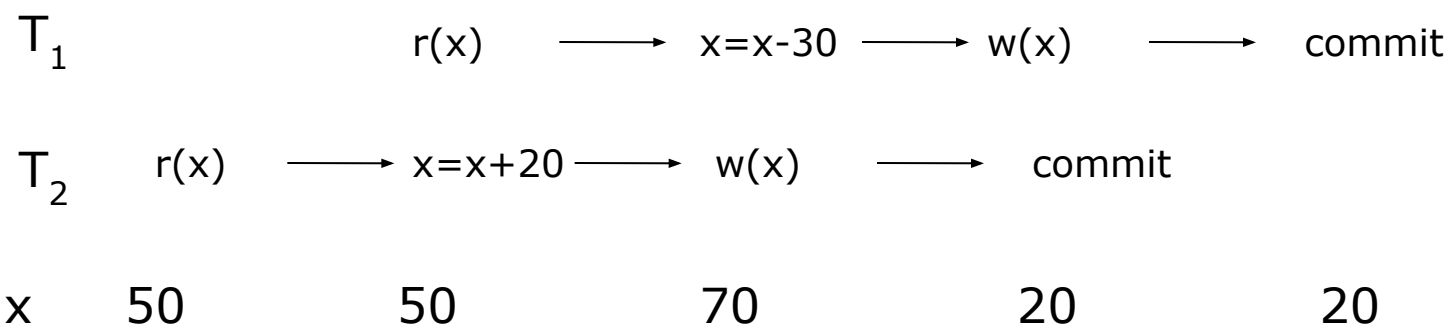
# Kontrola wielodostępu

- Konieczność zapewnienia dostępu do bazy danych wielu użytkownikom
- Zapewnienie możliwości wykonania współbieżnie transakcji
- Problemy wynikające z wielodostępu:
  - utrata zmian
  - niezatwierdzone zależności
  - niespójność

# Problemy wielodostępu – przykład(1)

- Utrata zmian

  x – stan konta – początek 50

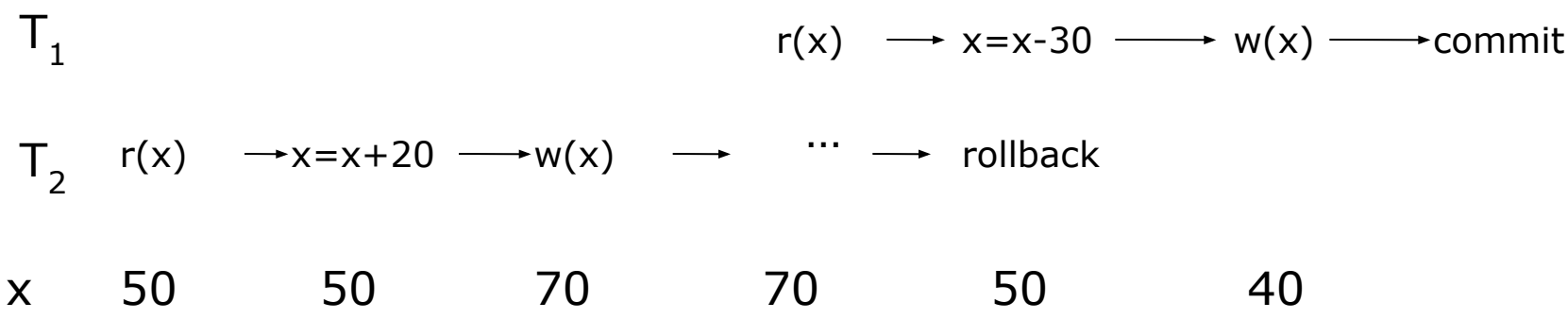



Stan konta po operacjach powinien wynieść 40. Została utracona informacja o zwiększeniu o kwotę 20

# Problemy wielodostępu – przykład(2)

- niezatwierdzone zależności

  x – stan konta – początek 50



# Problemy wielodostępu – przykład(3)

- Niespójność

stany kont początek x=50 y=30 z=40 s - suma

$T_1$ r(x) → x=x-30 → w(x) → r(z) → z=z+30 → w(z) → commit

$T_2$ s=0 → r(x) → s=s+x → r(y) → s=s+y → r(z) → s=s+z → commit

| | | | | | | |
|---|---|---|---|---|---|---|
| x | 50 | 50 | 20 | 20 | 20 | 20 |
| y | 30 | 30 | 30 | 30 | 30 | 30 |
| z | 40 | 40 | 40 | 40 | 70 | 70 |
| suma | 0 | 50 | 50 | 80 | 80 | 150 |

# Zakleszczenie transakcji

- Gdy dwie transakcje czekają

| T1 | T2 |
|---|---|
| Rlock(T1,Y) | |
| r(Y) | |
| | Rlock(T2,X) |
| | r(X) |
| | Wlock(T2,Y) |
| Wlock(T1,X) | wait |
| wait | wait |
| wait | wait |
| … | … |

# Metody wykrywania i rozwiązywania zakleszczeń

- Za pomocą grafu:
  - transakcje jako węzły
  - oczekiwanie transakcji $T_i$ na daną zablokowaną przez $T_i$ reprezentowane przez krawędź skierowaną
  - cykl w grafie oznacza zakleszczenie
  - eliminacja – wycofanie jednej z transakcji cyklu
  - graf sprawdzany
    - jeśli transakcja czeka zbyt długo – przekroczyła ustalony limit czasu
    - co określony czas