

TEORETYCZNE PODSTAWY INFORMATYKI

7/12/2015

WFAiS UJ, Informatyka Stosowana
I rok studiów, I stopień

Wykład 8a

2

Modele
danych:
relacyjne
bazy danych

- Relacje binarne: podstawowe własności
- Operacje dla zbiorów w algebrze relacyjnej
- Schemat bazy danych

Relacyjny model danych

3

- Jednym z najważniejszych zastosowań komputerów jest przechowywanie i przetwarzanie informacji.
- Relacyjny model danych opiera się na idei organizowania danych w zbiory dwuwymiarowych tabel nazywanych „relacjami”.
- Jest to uogólnienie modelu danych opartego na zbiorach, rozszerzającego relacje binarne do relacji o dowolnej krotności.
- Relacyjny model danych został pierwotnie opracowany z myślą o bazach danych oraz o systemach zarządzania bazami danych.
- Obecne zastosowania wykraczają poza ten pierwotny zakres.

Relacje

4

- Chociaż założyliśmy, że w ogólności elementy należące do zbiorów są niepodzielne, w praktyce często korzystnym rozwiązaniem jest przypisanie elementom pewnych struktur.
- Ważną strukturą dla elementów jest lista o stałej długości zwana krotką. Każdy element takiej listy nazywamy składową krotki.
- Zbiór elementów, z których każdy jest krotką o takiej samej liczności -powiedzmy k - nazywamy relacją. Licznością takiej relacji jest k . Jeśli liczność wynosi 2 mówimy o krotce lub relacji binarnej.

Iloczyn kartezjański $A \times B$

- ✚ Jest to zbiór par, z których pierwszy element pochodzi ze zbioru A , drugi ze zbioru B , czyli

$$A \times B = \{(a,b) : a \in A \text{ oraz } b \in B\}$$

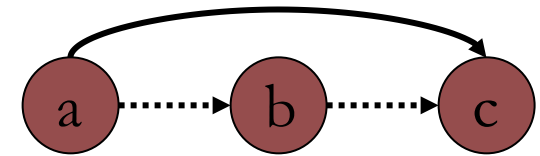
- ✚ Iloczyn kartezjański nie ma własności przemienności, $A \times B \neq B \times A$ (dla $A \neq B$)
- ✚ K -elementowy iloczyn kartezjański $A_1 \times A_2 \times A_3 \dots \times A_k$ to zbiór **k -krotek** (a_1, a_2, \dots, a_n)

Własności relacji binarnych

5

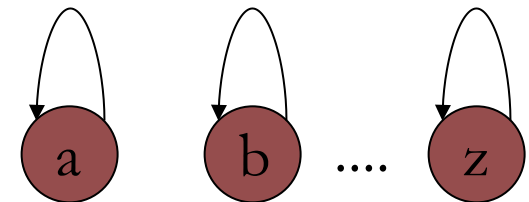
□ Przechodniość

- Niech R będzie relacją binarną na dziedzinie D .
- Mówimy, że relacja jest przechodnia jeśli zawsze gdy prawdziwe jest zarówno aRb i bRc , prawdziwe jest także aRc .
Np. relacja $>$



□ Zwrotność

- Niech R będzie relacją binarną na dziedzinie D .
- Mówimy, że relacja jest zwrotną jeśli dla każdego elementu a należącego do dziedziny, relacja zawiera parę aRa . Dla tych samych elementów dziedziny mogą też istnieć inne pary aRb .
Np. relacja \geq

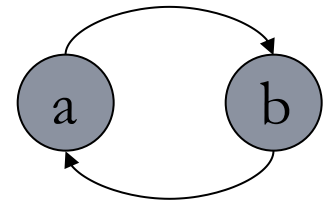


Własności relacji binarnych

6

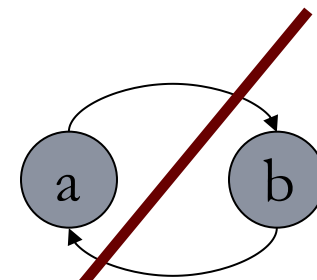
□ Symetria

- Niech R będzie relacją binarną na dziedzinie D .
- Mówimy, że relacja jest symetryczna jeśli jest odwrotnością samej siebie tzn. zarówno aRb i bRa ,
Np. relacja \neq

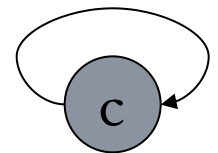


□ Antysymetria

- Niech R będzie relacją binarną na dziedzinie D .
- Mówimy, że relacja jest antysymetryczna jeśli aRb i bRa są jednocześnie prawdziwe tylko gdy $a=b$.
- Nie jest konieczne, by prawdziwe było aRa dla każdej wartości a należącej do dziedziny relacji antysymetrycznej.
Np. relacja $\geq, >$



nigdy



opcjonalnie

Własności relacji binarnych

7

- **Relacja porządku częściowego i całkowitego**
 - Relacja porządku częściowego jest to relacja binarna spełniająca własność przechodniości i antysymetrii.
 - Mówimy że jest to relacja porządku całkowitego jeśli poza przechodniością i antysymetrią spełnia także warunek, że wszystkie pary elementów należących do jej dziedziny są porównywalne.
 - Oznacza to, że jeśli R jest relacją porządku całkowitego oraz jeśli a i b są dowolnymi elementami tej dziedziny, to albo aRb , albo bRa jest prawdziwe (mówimy wtedy że relacja jest spójna).
 - Należy zauważyć że każdy porządek całkowity jest zwrotny, ponieważ możemy przyjąć a i b będące tym samym elementem – wymaganie porównywalności oznacza że aRa .
- **Relacja równoważności**
 - Relacja równoważności to relacją binarną, która jest zwrotna, symetryczna i przechodnia.
 - Dzieli ona swoją dziedzinę na klasy równoważności.

Relacyjny model danych

8

- Relacyjny model danych wykorzystuje pojęcie relacji (ang. relation) które jest bardzo mocno związane z przedstawioną wcześniej definicją z teorii zbiorów, jednak różni się w kilku szczegółach:
 - W relacyjnym modelu danych informacja jest przechowywana w tabelach.
 - Kolumny tabeli mają nadane konkretne nazwy i są atrybutami relacji.
 - Każdy wiersz w tabeli jest nazywany krotką i reprezentuje jeden podstawowy fakt.
 - Pojęcie relacji odwołuje się do każdej krotki.

Atrybuty relacji:

Zajęcia, StudentID, Ocena

Krotki:

(CS101, 12345, 5.0)

(CS101, 67890, 4.0)

...

| zajęcia | student ID | ocena |
|---------|------------|-------|
| CS101 | 12345 | 5.0 |
| CS101 | 67890 | 4.0 |
| EE200 | 12345 | 3.0 |
| EE200 | 22222 | 4.5 |
| CS101 | 33333 | 2.0 |
| PH100 | 67890 | 3.5 |

Relacyjny model danych

9

- **Tabele** możemy rozpatrywać w dwóch aspektach:
 - jako zbiór nazw kolumn
 - jako zbiór wierszy zawierających informacje.
- Pojęcie „**relacji**” odwołuje się do zbioru wierszy.
- Każdy wiersz reprezentuje jedną „krotkę” należąca do relacji, ich uporządkowanie nie ma znaczenia.
- Żadne dwa wiersze nie mogą mieć tych samych wartości we wszystkich kolumnach.
- Zbiór nazw kolumn (atrybutów) nazywamy **schematem** (ang. scheme) relacji.
- Kolejność atrybutów w schemacie relacji nie ma znaczenia, musimy jednak znać powiązania pomiędzy atrybutami i kolumnami w tabeli.

| zajęcia | student ID | ocena |
|---------|------------|-------|
| CS101 | 12345 | 5.0 |
| CS101 | 67890 | 4.0 |
| EE200 | 12345 | 3.0 |
| EE200 | 22222 | 4.5 |
| CS101 | 33333 | 2.0 |
| PH100 | 67890 | 3.5 |

Reprezentowanie relacji

10

- Podobnie jak w przypadku zbiorów istnieje wiele różnych sposobów reprezentowania relacji za pomocą struktur danych.
- Tabela postrzegana jako zbiór wierszy powinna być zbiorem struktur zawierających pola odpowiadające nazwom kolumn.

```
struct ZSO {  
    char Zajecia[5];  
    int StudentID;  
    char Ocena[3];  
}
```

- Sama **tabela** może być reprezentowana za pomocą:
 - ▣ **tablicy struktur tego typu**
 - ▣ **listy jednokierunkowej złożonej z takich struktur**
- Możemy identyfikować jeden lub więcej atrybutów jako „dziedzinę” relacji i traktować pozostałe atrybuty jako przeciwdziedzinę.

| zajęcia | student ID | ocena |
|---------|------------|-------|
| CS101 | 12345 | 5.0 |
| CS101 | 67890 | 4.0 |
| EE200 | 12345 | 3.0 |
| EE200 | 22222 | 4.5 |
| CS101 | 33333 | 2.0 |
| PH100 | 67890 | 3.5 |

Reprezentowanie relacji

11

- Zbiór relacji nazywamy **bazą danych**.
- Jedną z decyzji którą należy podjąć przy projektowaniu bazy danych to sposób w jaki przechowywane informacje powinny być rozłożone pomiędzy tabele.
- Najskuteczniejsze operacje na bazie danych polegają na wykorzystaniu wielu relacji do reprezentowania powiązanych ze sobą i wzajemnie skoordynowanych typów danych.
- Wykorzystując właściwe struktury danych możemy efektywnie przechodzić z jednej relacji do drugiej i pozyskiwać w ten sposób informacje z bazy danych której nie moglibyśmy otrzymać z pojedynczej relacji.
- **Zbiór schematów** dla różnych relacji w jednej bazie danych nazywamy schematem bazy danych.
 - **schemat bazy danych** - określa sposób organizowania informacji,
 - **zbiór krotek w każdej relacji** - stanowi właściwe informacje które są przechowywane.

Schemat bazy danych

12

| Zajęcia | Dzień | Godzina |
|---------|-------|---------|
| CS101 | Pn | 9.00 |
| CS101 | S | 9.00 |
| EE200 | Pt | 8.30 |
| EE200 | W | 13.00 |
| CS101 | Pt | 9.00 |
| PH100 | C | 8.15 |

| Zajęcia | Wymagania |
|---------|-----------|
| CS101 | CS100 |
| EE200 | EE005 |
| EE200 | CS100 |
| CS120 | CS101 |
| CS121 | CS120 |
| CS205 | CS101 |
| CS206 | CS121 |
| CS206 | CS205 |

| Zajęcia | Student ID | Ocena |
|---------|------------|-------|
| CS101 | 12345 | 5.0 |
| CS101 | 67890 | 4.0 |
| EE200 | 12345 | 3.0 |
| EE200 | 22222 | 4.5 |
| CS101 | 33333 | 2.0 |
| PH100 | 67890 | 3.5 |

| Zajęcia | Klasa |
|---------|---------|
| CS101 | Aula |
| EE200 | Hala |
| PH100 | Laborat |

Zapytania w bazie danych

13

- Operacja **insert(t,R)**
 - ▣ Dodajemy krotkę t do relacji R , jeśli relacja R nie zawiera jeszcze takiej krotki.
 - ▣ Operacja działa w podobny sposób jak operacja insert dla słowników i relacji binarnych.
- Operacja **delete(X,R)**
 - ▣ W tym przypadku X jest specyfikacją kilku krotek.
 - ▣ Składa się z elementów, po jednym dla każdego z atrybutów relacji R; każdy element (składowa) może być
 - wartością
 - symbolem *, co oznacza że dozwolona jest dowolna wartość.
 - ▣ Efektem wykonania tej operacji jest usunięcie wszystkich krotek zgodnych ze specyfikacją X.
Np. `delete((„CS101”,*,*),Zajęcia-StudentID-Ocena)`
- Operacja **lookup(X,R)**
 - ▣ Wynikiem tej operacji jest zbiór krotek z relacji R, które są zgodne ze specyfikacją X.

Klucze

14

- Wiele relacji w bazie danych możemy traktować jak funkcję odwzorowującą jeden zbiór atrybutów na pozostałe atrybuty.
 - Przykładowo, relacje **Zajęcia – StudentID – Ocena** możemy traktować jak funkcję, której dziedziną jest zbiór par **Zajęcia-StudentID**, a przeciwdziedziną wartość atrybutu **Ocena**.
- Ponieważ funkcje są prostszymi strukturami danych niż relacje, pomocna może być znajomość zbioru atrybutów, które mogą tworzyć dziedzinę funkcji. Taki zbiór atrybutów nazywamy kluczem.
- **Klucz relacji**
 - Jest to zbiór złożony z jednego lub większej liczby takich atrybutów, że relacja nigdy nie będzie zawierała dwóch krotek, których wartości będą takie same we wszystkich kolumnach należących do klucza.

Główne struktury przechowywania danych w relacjach

15

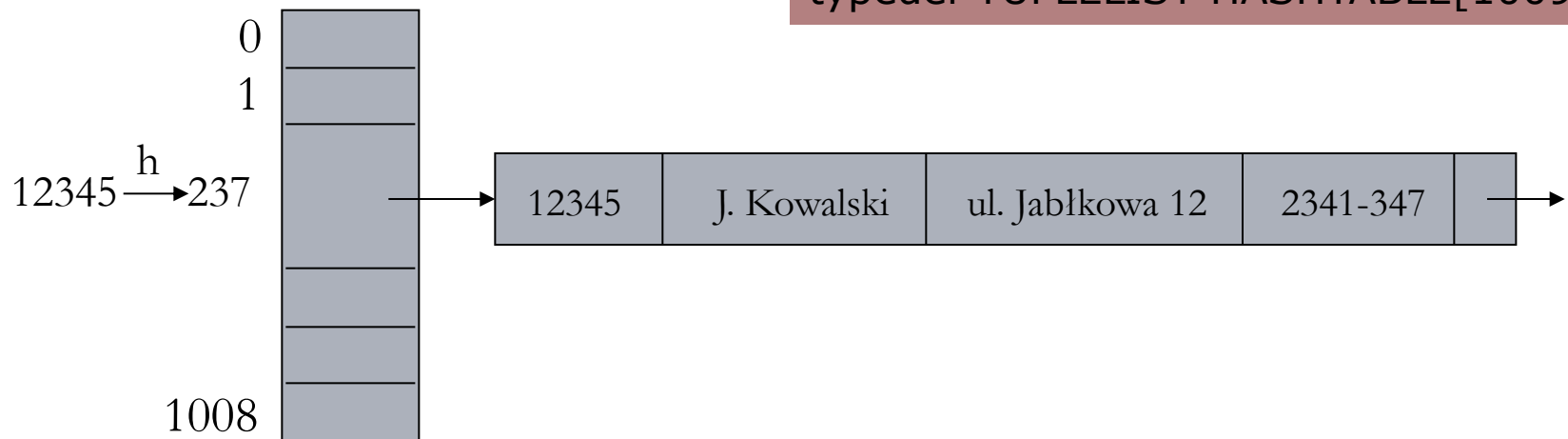
- **Drzewo przeszukiwania binarnego** z relacją „mniejszy od” na wartościach dziedziny, która wyznacza pozycje krotek w drzewie. Struktura może znacznie ułatwić wykonywanie operacji, w których daną jest wartość z dziedziny.
- **Tablica** wykorzystywana jako wektor własny z wartościami z dziedziny pełniącymi funkcję indeksu tablicy.
- **Tablica mieszająca**, w której mieszamy wartości z dziedziny w celu wyznaczenia właściwej komórki.
- Wybraną strukturę nazywamy **strukturą indeksu głównego** (ang. *primary index structure*) relacji.
 - **Główny** bo lokalizacja komórek jest wyznaczana przez tę strukturę.
 - **Index** jest strukturą danych ułatwiającą znajdowanie komórek dla danej wartości jednej lub kilku składowych szukanej komórki.

Struktura indeksu głównego

16

- Kluczem jest atrybut **StudentID**, będzie on dziedziną.
- Musimy wybrać funkcję mieszającą, np. **$h(x) = x \% 1009$** .
- Tablica złożona z 1009 nagłówek zawiera listę jednokierunkową struktur.

```
typedef struct TUPLE * TUPLELIST;  
struct TUPLE {  
    int StudentID;  
    char Nazwisko[30];  
    char Adres[60];  
    char Telefon[8];  
    TUPLELIST next;  
};  
typedef TUPLELIST HASHTABLE[1009];
```



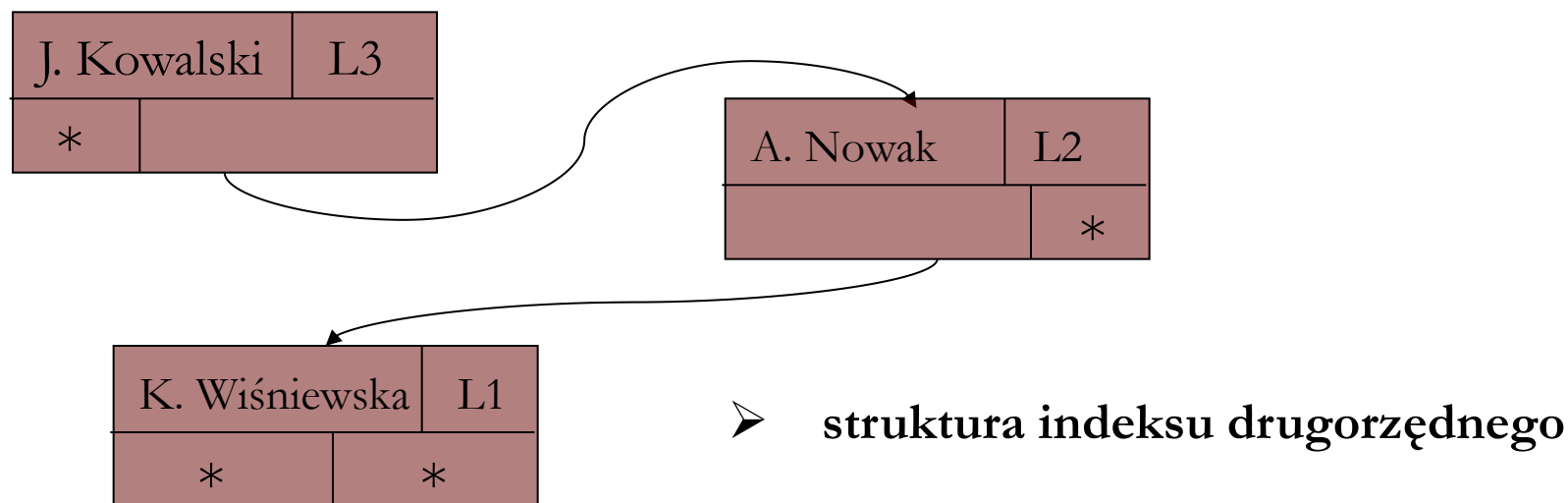
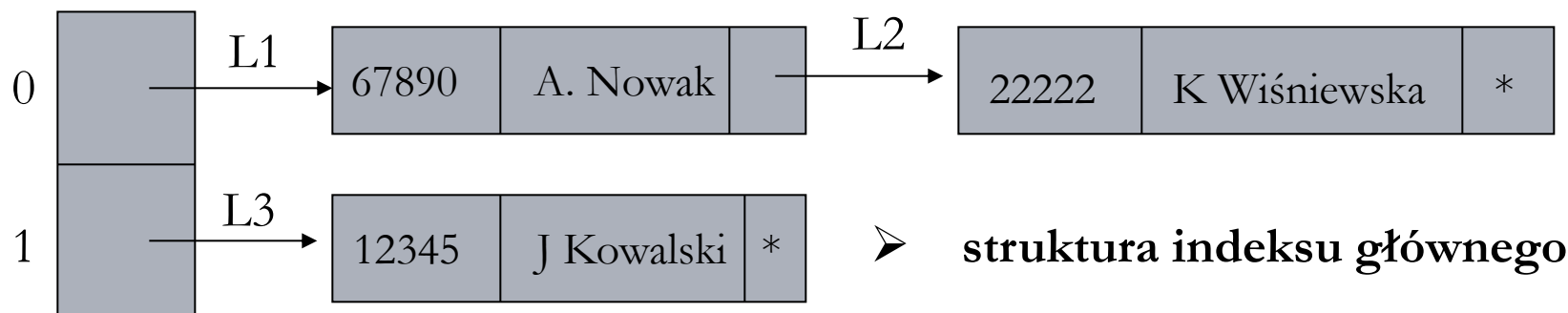
Struktura indeksu drugorzędnego

17

- Nie służy do pozycjonowania krotek wewnątrz całej struktury tylko do szybkiego znajdowania lokalizacji krotki której jedno z pól ma wartość zgodną z poszukiwaną.
- **Indeksem drugorzędnym** jest relacja binarna.
 - Indeks drugorzędny na atrybucie A relacji R jest zbiorem par (n, p) , gdzie:
 - **n** jest wartością atrybutu A
 - **p** jest wskaźnikiem do jednej z krotek ze struktury indeksu głównego dla relacji R, w której składowa A ma wartość n.

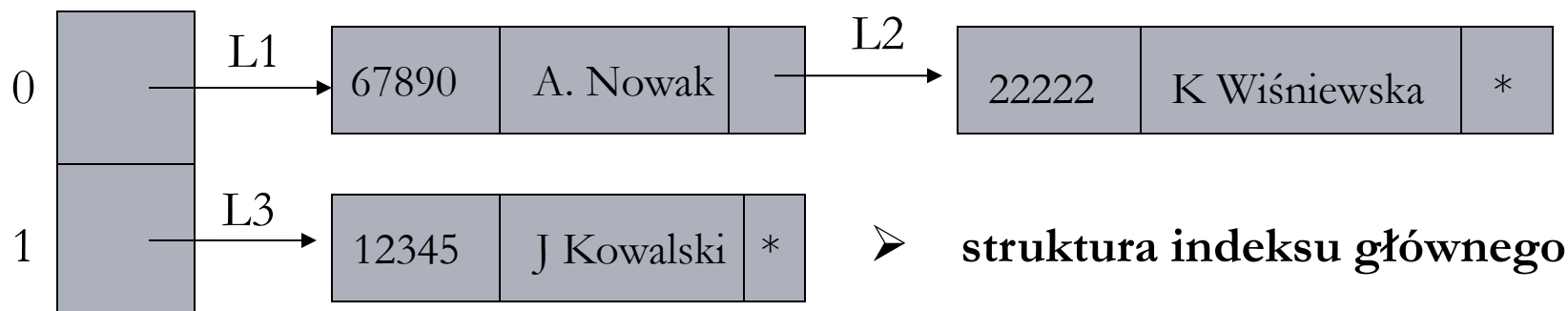
Struktura indeksu głównego i drugorzędowego

18



Struktura indeksu głównego

19



➤ **struktura indeksu głównego**

```
typedef struct KROTKA * KROTKALIST;  
struct KROTKA {  
    int StudentID;  
    char Nazwisko[30];  
    char Adres[60];  
    char Telefon[8];  
    KROTKALIST next;  
};  
typedef KROTKALIST HASHTABLE[2];
```

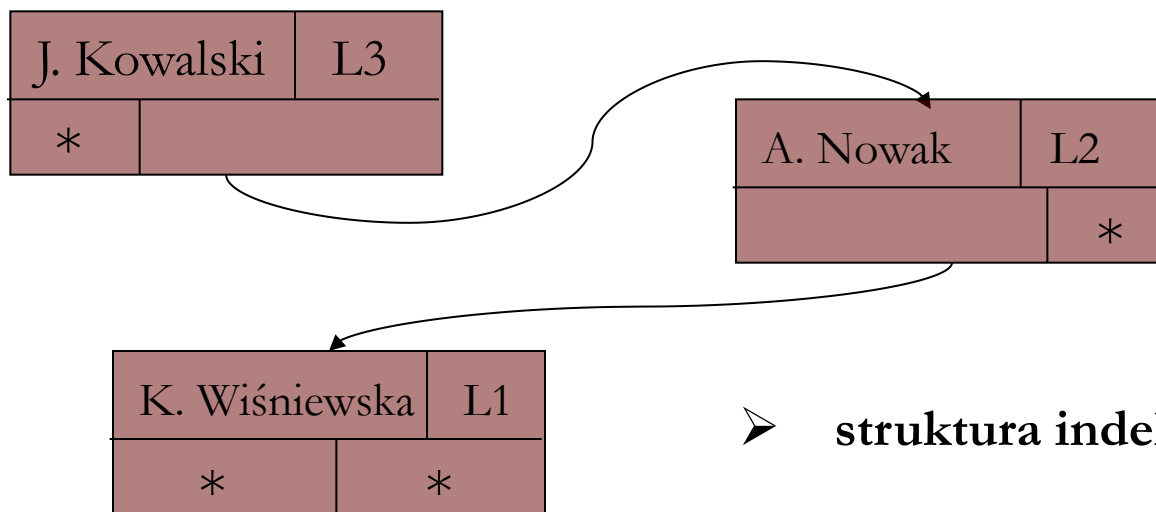
- Tablica mieszająca o atrybucie StudentID, pełniąca funkcje indeksu głównego.
- Krotki z informacją dotyczącą studenta przechowywane w formie struktur KROTKA w liście jednokierunkowej zajmującej pojedynczą komórkę tablicy mieszającej.

Struktura indeksu drugorzędnego

20

- NODE jest węzłem drzewa binarnego z dwoma polami, Nazwisko i toKrotka, czyli wartość elementu nazwisko i wskaźnik do krotki gdzie jest przechowywana inna informacja dotycząca tego studenta.
- Pozostałe dwa pola to wskaźniki do lewego i prawego dziecka węzła.

```
typedef struct NODE * TREE;  
struct NODE {  
    char Nazwisko[30];  
    KROTKALIST toKrotka;  
    TREE leftChild;  
    TREE rightChild;  
};
```



➤ struktura indeksu drugorzędnego

Analizowanie struktury indeksu drugorzędowego

21

- Jeżeli dla danej relacji istnieje jeden lub więcej indeksów drugorzędnych, operacje wstawiania i usuwania krotek stają się nieco trudniejsze.
 - ▣ **Wstawianie:**

Jeśli wstawiamy nową krotkę z wartością n atrybutu A , musimy utworzyć parę (n, p) , gdzie p wskazuje na nowy element w strukturze indeksu głównego.
Następnie, musimy wstawić tę samą parę (n, p) do struktury indeksu drugorzędowego.
 - ▣ **Usuwanie:**

Kiedy usuwamy krotkę z wartością n atrybutu A , musimy najpierw zachować wskaźnik – nazwijmy go p – do usuwanej krotki.
Następnie przechodzimy do struktury indeksu drugorzędowego i sprawdzamy wszystkie pary z pierwszą składową zawierającą wartość n , aż znajdziemy tę, której druga składowa ma wartość p .
Znaleziona w ten sposób para jest teraz usuwana ze struktury indeksu drugorzędowego.

Poruszanie się wśród wielu relacji

22

- Do tej pory rozważaliśmy wyłącznie operacje na pojedynczych relacjach, takie jak znajdowanie krotki dla danych wartości jednej lub kilku jej składowych.
- Możliwości modelu relacyjnego można jednak w pełni docenić w momencie, gdy rozważamy operacje wymagające „poruszania się”, lub „przechodzenia” z jednej relacji do drugiej.
- Aby znaleźć odpowiedź na pytanie:

„Jaką ocenę uzyskał J. Kowalski z przedmiotu CS101?”

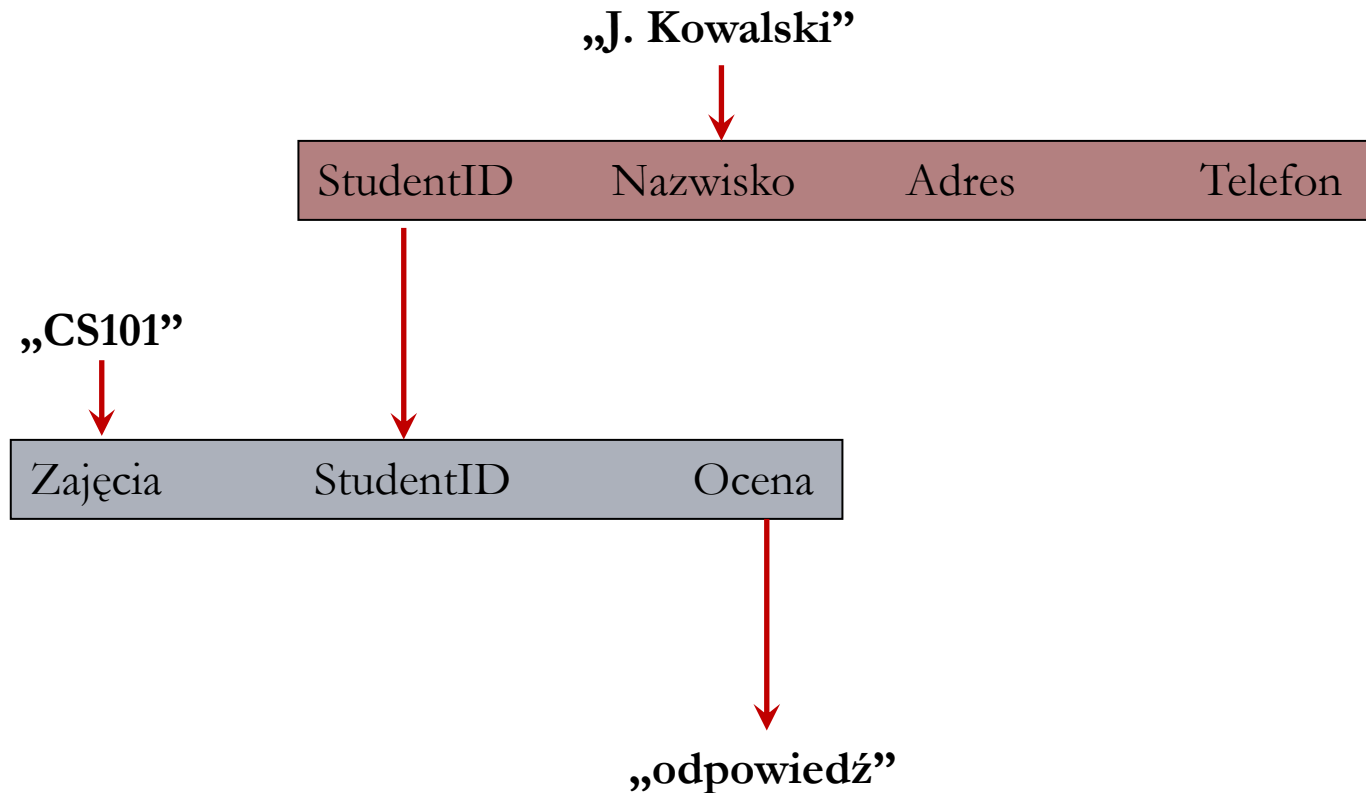
musimy:

- odwołać się do relacji StudentID-Nazwisko-Adres-Telefon i przełożyć dane nazwisko „J. Kowalski” na odpowiedni numer indeksu (możliwość istnienia duplikatu nazwiska ale nie numeru indeksu),
- odwołać się do relacji Zajęcia-StudentID-Ocena i wyznaczyć krotkę mającą w polu Zajęcia wartość „CS101” a w polu numer indeksu wyznaczoną poprzednio wartość,
- odczytać wartość umieszczoną w polu Ocena.

Diagram wykonania zapytania:

„Jaka ocenę uzyskał J. Kowalski z przedmiotu CS101?”

23

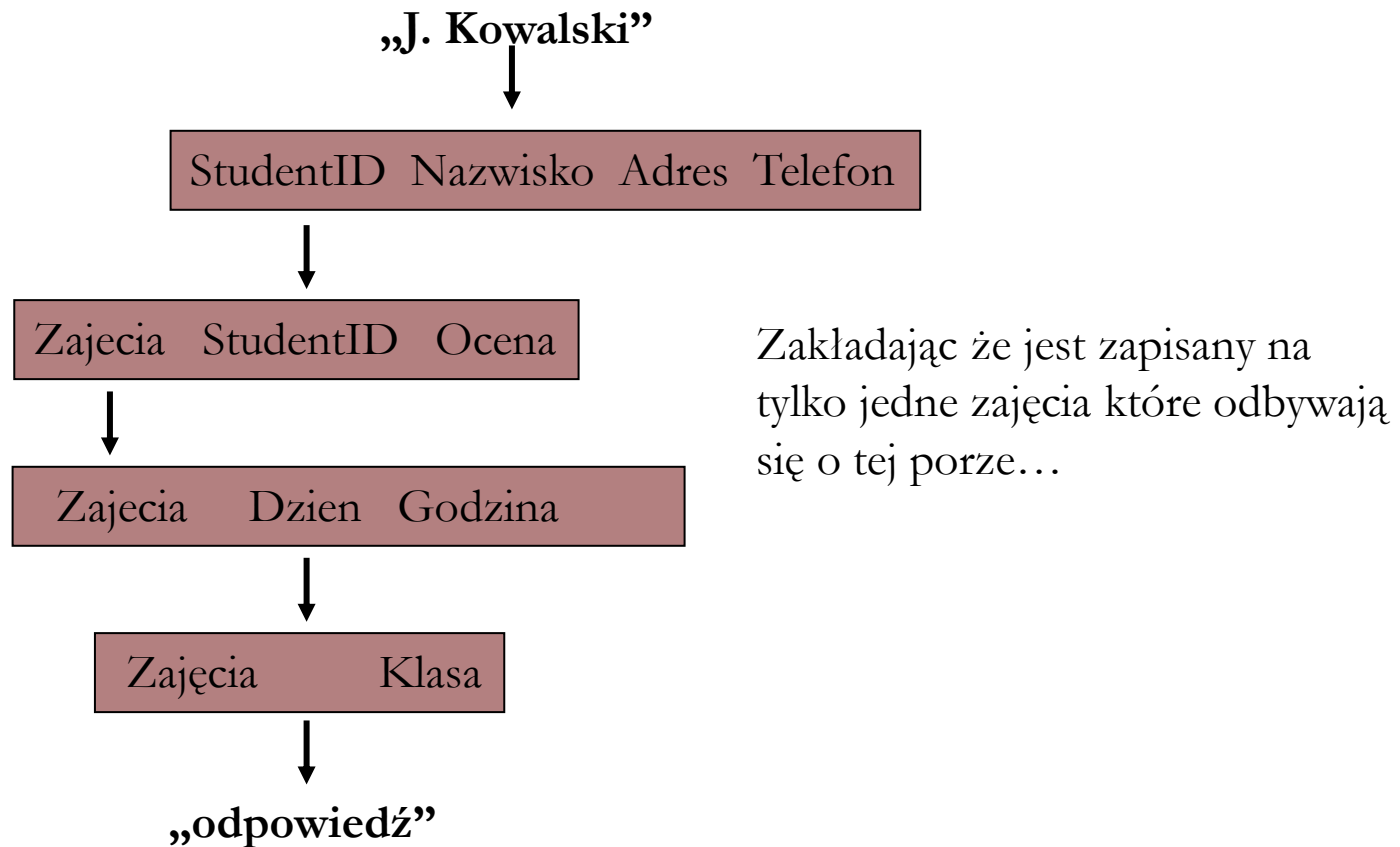


Czas realizacji zapytania będzie dużo krótszy jeżeli wykorzystamy indeksowanie drugorzędne

Diagram wykonania zapytania:

„*Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?*”

24



Indeksowanie drugorzędowe bardzo przyspiesza czas wykonania.

Projektowanie bazy danych

25

- **Projektowanie I:** wybór schematu baz danych
 - rozdzielamy informacje budując kilka relacji (krotek) zamiast umieszczać je w jednej dużej krotce,
 - nie należy rozdzielać atrybutów reprezentujących powiązane ze sobą informacje.
- **Projektowanie II:** wybór klucza
 - jeden z ważniejszy aspektów projektowania bazy danych,
 - nie istnieje „jedyna” właściwa metoda wybierania klucza.
- **Projektowanie III:** wybór indeksu głównego
 - ma zdecydowany wpływ na szybkość z jaką możemy wykonywać „typowe” zadanie.
- **Projektowanie IV:** kiedy tworzyć indeks drugorzędny?
 - utworzenie ułatwia wykonywanie operacji wyszukiwania krotki dla danej wartości jednej lub więcej składowych,
 - każdy indeks drugorzędny wymaga dodatkowego czasu wstawiania i usuwania informacji z relacji.

Podsumowanie

26

- Istnieje wiele istotnych własności **relacji binarnych**. Do najważniejszych należą: **zwrotność, przechodniość, symetria i antysymetria**. Relacja porządku częściowego, porządku całkowitego oraz relacja równoważności to specyficzne rodzaje relacji binarnych;
- Dwuwymiarowe tabele zwane relacjami, są uniwersalnym sposobem przechowywania informacji. Wiersze relacji nazywamy **krotkami**, zaś kolumny noszą nazwę **atrybutów**.
- „**Indeks główny**” reprezentuje krotki relacji w formie struktury danych i rozdziela je w taki sposób, by ułatwić (przyśpieszyć) operacje wykorzystujące dane wartości należące do „dziedziny” indeksu;

Podsumowanie

27

- **„Kluczem”** relacji jest zbiór atrybutów, które jednoznacznie określają wartości wszystkich pozostałych atrybutów tej samej relacji. Klucz jest często wykorzystywany jako dziedzina indeksu głównego;
- **„Indeksy drugorzędowe”** są strukturami danych ułatwiającymi operacje, w których dane są wartości konkretnych atrybutów nie będących zazwyczaj częścią indeksu głównego. Ułatwiają szybkie odczytanie lub zmodyfikowanie informacji zawartych w tabeli.

Wykład 8b

28

Modele
danych:
relacyjne
bazy danych

- Algebra relacyjna
- Prawa algebry relacyjnej
- Przykład wykorzystania

Algebra relacyjna

29

- **Algebra relacyjna** (ang. relational algebra) to specjalny język opracowany w celu ułatwienia i sformalizowania zapytań realizowanych w bazach danych.
- Umożliwia przekształcanie wyrażeń realizujących zapytania za pomocą odpowiednich **praw algebraicznych**.

Operandy algebry relacyjnej

30

- W algebrze relacyjnej **operandami są relacje**.
- Operandy mogą być albo stałymi (konkretnymi relacjami) albo zmiennymi reprezentującymi nieznaną relację.
- Każdy operand jest zgodny ze specyficznym schematem – jest listą atrybutów będących nazwami kolumn relacji.
- Przykład:
 - **Schematem relacji** jest $\{A, B, C\}$, zaś należące do niej krotki to $(0, 1, 2)$, $(0, 3, 4)$ oraz $(5, 2, 3)$.

| A | B | C |
|---|---|---|
| 0 | 1 | 2 |
| 0 | 3 | 4 |
| 5 | 2 | 3 |

Operatory dla zbiorów w algebrze relacyjnej

31

□ Suma, przecięcie oraz różnica zbiorów

- Dodatkowe założenie w stosunku do tego co już znamy z operacji na zbiorach to to, że schematy operandów muszą być takie same.

relacja R

| A | B |
|---|---|
| 0 | 1 |
| 2 | 3 |

relacja S

| A | B |
|---|---|
| 0 | 1 |
| 4 | 5 |

$R \cup S$

| A | B |
|---|---|
| 0 | 1 |
| 2 | 3 |
| 4 | 5 |

$R \cap S$

| A | B |
|---|---|
| 0 | 1 |

$R \setminus S$

| A | B |
|---|---|
| 2 | 3 |

- Relacje są zbiorami nie mogą więc zawierać dwóch lub więcej kopii tej samej krotki.

Operator selekcji (ang. selection operator)

32

- Operuje na pojedynczej relacji będącej jego operandem, ale zawiera także dodatkowe wyrażenia warunkowe stanowiące jego parametry.
- Operator selekcji zapisujemy w postaci:

$$\sigma_C(R)$$

R – relacja

C – warunek

- Warunek C może zawierać stałe, jak i operandy będące atrybutami ze schematu relacji R.
- Operatorami wykorzystywanymi w warunku C są typowe wyrażenia warunkowe z języka programowania C, czyli wyrażenia złożone z porównań arytmetycznych oraz logicznych łączników.
- Wynikiem operacji jest relacja której schemat jest identyczny ze schematem relacji R.
- W relacji tej umieszczamy wszystkie krotki t z relacji R, dla których warunek C jest prawdziwy po podstawieniu za każdy atrybut A właściwej dla niej składowej krotki t.

Przykład: Relacja ZSO Zajęcia-StudentID-Ocena

33

| Zajęcia | Student ID | Ocena |
|---------|------------|-------|
| CS101 | 12345 | 5.0 |
| CS101 | 67890 | 4.0 |
| EE200 | 12345 | 3.0 |
| EE200 | 22222 | 4.5 |
| CS101 | 33333 | 2.0 |
| PH100 | 67890 | 3.5 |

Operator selekcji

$\sigma_{\text{Zajęcia} = \text{„CS101”}}$ (ZSO)

| Zajęcia | Student ID | Ocena |
|---------|------------|-------|
| CS101 | 12345 | 5.0 |
| CS101 | 67890 | 4.0 |
| CS101 | 33333 | 2.0 |

Operator rzutowania (ang. projection operator)

34

- Operuje na pojedynczej relacji będącej jego operandem, ale zawiera także dodatkowe wyrażenia warunkowe stanowiące jego dodatkowe parametry.
- Operator rzutowania zapisujemy w postaci:

$$\pi_{B_1, B_2, \dots, B_n}(R)$$

R – relacja

B₁, B₂, ..., B_n - atrybuty

- Jeśli R jest relacją ze zbiorem atrybutów $\{A_1, A_2, \dots, A_k\}$ oraz (B_1, B_2, \dots, B_n) jest listą pewnych atrybutów A , to $\pi_{B_1, B_2, \dots, B_n}(R)$, czyli rzutowanie relacji R na atrybuty B_1, B_2, \dots, B_n jest zbiorem krotek utworzonych przez wybranie z każdej krotki t tylko atrybutów B_1, B_2, \dots, B_n .
- Jedna lub więcej krotek może posiadać te same wartości atrybutów B_1, B_2, \dots, B_n .
- Jako wynik operacji rzutowania pojawia się tylko jedna taka krotka.

Przykład: Relacja ZSO Zajęcia-StudentID-Ocena

35

| zajęcia | student ID | ocena |
|---------|------------|-------|
| CS101 | 12345 | 5.0 |
| CS101 | 67890 | 4.0 |
| EE200 | 12345 | 3.0 |
| EE200 | 22222 | 4.5 |
| CS101 | 33333 | 2.0 |
| PH100 | 67890 | 3.5 |

Operator rzutowania

$\pi_{\text{StudentID}} (\sigma_{\text{Zajecia} = \text{„CS101”}} (\text{ZSO}))$

| Student ID |
|------------|
| 12345 |
| 67890 |
| 33333 |

Operator łączenia (ang. join operator)

36

- Umożliwia nam przechodzenie z jednej relacji do drugiej.
- Operator łączenia zapisujemy w postaci **&&**
- Przypuśćmy, że mamy dwie relacje R i S, których zbiory atrybutów (schematy) mają odpowiednio postać $\{A_1, A_2, \dots, A_n\}$ oraz $\{B_1, B_2, \dots, B_m\}$
- Z obu zbiorów wybieramy po jednym atrybucie – powiedzmy A_i i B_j – i te atrybuty są parametrami naszej operacji złączenia, której argumentami są relacje R i S.
- Złączenie relacji R i S zapisujemy:

$$R \quad \&\& \quad S \\ A_i = B_j$$

i jest utworzone w wyniku porównania każdej krotki r z relacji R z każdą krotką s z relacji S.

- Jeśli składowa r odpowiadająca atrybutowi A_i jest równa składowej s odpowiadającej atrybutowi B_j to tworzymy jedną krotkę.
- Schemat złączonej relacji jest $\{A_1, A_2, \dots, A_n, B_1, B_2, \dots, B_{j-1}, B_{j+1}, \dots, B_m\}$
- A więc atrybut B_j się nie pojawia.
- Jeżeli atrybuty A_i i B_j mają tę samą nazwę to mówimy o złączeniu naturalnym.

Przykład

37

ZDG

| Zajęcia | Dzień | Godzina |
|---------|-------|---------|
| CS101 | Pn | 9.15 |
| CS101 | S | 9.15 |
| EE200 | Pt | 8.30 |
| EE200 | W | 13.00 |
| CS101 | Pt | 9.15 |

ZK

| Zajęcia | Klasa |
|---------|---------|
| CS101 | Aula |
| EE200 | Hala |
| PH100 | Laborat |

Operacja łączenia

ZK **&&** **ZDG**
Zajęcia=Zajęcia

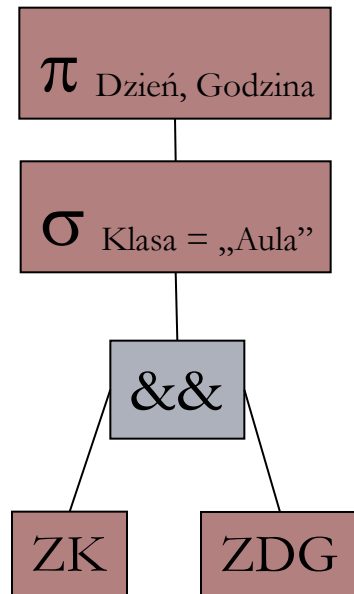
| Zajęcia | Klasa | Dzień | Godzina |
|---------|-------|-------|---------|
| CS101 | Aula | Pn | 9.15 |
| CS101 | Aula | S | 9.15 |
| EE200 | Hala | Pt | 8.30 |
| EE200 | Hala | W | 13.00 |
| CS101 | Aula | Pt | 9.15 |

Drzewa wyrażeń dla algebry relacyjnej

38

Drzewo wyrażenia algebry relacyjnej:

$\pi_{\text{Dzień, Godzina}} (\sigma_{\text{Klasa} = \text{„Aula”}} (\text{ZK} \ \&\& \ \text{ZDG}))$



Wynik po realizacji drzewa wyrażenia

| Dzień | Godzina |
|-------|---------|
| Pn | 9.15 |
| S | 9.15 |
| Pt | 9.15 |

SQL – język oparty na algebrze relacyjnej

39

- Wiele współczesnych systemów baz danych wykorzystuje język **SQL** (*ang. Structured Query Language* – strukturalny język zapytań) do wyrażania zapytań.

Operacja: $\pi_{\text{StudentID}} (\sigma_{\text{Zajęcia} = \text{„CS101”}} (\text{ZSO}))$

```
SELECT StudentID
FROM ZSO
WHERE Zajęcia=„CS101”
```

Operacja: $\pi_{\text{Dzień, Godzina}} (\sigma_{\text{Klasa} = \text{„Aula”}} (\text{ZK} \ \&\& \ \text{ZDG}))$

```
SELECT Dzień, Godzina
FROM ZK, ZDG
WHERE ZK.Zajęcia = ZGD.Zajęcia AND Klasa = „Aula”
```

Implementowanie operacji algebry relacyjnej

40

□ **Suma, przecięcie i różnica:**

- Można implementować analogicznie jak dla zbiorów.
- Należy przewidzieć sposób eliminacji duplikatów na pewnym etapie tych operacji.
- Można wykorzystać indeks.

□ **Rzutowanie:**

- Operacja wymaga przejrzania wszystkich krotek i stworzenia kopii pozbawionych składowych odpowiadających atrybutom, których nie ma na liście rzutowania.
- Po wyznaczeniu rzutowania, np. $S = \pi_L(R)$, dla pewnej relacji R i liście atrybutów L , musimy wyeliminować duplikaty (stosujemy jedną z omówionych już metod).

Implementowanie operacji algebry relacyjnej

41

□ **Selekcja:**

- Wykonywanie operacji selekcji $S = \sigma_C(R)$ na relacji R , dla której nie zdefiniowano żadnych indeksów.
- Musimy przeanalizować wszystkie krotki w tej relacji w celu sprawdzenia warunku C .
- Jeżeli takie indeksy istnieją oraz można je wykorzystać do całkowitego lub częściowego sprawdzenia warunku C to bardzo przyspiesza to czas wykonania operacji.

□ **Łączenie:**

- Istnieje szereg metod łączenia, różniących się czasem wykonania.
- Tylko wymieńmy: łączenie pętli zagnieżdżonej (ang. nested loop join), łączenie indeksowe (ang. index-join), łączenie przez sortowanie (ang. sort-join).
- Łączenie indeksowe wymaga istnienia indeksu na jednym z atrybutów wykorzystywanych do łączenia, łączenie przez sortowanie może być wykonywane na dowolnych relacjach.

Prawa algebraiczne dla relacji

42

- Podobnie jak w przypadku innych algebr, przekształcanie wyrażeń algebry zapytań umożliwia często „optymalizację” zapytań.
- Oznacza to, że możemy przekształcić kosztowne obliczeniowo wyrażenie w równoważne, którego obliczenie charakteryzuje się niższym kosztem.
- Podczas gdy przekształcenie wyrażeń arytmetycznych lub logicznych umożliwia niekiedy uzyskanie oszczędności rzędu kilku operacji, właściwe przekształcenia zastosowane do algebry relacyjnej może w znacznym stopniu skrócić czas potrzeby do wyznaczenia wartości wyrażenia.

Prawa dla łączenia:

- Operator łączenia jest w pewnym sensie przemienny, w innym nie jest. Jeżeli istotna jest kolejność atrybutów to nie jest, **S && R** ma inne kolumny niż **R && S**.
- Operator łączenia nie zawsze spełnia warunki prawa łączności.

Prawa algebraiczne dla relacji

43

□ Prawa dla selekcji:

- Najbardziej przydatne prawa algebry relacyjnej dotyczą operatora selekcji. Staramy się dokonywać selekcji na jak najwcześniejszym etapie.

- **Prawa przenoszenia selekcji** (ang. *selection pushing*)

$$\begin{aligned}(\sigma_C (R \ \&\& \ S)) &== (\sigma_C (R) \ \&\& \ S) \\(\sigma_C (R \ \&\& \ S)) &== (R \ \&\& \ \sigma_C (S))\end{aligned}$$

- **Prawo podziału selekcji** (ang. *selection splitting*)

$$\sigma_{(C \ \text{AND} \ D)} (R) == \sigma_C (\sigma_D (R))$$

- **Prawo przemienności selekcji**

$$\sigma_C (\sigma_D (R)) == \sigma_D (\sigma_C (R))$$

- Operacje selekcji możemy przenosić poniżej sumy, przecięcia i różnicy zbiorów.

Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?”

44

- **Wykonanie tego zapytania** wymaga przechodzenia pomiędzy czterema relacjami:
 - ZSO (Zajęcia-StudentID-Ocena)
 - SNAT (StudentID-Nazwisko-Adres-Telefon)
 - ZDG (Zajęcia-Dzień-Godzina)
 - ZK (Zajęcia-Klasa)
- Aby wykonać wyrażenie algebraiczne dla tego zapytania rozpoczniemy od **złączenia „naturalnego”** dla wszystkich czterech relacji.
 - ZSO && SNAT (porównując StudentID)
 - (ZSO && SNAT) && ZDG (porównując Zajęcia)
 - ((ZSO && SNAT) && ZDG) && ZK (porównując Zajęcia)
- Relacja wynikowa to: {Zajęcia, StudentID, Ocena, Nazwisko, Adres, Telefon, Dzień, Godzina, Klasa}

Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?"

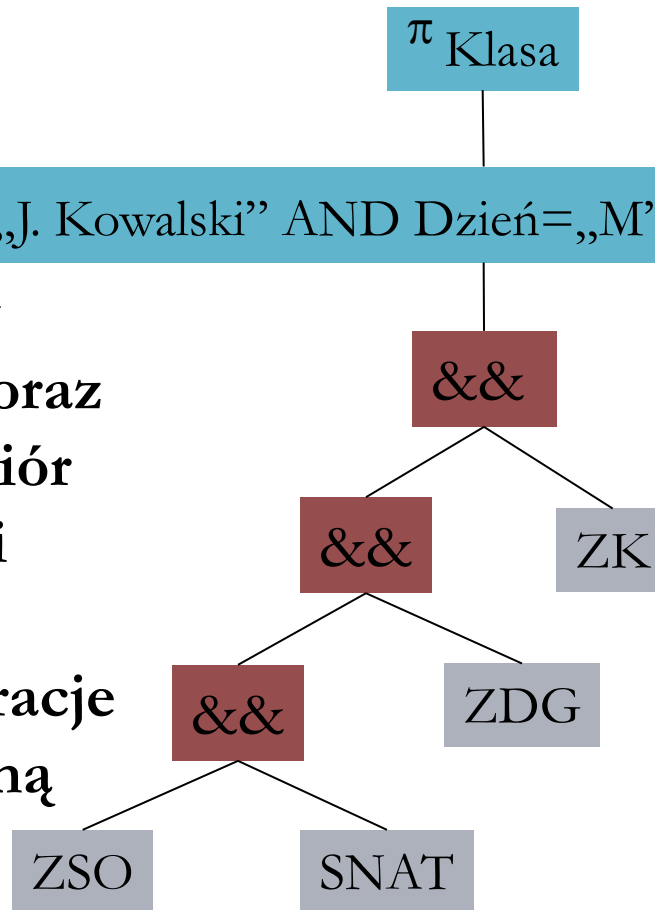
45

π Klasa

σ Nazwisko = „J. Kowalski” AND Dzień=„M” AND Godzina=„9.00”

Skonstruowaliśmy olbrzymią relację oraz ograniczyliśmy zbiór danych do 1 krotki

Wykonaliśmy operacje rzutowania na jedną składową.



Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?"

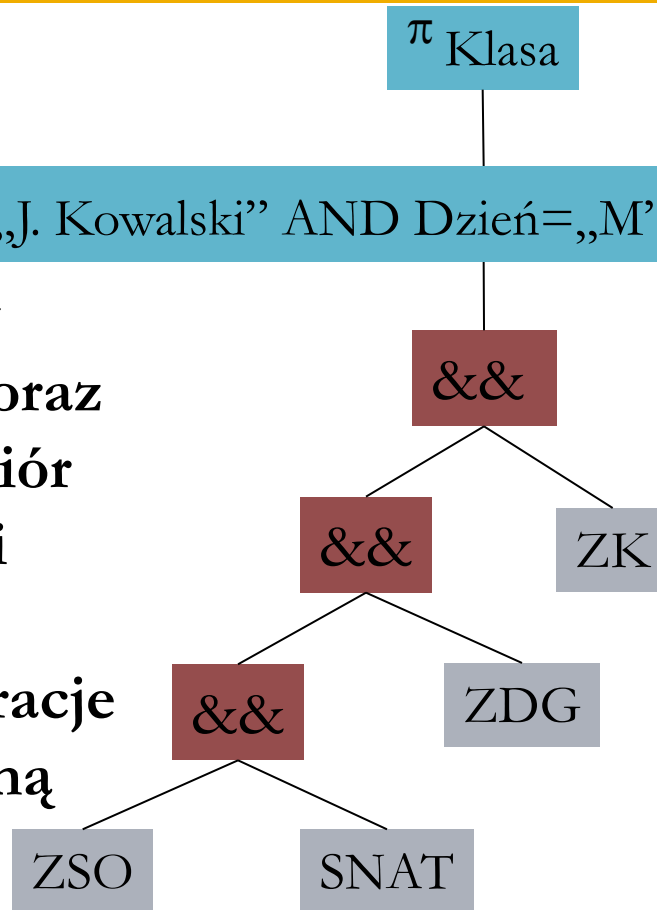
46

π Klasa

σ Nazwisko = „J. Kowalski” AND Dzień=„M” AND Godzina=„9.00”

Skonstruowaliśmy
olbrzymią relację oraz
ograniczyliśmy zbiór
danych do 1 krotki

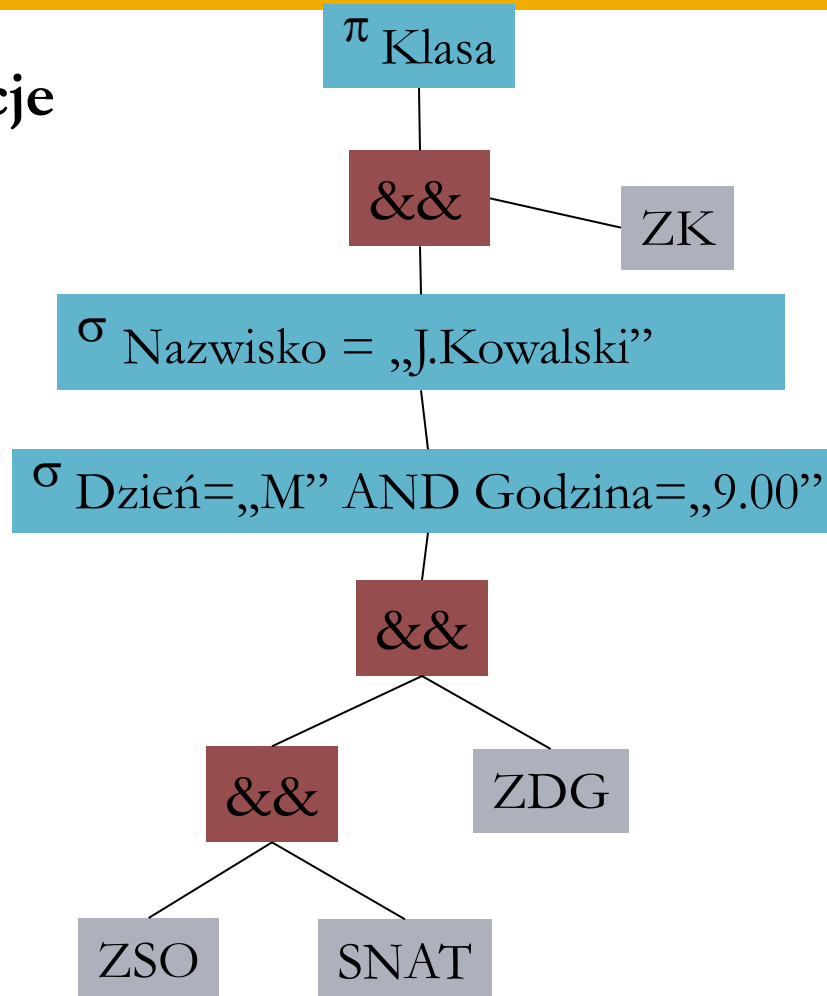
Wykonaliśmy operacje
rzutowania na jedną
składową.



Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?"

47

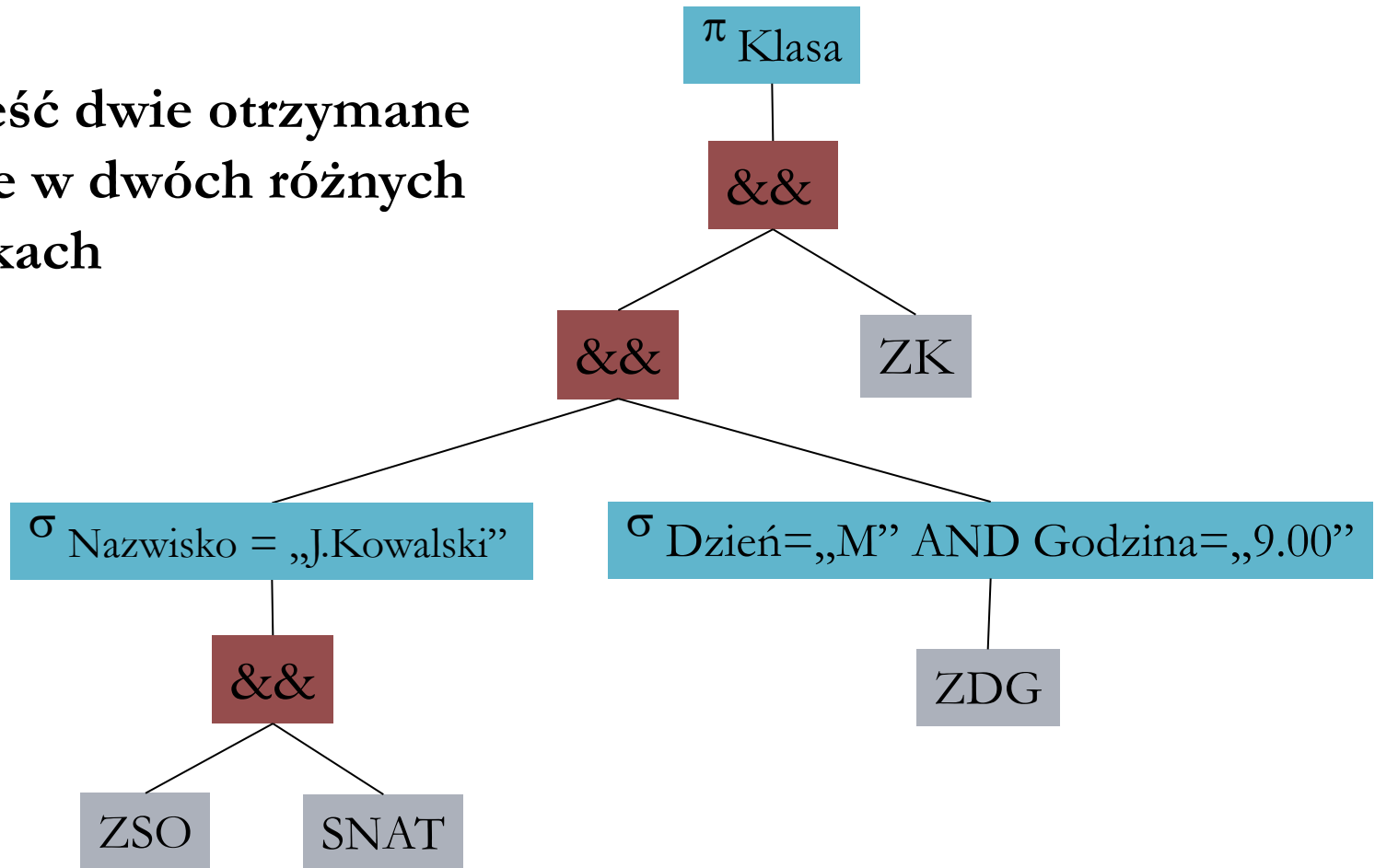
Rozdziel selekcje



Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?”

48

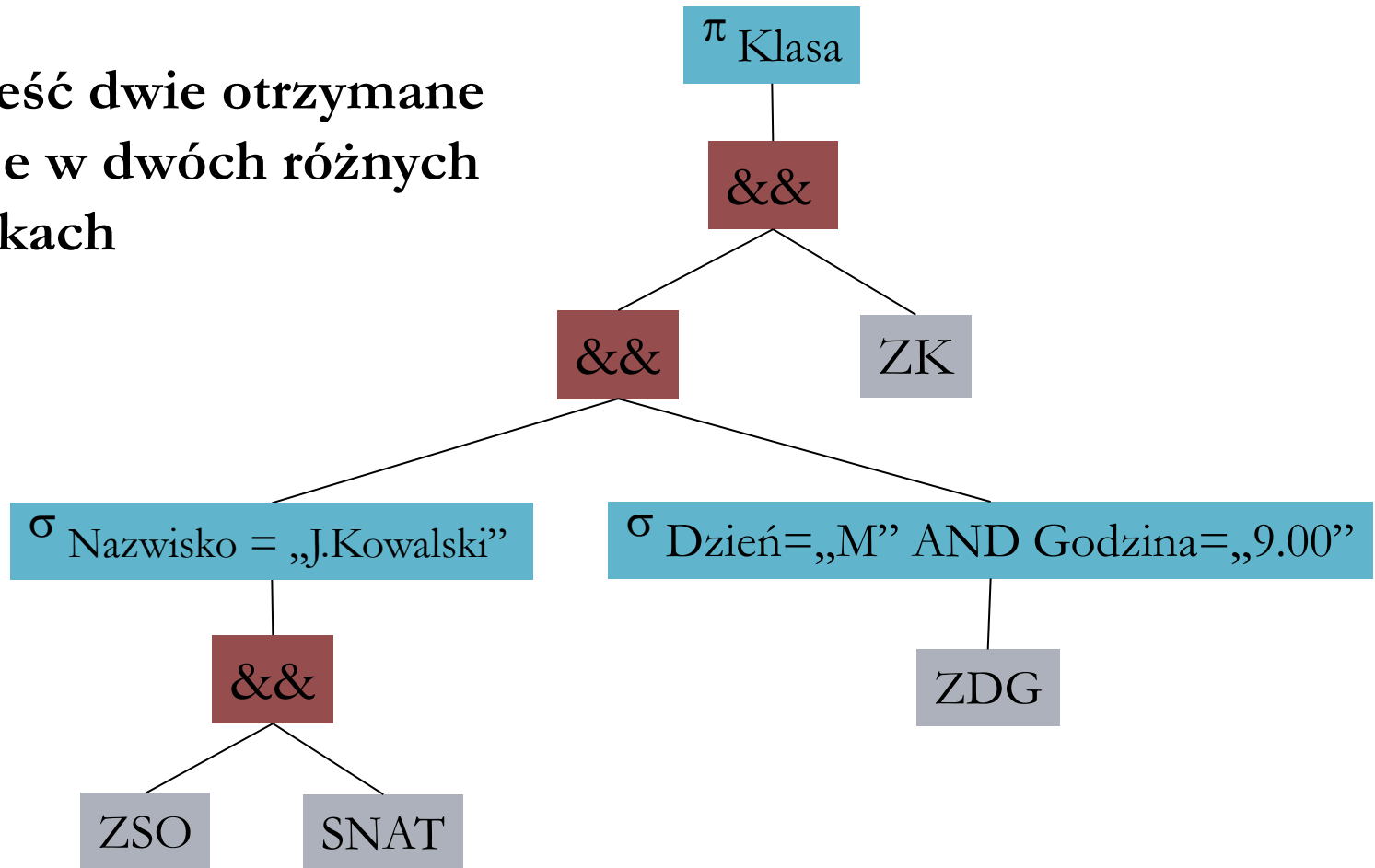
Przenieść dwie otrzymane selekcje w dwóch różnych kierunkach



Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?”

49

Przenieść dwie otrzymane selekcje w dwóch różnych kierunkach



Prawa dla rzutowania

50

- Rzutowanie możemy przenosić poniżej sum:
 - $\pi_L(R \cup S) = (\pi_L(R) \cup \pi_L(S))$
- ale nie poniżej przecięć:
 - $\pi_L(R \cap S) \neq (\pi_L(R) \cap \pi_L(S))$
- **Prawo przenoszenia rzutowania:**

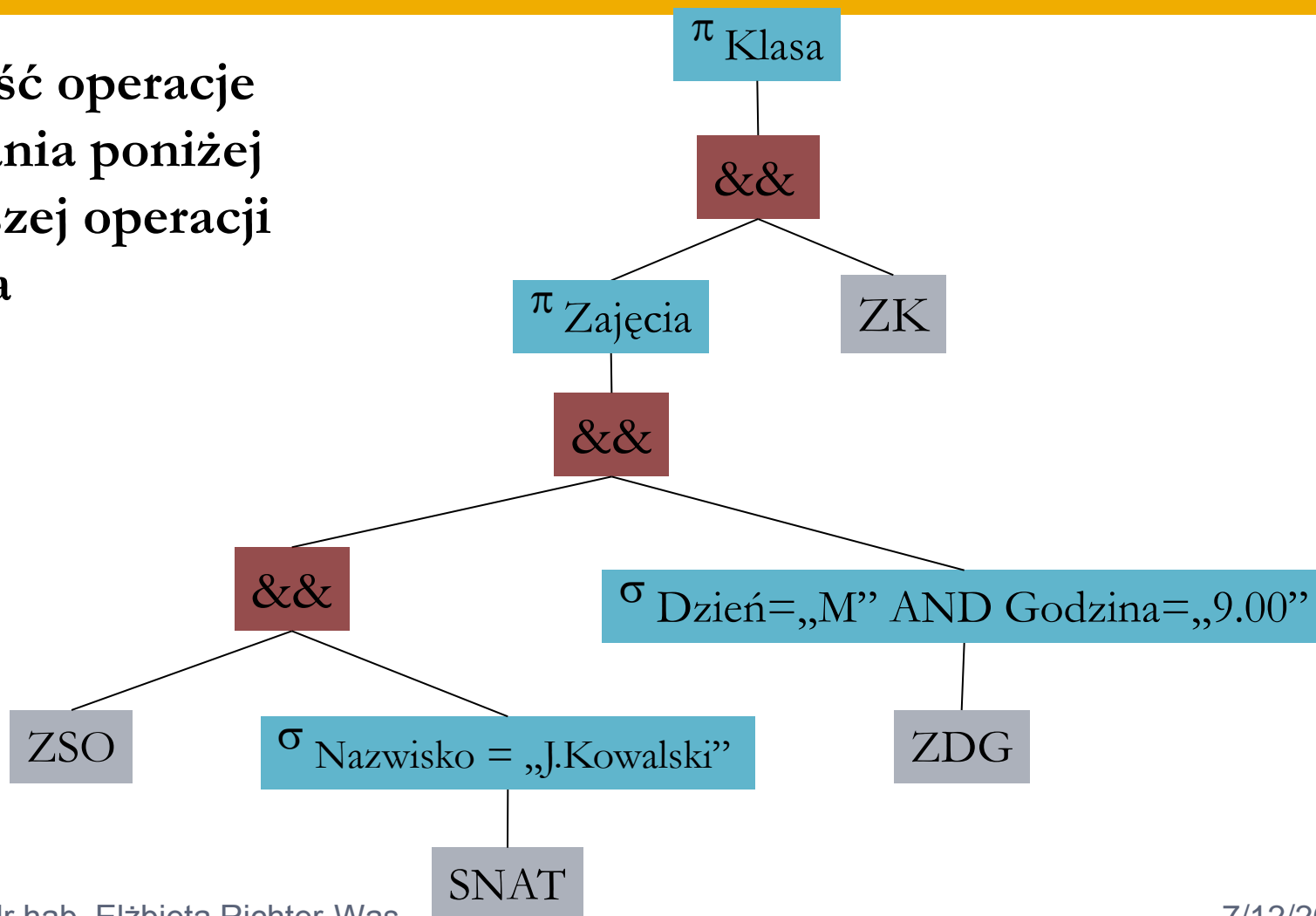
$$(\pi_L(R \underset{A=B}{\&\&} S)) = (\pi_L(\pi_M(R) \underset{A_i=B_j}{\&\&} \pi_N(S)))$$

- M – lista atrybutów z listy L, które należą do schematu relacji R, plus atrybut A, jeśli nie ma go na liście L
- N – lista atrybutów z listy L, które należą do schematu relacji R, plus atrybut B, jeśli nie ma go na liście L

Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?"

51

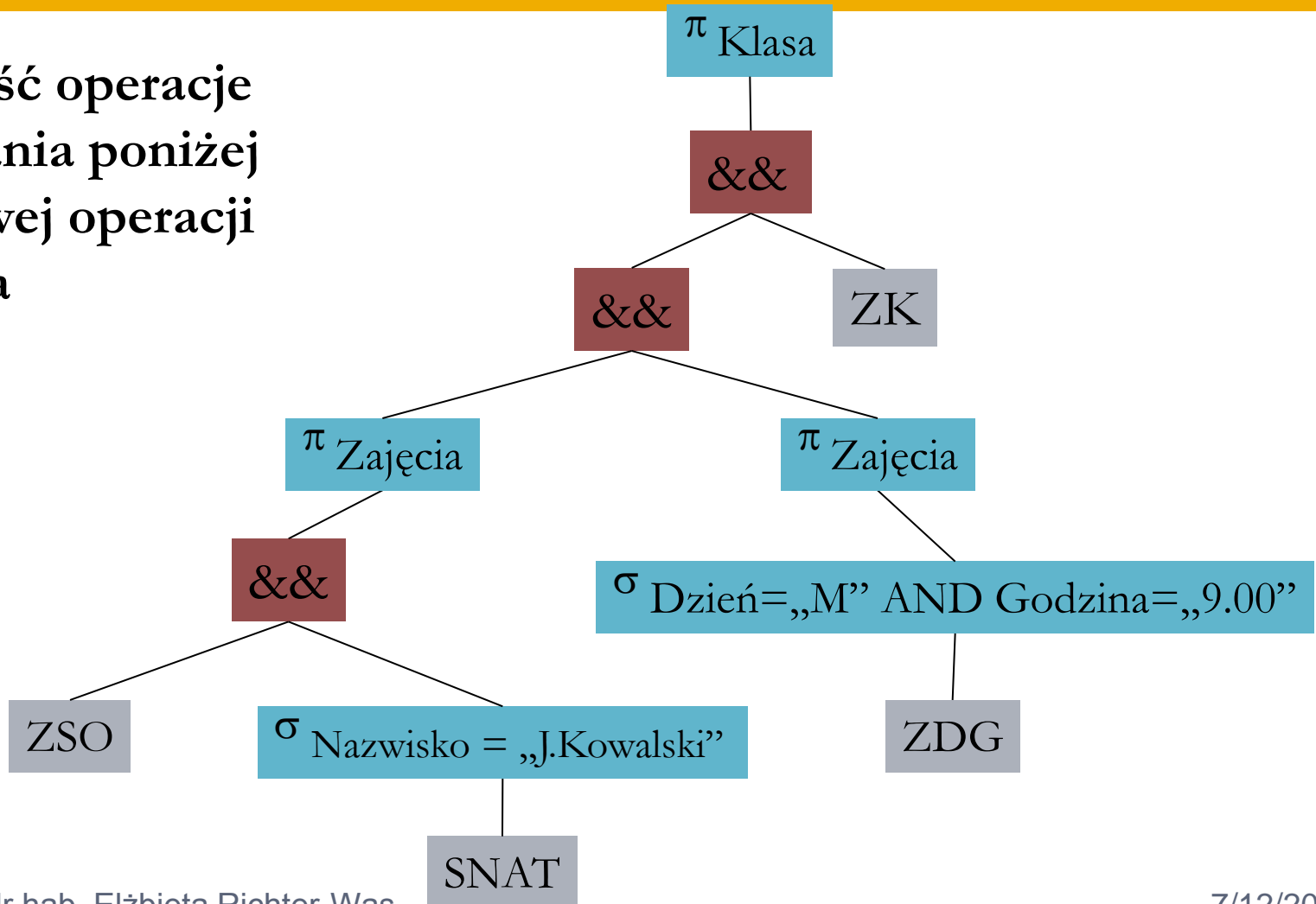
Przenieść operacje rzutowania poniżej najwyższej operacji łączenia



Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?"

52

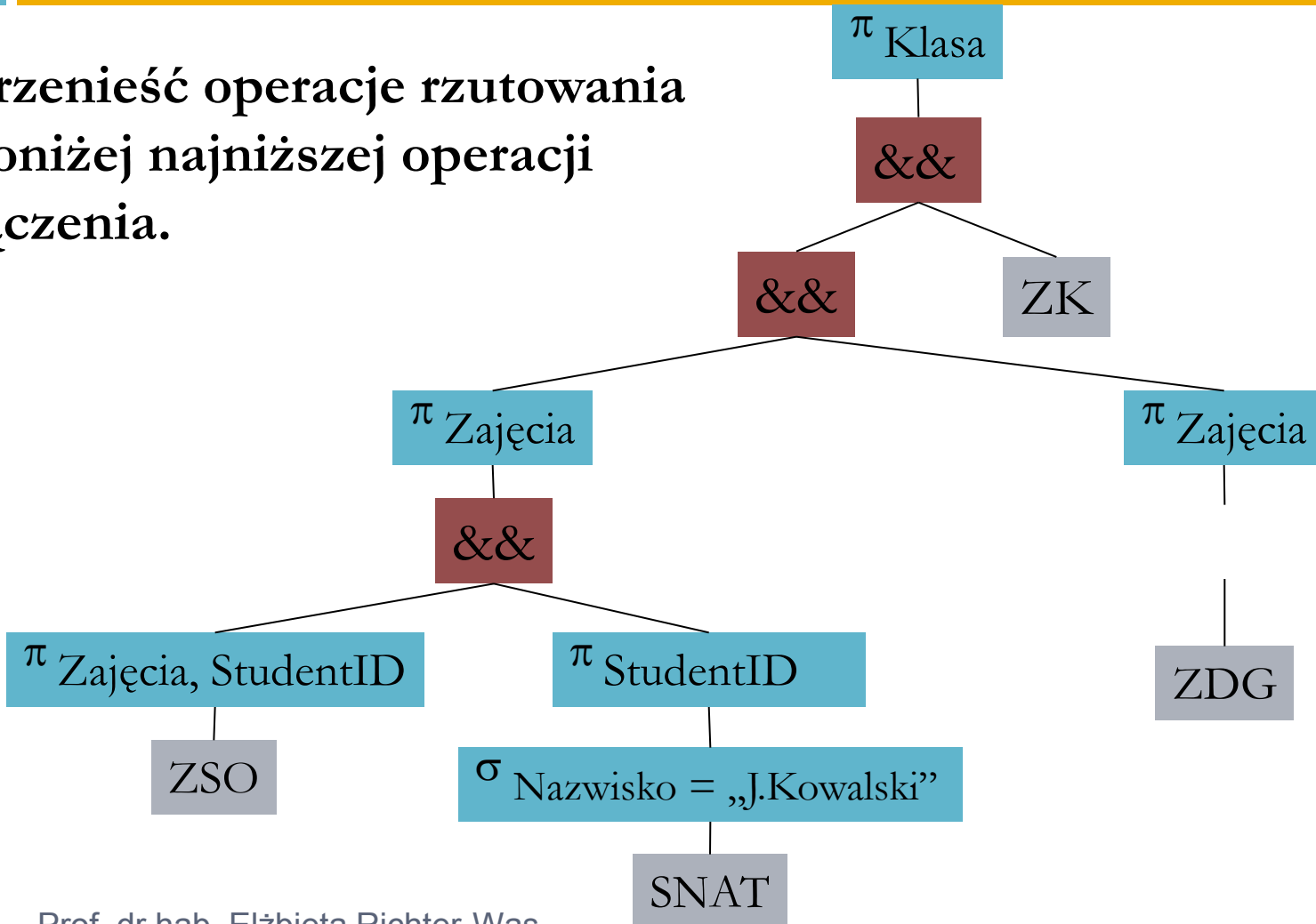
Przenieść operacje rzutowania poniżej środkowej operacji łączenia



Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?"

53

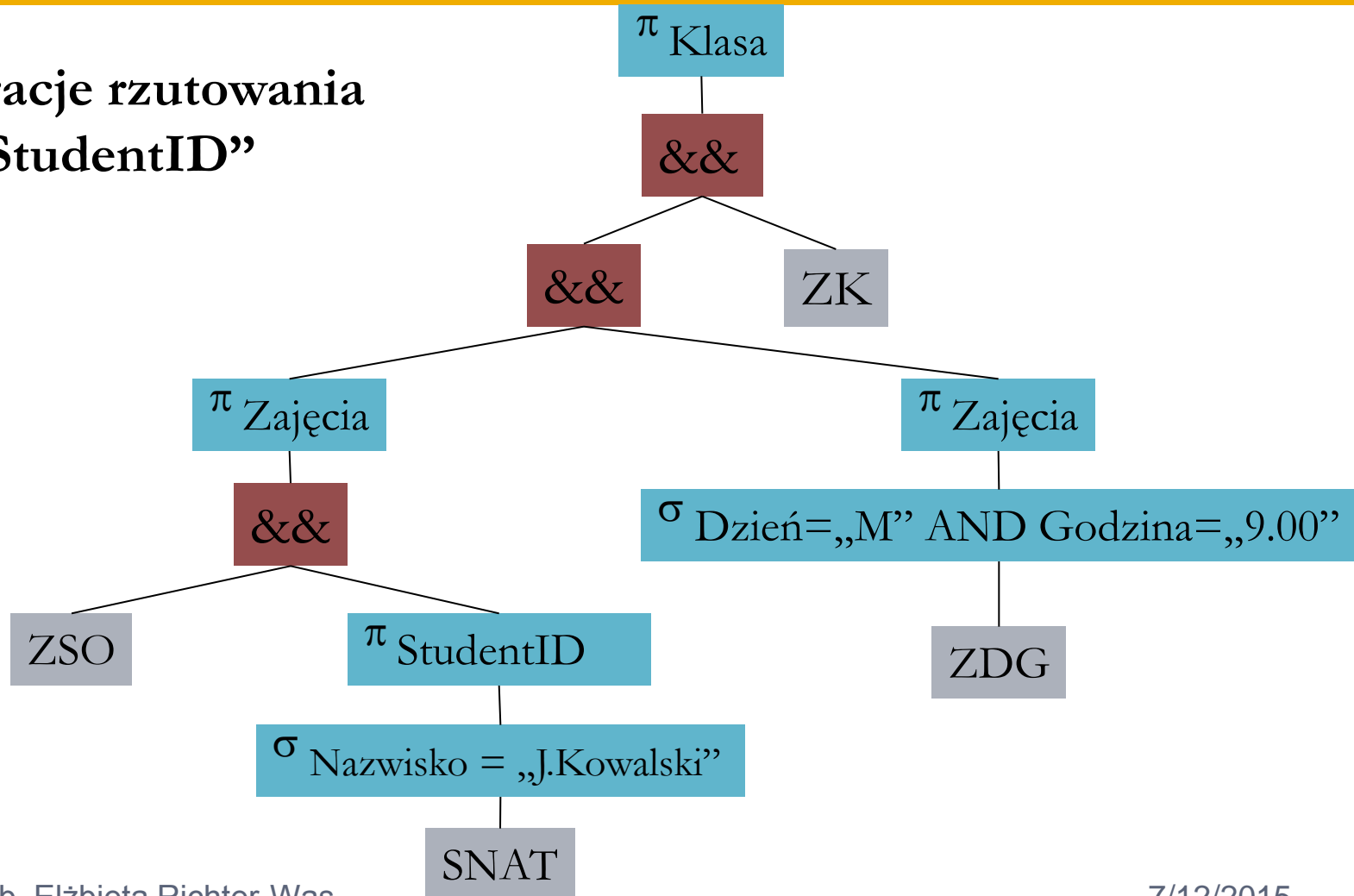
Przenieść operacje rzutowania poniżej najniższej operacji łączenia.



Gdzie przebywa J. Kowalski w poniedziałek o 9-tej rano?"

54

Usuń operacje rzutowania „Zajęcia, StudentID”



Podsumowanie

55

- Algebra relacyjna jest wysoko poziomową notacją definiowania operacji zapytań dotyczących jednej lub wielu relacji. Głównymi operacjami tej algebry są: **suma, przecięcie, różnica, selekcja, rzutowanie i złączenie**. Jest silną notacją wyrażania zapytań bez podawania szczegółów dotyczących planowanych operacji na otrzymanych danych.
- Istnieje wiele sposobów efektywnego implementowania operacji złączenia.
- Optymalizacja wyrażeń algebry relacyjnej może w znaczący sposób skrócić czas wyznaczania ich wartości, jest więc istotnym elementem wszystkich języków opartych na algebrze relacyjnej wykorzystywanych w praktyce do wyrażania zapytań.
- Istnieje **wiele sposobów skracania czasu** obliczania danego wyrażenia. Najlepsze efekty przynosi przenoszenie operacji selekcji w dół drzewa wyrażenia.