

Bazy danych



01:

Wprowadzenie,
modele danych

Krzysztof Stencel

Zaliczanie

- Laboratorium (klasówki i projekt)
 - 2 klasówki: SQL, projektowanie
 - 4 etapy pracy zaliczeniowej: projekt bazy danych, oprogramowanie serwera, aplikacja, strojenie
- Oceny
 - $\geq 90\%$ z klasówek + projekty oddane w terminie = 5
 - $\geq 75\%$ z klasówek + ≤ 3 tygodnie opóźnień łącznie = 4
 - $\geq 50\%$ z klasówek + projekty oddane tydzień przed egzaminem = 3
 - Inne = 2, niedopuszczenie na egzamin bez zaliczenia

Literatura

- J.Ulmann, J.Widom, *Podstawowy wykład z systemów baz danych*, WNT 2000. [jest już trzecie wydanie ang.]
- L.Banachowski, K.Stencel, *Bazy danych. Projektowanie aplikacji na serwerze*, EXIT, 2001. [PL/SQL]
- L.Banachowski, A.Chądzyńska, K.Matejewski, E. Mrówka, K.Stencel: *Bazy danych. Wykłady i ćwiczenia*, Wydawnictwo PJWSTK, Warszawa, 2003 [dużo zadań]
- L.Banachowski, E.Mrówka, K.Stencel: *Systemy baz danych. Wykłady i ćwiczenia*, Wydawnictwo PJWSTK, 2004 [dużo zadań]

Baza danych to...

Pamięć
zewnętrzna, koszt
transmisji danych

Modele
danych

zbiór danych o określonej strukturze,

zapisany na zewnętrznym nośniku informacji,

który może zaspokoić
potrzeby wielu użytkowników

korzystających z niego **SELECTywnie**

i w dogodnym dla siebie czasie.

Wielodostęp,
współbieżność

Język
zapytań

Awarie,
odtworzenie

Według Papcia Chmiela

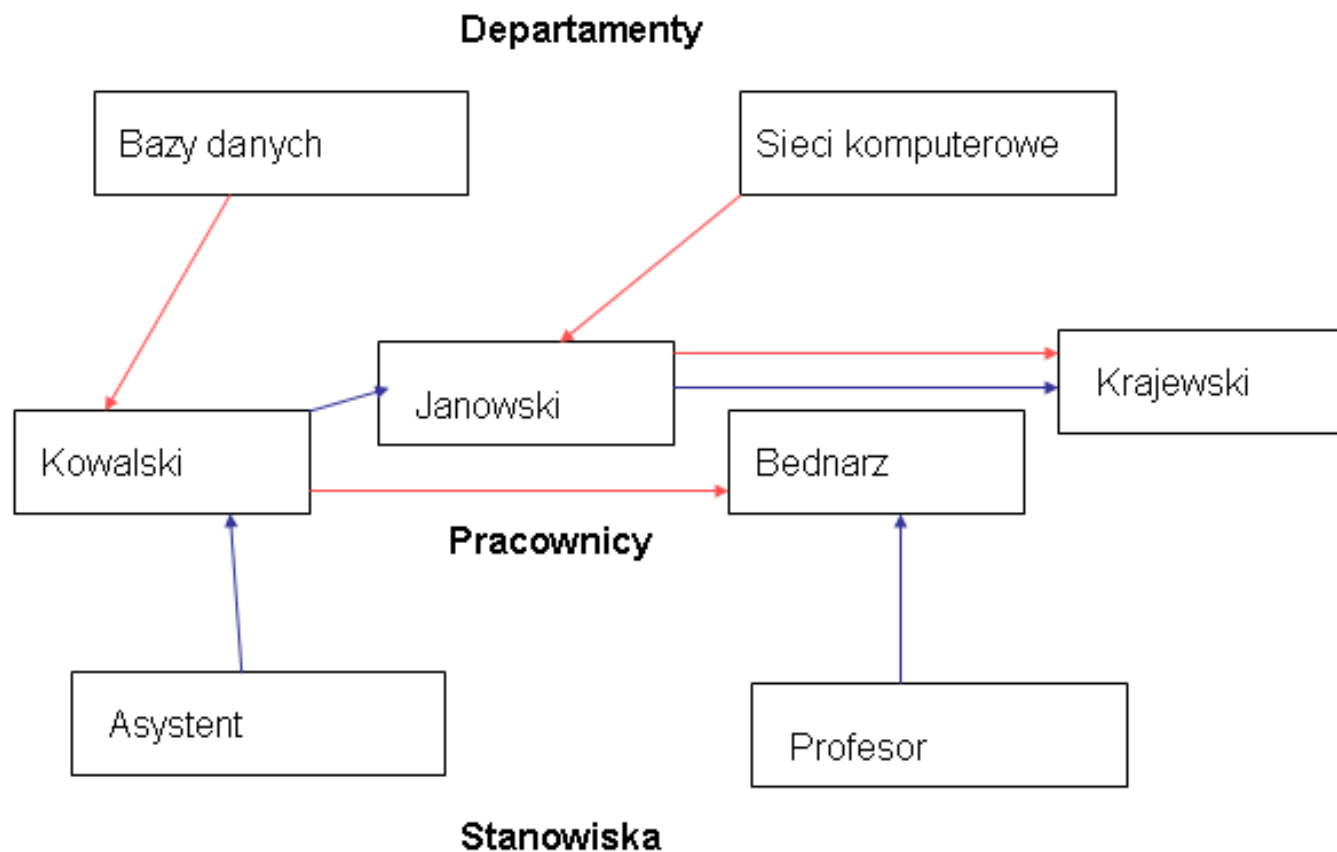


System zarządzania bazą danych (SZBD)

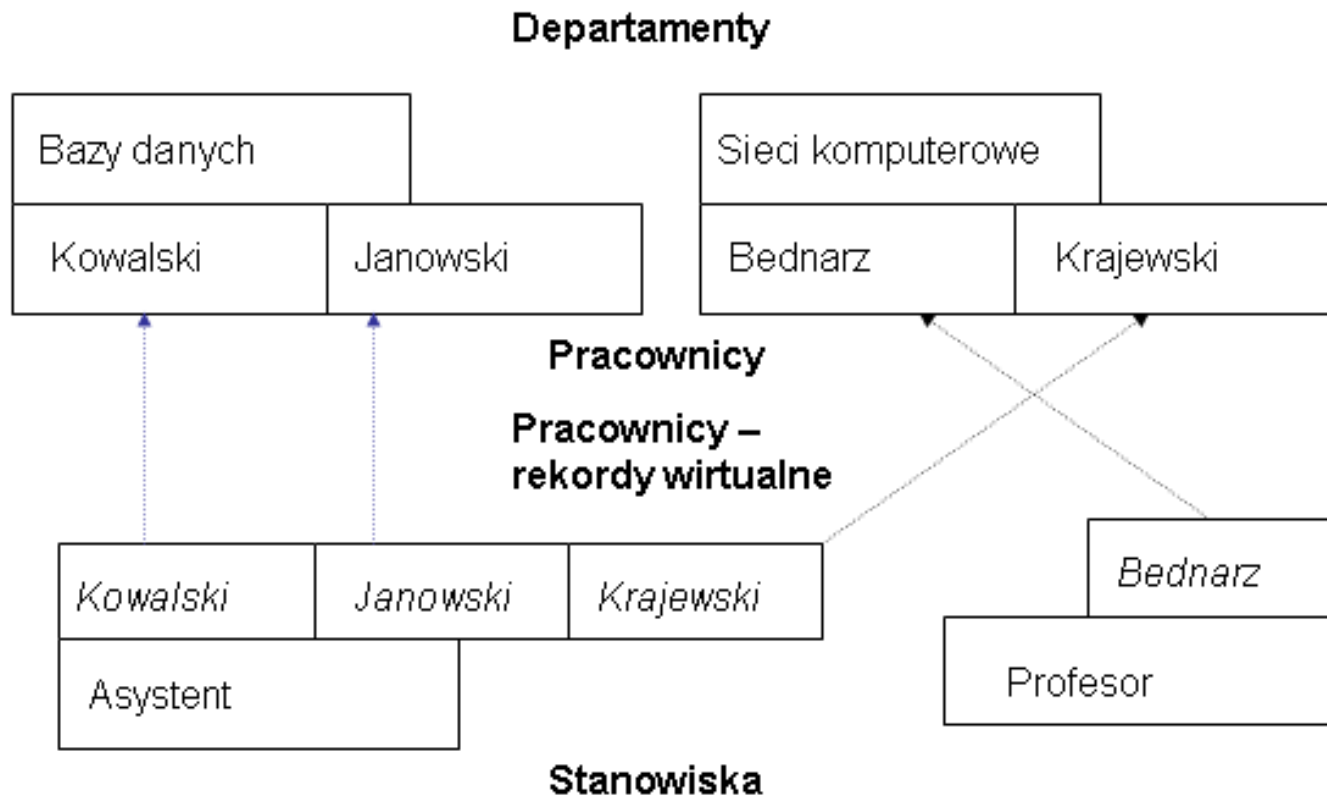
- Stały komplet oprogramowania zarządzający bazą danych.
- SZBD się nie pisze.
- SZBD się kupuje.
- Akurat my piszemy (LoXiM, JLoXiM, COHERENT).



Sieciowy model danych



Hierarchiczny model danych



Relacyjny model danych

Departamenty

10 Bazy danych
20 Sieci
komputerowe

.....

Pracownicy

10 Kowalski 1
20 Janowski 1
10 Bednarz 2
20 Krajewski 1

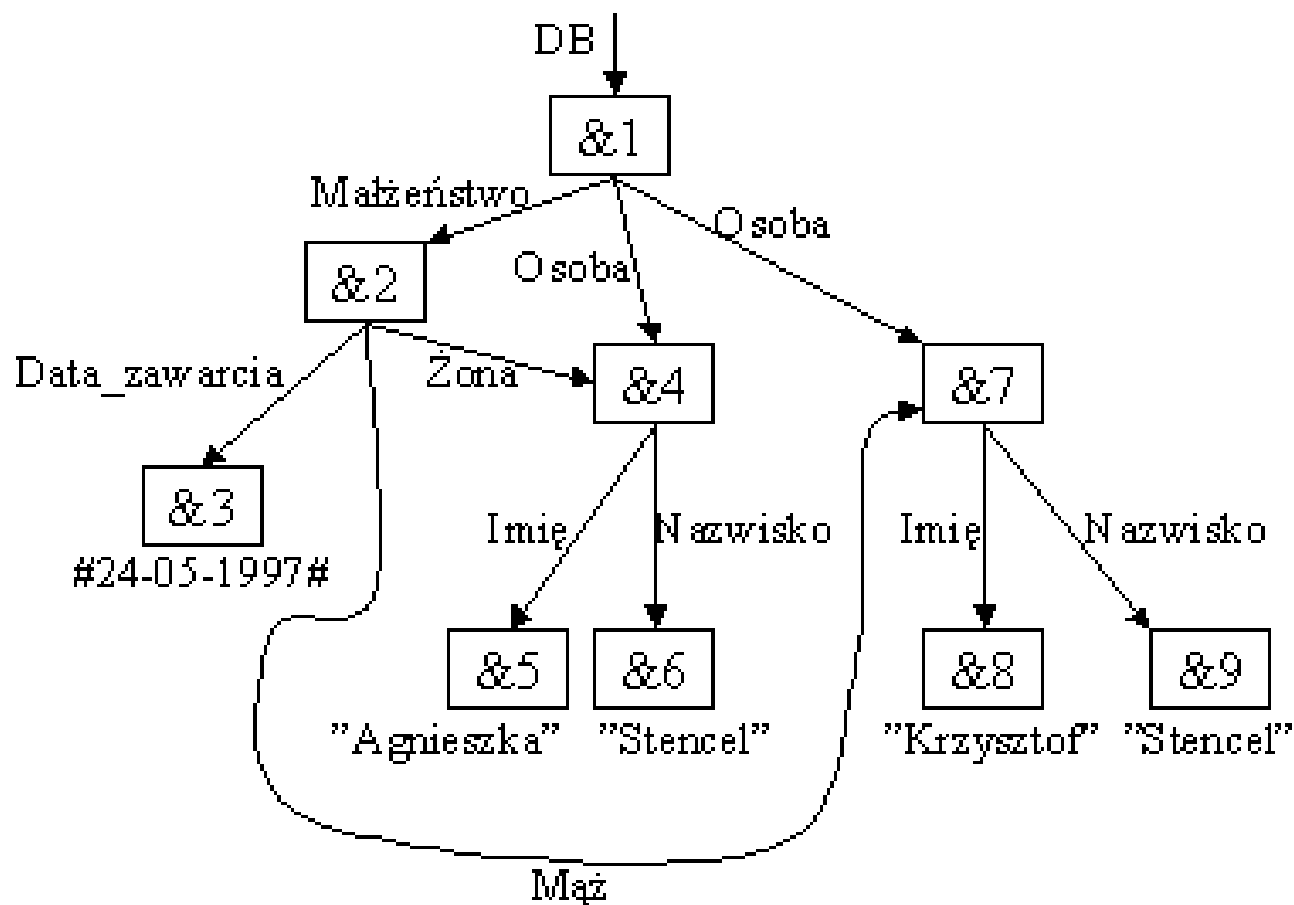
.....

Stanowiska

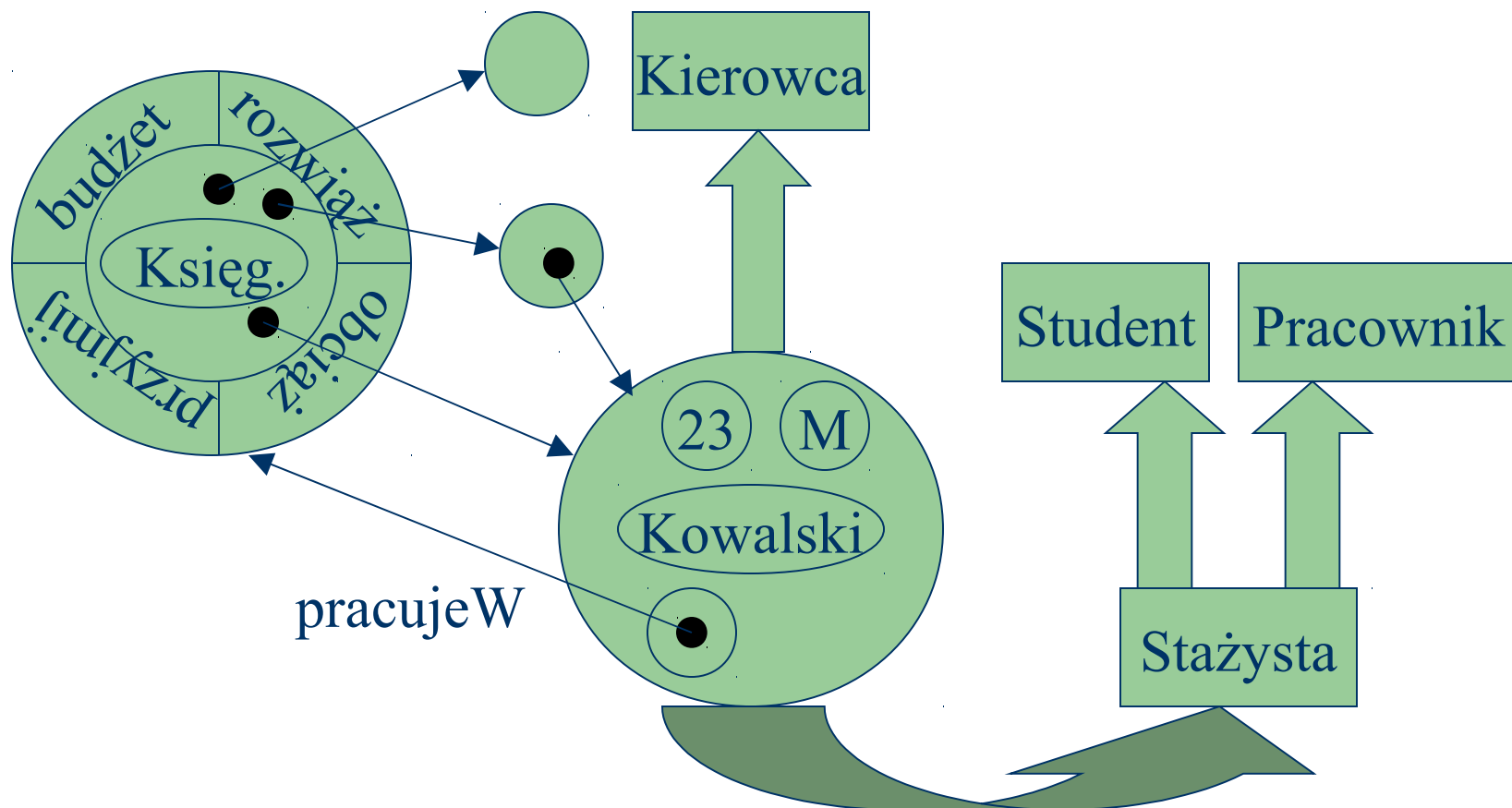
1 Asystent
2 Profesor

.....

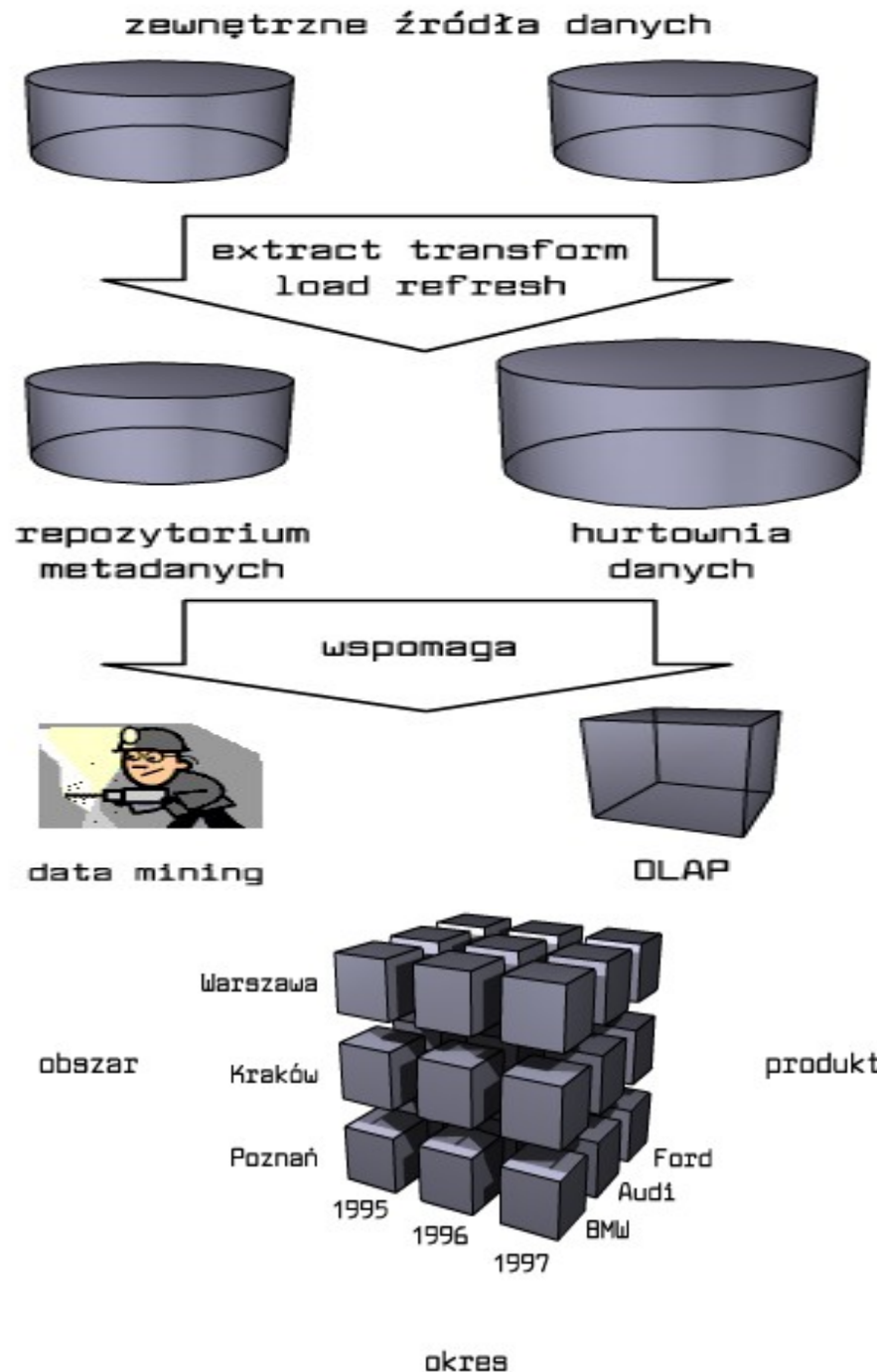
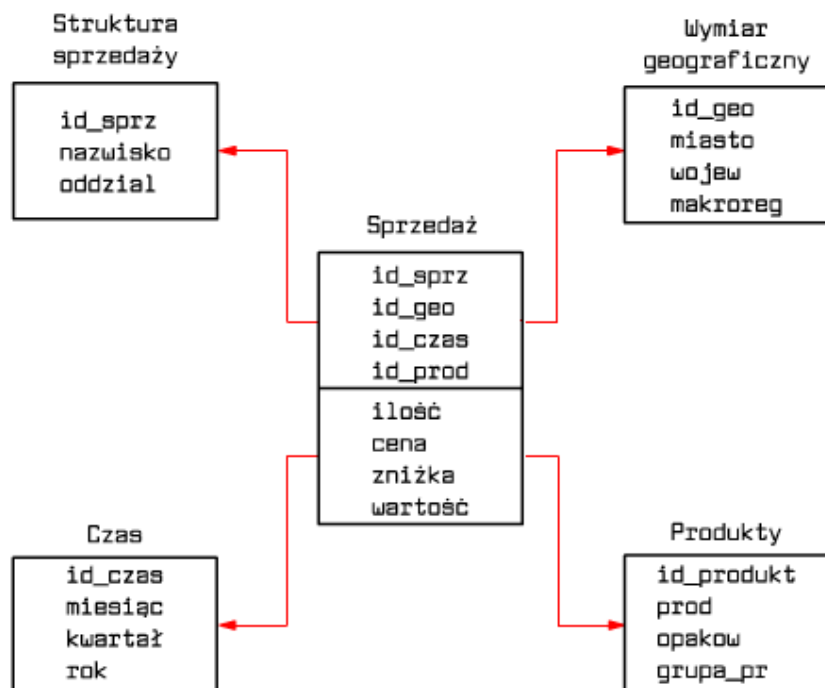
Półstrukturalny model danych



Obiektowy model danych



Hurtownie danych



Bazy danych



02:


Relacyjny model
danych, języki zapytań

Krzysztof Stencel

Pojęcia pierwotne

- **A** – zbiór nazw atrybutów
- **D** – zbiór wartości atomowych (napisy, liczby, daty, wartości logiczne)
- **E** – zbiór typów atomowych (integer, float, string, boolean, date)
- **T** – zbiór nazw relacji

Typy atomowe w SQL (Oracle)

- NUMBER, NUMBER(c), NUMBER(c,d)
- INTEGER
- CHAR(x)
- VARCHAR2(x)
- DATE (data + czas)
- TIMESTAMP (DATE + setne sekundy)
- 

Schemat relacji

$\text{kol} : \mathbf{T} \rightarrow P_{\text{fin}}(\mathbf{A})$ (nazwie relacji przyporządkowujemy skończone zbiory nazw kolumn)

$\text{dom} : \mathbf{T} \times \mathbf{A} \rightarrow \mathbf{E}$ (nazwie relacji i nazwie atrybutu przyporządkowujemy typ elementarny)

Przykład schematu relacji

$T = \{ \text{Osoba} \}$

$\text{kol}(\text{Osoba}) = \{ \text{PESEL}, \text{Imię}, \text{Nazwisko}, \text{Wiek} \}$

$\text{dom}(\text{Osoba}, \text{PESEL}) = \text{CHAR}(11)$

$\text{dom}(\text{Osoba}, \text{Imię}) = \text{VARCHAR2}(20)$

$\text{dom}(\text{Osoba}, \text{Nazwisko}) = \text{VARCHAR2}(25)$

$\text{dom}(\text{Osoba}, \text{Wiek}) = \text{NUMBER}(3,0)$

Schemat relacyjnej bazy danych

$$SCH_{rel} = \langle \mathbf{T}, kol, dom \rangle$$

T – skończony zbiór nazw tabel

kol – przyporządkowanie tabelom nazw kolumn

dom – przyporządkowanie kolumnom typów (***dziedzin***)

Krotka relacji

Jeśli $R \in \mathbf{T}$, to krotka t o schemacie R jest funkcją przypisującą wartości nazwom kolumn:

$$t : \text{kol}(R) \rightarrow \mathbf{D}$$

$$\text{dla } A \in \text{kol}(R), t(A) \in \mathbf{D}_{\text{dom}(R,A)}$$

Przykład krotki

$\text{kol}(\text{Osoba}) = \{ \text{PESEL}, \text{Imię}, \text{Nazwisko}, \text{Wiek} \}$

$t(\text{PESEL}) = '91081256453'$

$t(\text{Imię}) = 'Alfred'$

$t(\text{Nazwisko}) = 'Miśkiewicz'$

$t(\text{Wiek}) = 42$

t jest krotką o schemacie Osoba.

Egzemplarz relacji i egzemplarz bazy danych

- Jeśli $R \in T$, to relacja o nazwie R jest skończonym zbiorem krotek o schemacie R .
- Egzemplarz relacyjnej bazy danych, to przyporządkowanie egzemplarza relacji każdej nazwie relacji zdefiniowanej w schemacie.

Relacja jako tabela

PESEL	Imię	Nazwisko	Wiek
91081256453	Alfred	Miśkiewicz	42
99060134534	Julia	Miśkiewicz	10
05222733453	Klaudia	Miśkiewicz	8

Relacja jako relacja

```
{  
  < '91081256453', 'Alfred', 'Miśkiewicz', 42 > ,  
  < '99060134534', 'Julia', 'Miśkiewicz', 10 > ,  
  < '05222733453', 'Klaudia', 'Miśkiewicz', 8 >  
}
```

Języki zapytań

- Praktyczne: SQL, XQuery
- Teoretyczne: algebra relacji (ALG), rachunek pierwszego rzędu (FOL)
- Eksperymentalne: OQL, Lorel, UnQL, SBQL...
- Zajmiemy się pierwszymi trzema
- Systematyczny kurs SQL na labach

SQL = Structured Query Language

- Standard przemysłowy, choć nie zapewnia przenośności.
- Trzy warstwy: podstawowa (SQL-87), pośrednia (SQL-92) i pełna (SQL-99).
- Większość implementacji zgodna z SQL na poziomie podstawowym.

SQL

- = DML (Data Manipulation Language)
- + DDL (Data Definition Language)
- + DCL (Data Control Language)
- + SQL/CLI (API)
- + SQL/MM (multi-media)
- + SQL/Transaction
- + SQL/MED + SQL/JRT + SQL/OLB + ...

Rachunek pierwszego rzędu (FOL)

- Zapytania mają postać:

$$\{ \langle x_{i_1}, x_{i_2}, \dots, x_{i_n} \rangle : \Phi(x_1, x_2, \dots, x_m) \}$$

- Φ to formuła pierwszego rzędu o zmiennych wolnych x_1, x_2, \dots, x_m
- Szukamy wszystkich wartościowań zmiennych x_1, x_2, \dots, x_m spełniających Φ i dla każdego takiego wartościowania generujemy krotkę $\langle x_{i_1}, x_{i_2}, \dots, x_{i_n} \rangle$

Algebra relacji (ALG)

- Nośnik: wszystkie relacje dla zadanego zbioru atrybutów i dziedzin

- Operacje:

Selekcja σ Suma \cup

Rzut π Różnica $-$

Zmiana nazw ρ Przecięcie \cap

Złączenie \bowtie Produkt \times

Selekcja (wybór wierszy)

- φ – formuła z nazwami kolumn jako zmiennymi

$$\sigma_{\varphi}(r) = \{ t \in r : \varphi(t) \}$$

$$\text{kol}(\sigma_{\varphi}(r)) = \text{kol}(r)$$

$$\{ \langle x_1, x_2, \dots, x_m \rangle : \varphi(x_1, x_2, \dots, x_m) \wedge r(x_1, x_2, \dots, x_m) \}$$

SELECT * FROM r WHERE φ ;

Rzut (wybór kolumn)

- $X \subseteq \text{kol}(r)$ – pewien podzbiór kolumn

$$\pi_X(r) = \{ t \upharpoonright X : t \in r \}$$

$$\text{kol}(\pi_X(r)) = X$$

$$\{ \langle x_{i_1}, x_{i_2}, \dots, x_{i_n} \rangle : r(x_1, x_2, \dots, x_m) \}$$

SELECT **DISTINCT** X FROM r;

Zmiana nazw (kolumn)

- $p : \text{kol}(r) \rightarrow A$ – funkcja różnowartościowa

$$\rho_p(r) = \{ u : \exists t \in r \ \forall a \in \text{kol}(r) \ u(p(a)) = t(a) \}$$

$$\text{kol}(\rho_p(r)) = \{ p(a) : a \in \text{kol}(r) \} \quad // \text{ obraz } p$$

$$\{ \langle x_{p(1)}, x_{p(2)}, \dots, x_{p(m)} \rangle : r(x_1, x_2, \dots, x_m) \}$$

SELECT a_1 AS $p(a_1), \dots, a_m$ AS $p(a_m)$ FROM r ;

Suma, różnica, przecięcie

- $\text{kol}(r) = \text{kol}(s)$. Wówczas $\text{kol}(r \cup s) = \text{kol}(r) = \text{kol}(s)$.

$$r \cup s = \{ t : t \in r \vee t \in s \} \quad r \cap s = \{ t : t \in r \wedge t \in s \}$$

$$r - s = \{ t : t \in r \wedge t \notin s \}$$

$$\{ \langle x_1, x_2, \dots, x_m \rangle : r(x_1, x_2, \dots, x_m) \vee s(x_1, x_2, \dots, x_m) \}$$

SELECT * FROM r

UNION EXCEPT INTERSECT

SELECT * FROM s;

Produkt kartezjański

$$r \times s = \{ t : t \upharpoonright \text{kol}(r) \in r \wedge t \upharpoonright \text{kol}(s) \in s \}$$

$$\text{kol}(r \times s) = \text{kol}(r) \uplus \text{kol}(s) \quad // \text{ suma rozłączna}$$

$$\{ \langle x_1, \dots, x_m, y_1, \dots, y_k \rangle : r(x_1, \dots, x_m) \wedge s(y_1, \dots, y_k) \}$$

SELECT * FROM r,s; // SQL87

SELECT * FROM r CROSS JOIN s; // SQL92

Złączenie naturalne

$$r \bowtie s = \{ t : t \upharpoonright \text{kol}(r) \in r \wedge t \upharpoonright \text{kol}(s) \in s \}$$

$$\text{kol}(r \bowtie s) = \text{kol}(r) \cup \text{kol}(s) \quad // \text{ suma zwykła}$$

$$\{ \langle x_1, \dots, x_m, z_1, \dots, z_i, y_1, \dots, y_k \rangle : r(x_1, \dots, x_m, z_1, \dots, z_i) \wedge \\ s(y_1, \dots, y_k, z_1, \dots, z_i) \}$$

Zakładamy, że $\text{kol}(r) \cap \text{kol}(s)$ zawiera i kolumn.

Złączenie naturalne (SQL)

SQL87:

```
SELECT  $x_1, \dots, x_m, z_1, \dots, z_i, y_1, \dots, y_k$   
FROM r,s  
WHERE  $r.z_1 = s.z_1$  AND ... AND  $r.z_i = s.z_i$ 
```

SQL92:

```
SELECT * FROM r NATURAL JOIN s;  
SELECT * FROM r JOIN s ON  
    ( $r.z_1 = s.z_1$  AND ... AND  $r.z_i = s.z_i$ );  
SELECT * FROM r NATURAL JOIN s  
    USING ( $z_1, \dots, z_i$ );
```

Złączenie

- Złożenie produktu kartezjańskiego z selekcją

$$r \bowtie_{\varphi} s = \sigma_{\varphi}(r \times s)$$

SELECT * FROM r,s WHERE φ ; // SQL87

SELECT * FROM r JOIN s ON φ ; // SQL92

SELECT * FROM r INNER JOIN s ON φ ; // SQL92

Domknięcie

- ALG – tak, każda operacja daje relację skończoną
- FOL – nie, wynik może być nieskończony:
$$\{ \langle x, y \rangle : r(x) \vee s(y) \}$$
- SQL – tak, ale uwaga na wielozbiory i DISTINCT.

Niezależność dziedzinowa i bezpieczeństwo FOL

- *Dziedzina aktywna* = stałe + wartości z relacji
- Zapytanie jest niezależne dziedzinowo (n. dz., *domain independent*), gdy dodawanie do dziedziny elementów nie zmienia jego wyniku [Wynik zależy tylko od dziedziny aktywnej]
- Zapytanie jest bezpieczne (*safe*), gdy spełnia pewne warunki składniowe (będą dalej)
- Nie znamy właściwości zapytań n. dz.
- Umiemy sprowadzić każde zapytanie n. dz. do bezpiecznego

Niezależność dziedzinowa a spełnialność

- Problem decyzyjny, czy formuła I rzędu jest spełnialna, jest nierozstrzygalny
- Jeśli umielibyśmy decydować, czy formuła jest niezależna dziedzinowo, to też umielibyśmy decydować, czy formuła I rzędu jest spełnialna

$$(r(x) \vee s(y)) \wedge \Phi$$

jest bezpieczna wtedy i tylko wtedy, gdy Φ jest niespełnialna

- Nie może więc istnieć skończony algorytm decydujący, czy formuła jest niezależna dziedzinowo

Bezpieczny FOL

- Zmienne: nieograniczone i ograniczone (mające wartości z dziedziny aktywnej)
- Definicja rekurencyjna ze względu na budowę:
 - $r(x_1, x_2, \dots, x_n)$ wszystkie ograniczone
 - $x = a$, x jest ograniczona (a to stała lub zmienna ograniczona)
 - $\neg\Phi$ wszystkie nieograniczone
 - $\Phi_1 \wedge \Phi_2$ ograniczone są te, które są ograniczone w Φ_1 **lub** w Φ_2
 - $\Phi_1 \vee \Phi_2$ ograniczone są te, które są ograniczone w Φ_1 **i** w Φ_2
- Wszystkie zmienne w formule muszą być ograniczone żeby była bezpieczna

Bezpieczeństwo a niezależność dziedzinowa

- Niech Φ będzie niezależna dziedzinowo
- Każdą zmienną można więc bez zmiany semantyki ograniczyć do dziedziny aktywnej. Dodajemy więc czynnik ograniczający dla każdej zmiennej x (a_1, a_2, \dots, a_n to stałe; r_1, r_2, \dots, r_m to symbole relacyjne):

$$\Phi \wedge (x=a_1 \vee x=a_2 \vee \dots \vee x=a_n$$

$$\vee \exists y_1, y_2, \dots, y_k (r_1(y_1, y_2, \dots, y_k) \wedge x=y_1)$$

$$\vee \exists y_1, y_2, \dots, y_k (r_1(y_1, y_2, \dots, y_k) \wedge x=y_2)$$

$$\vee \dots \quad [\text{dla wszystkich relacji i argumentów}]$$

Prawie równe moce (prawie robi różnicę)

- sFOL = bezpieczne zapytania FOL
- nFOL = niezależne dziedzinowo zapytania FOL
- sFOL = nFOL = ALG
- ALG = SELECT-FROM-WHERE
 - bo nie cały SQL!
 - gdy wszystko otoczyć SELECT DISTINCT * FROM
- FOL prawie = ALG (czyli $FOL \neq ALG$)
 - Bo zapytania zależne dziedzinowo

Bazy danych



03:

Więcej o językach
zapytań

Krzysztof Stencel

Złączenie wewnętrzne

- Takie normalne jak dotychczas
- Znikają dane nie mające pary

$$\pi_{\text{kol}(r)}(r \bowtie s) \subseteq r$$

$$\pi_{\text{kol}(s)}(r \bowtie s) \subseteq s$$

- Przy integracji i normalizacji baz danych przydaje się złączenie nie gubiące danych, tj. mające równości w powyższych formułach

Złączenie zewnętrzne

Osoby	
Imię	Płeć
Anna	K
Bolek	M

Psy	
Imię	Pies
Anna	As
Mirek	Fafik

wewnętrzne

lewostronne

prawostronne

Osoby ⋈ Psy			
O.Imię	Płeć	P.Imię	Pies
Anna	K	Anna	As
Bolek	M	NULL	NULL
NULL	NULL	Mirek	Fafik

Złączenie z agregacją (czy aby OK?)

- Ile psów ma każda osoba?
SELECT Osoby.Imię, COUNT(*) AS IlePsów
FROM Osoby, Psy
WHERE Osoby.Imię = Psy.Imię
GROUP BY Osoby.Imię;

Osoby	
Imię	Płeć
Anna	K
Bolek	M

Imię	IlePsów
Anna	1

Oracle (drzewiej i teraz)

```
SELECT O.Imię, O.Płeć, P.Imię, P.Pies  
FROM Osoby O, Psy P  
WHERE O.Imię = P.Imię (+);
```

- Tylko złączenia jednostronne
- Dość kłopotliwa składnia

SQL-92 (teraz też Oracle)

```
SELECT O.Imię, O.Płeć, P.Imię, P.Pies  
FROM Osoby O FULL OUTER JOIN Psy P  
ON O.Imię = P.Imię;
```

- FULL opcjonalne
- Zamiast FULL może być LEFT lub RIGHT
- Jeśli jest FULL/LEFT/RIGHT nie musi być OUTER
- Niekompletne układanki dla nerwowo chorych

Złączenie z agregacją (teraz OK?)

- Ile psów ma każda osoba?

```
SELECT Osoby.Imię, COUNT(*) AS IlePsów  
FROM Osoby LEFT JOIN Psy ON  
      (Osoby.Imię = Psy.Imię)  
GROUP BY Osoby.Imię;
```

Osoby	
Imię	Płeć
Anna	K
Bolek	M

Imię	IlePsów
Anna	1
Bolek	1

Złączenie z agregacją (teraz OK!)

- Ile psów ma każda osoba?

```
SELECT Osoby.Imię, COUNT(Pies) AS IlePsów  
FROM Osoby LEFT JOIN Psy ON  
      (Osoby.Imię = Psy.Imię)  
GROUP BY Osoby.Imię;
```

Osoby	
Imię	Płeć
Anna	K
Bolek	M

Imię	IlePsów
Anna	1
Bolek	0

Kuriozum SQL-92 – UNION JOIN

$r \text{ UNION JOIN } s \equiv r \text{ FULL JOIN } s \text{ ON FALSE}$

kol(r)	kol(s)
r	NULL
NULL	s

Pseudo-wartość NULL

- Oznacza brak wartości
- Nie jest zerem, napisem pustym, etc.
- Ma rozmaite interpretacje
 - Niesie informacje o braku powiązania (np. Emp.Mgr gdy ma wartość NULL oznacza, że nie ma szefa)
 - Oznacza, że informacja nie jest znana (np. nie znamy NIPu jakiegoś studenta)
 - Oznacza nie stosowalność danej (np. kategoria wojskowa dziewczyn)

NULL – rozmaite interpretacje

- Porównanie z NULL (nie-wiadomo-co) daje trzecią wartość logiczną (UNKNOWN = U)
- NULL jest ignorowany przez agregacje (tak jakby go nie było), np. $\text{AVG}(1, 2, \text{NULL}) = 1.5$
- GROUP BY traktuje wszystkie NULL jako równe (wrzuca je do jednej grupy)
- WHERE traktuje wartość logiczną U jako F.

Tabelki dla operatorów logiki trójwartościowej

AND	F	U	T
F	F	F	F
U	F	U	U
T	F	U	T

OR	F	U	T
F	F	U	T
U	U	U	T
T	T	T	T

NOT	
F	T
U	U
T	F

Przykładowy schemat danych

```
SQL> describe EMP
```

Name	Type
------	------

EMPNO	NUMBER(4)
ENAME	VARCHAR2(10)
JOB	VARCHAR2(9)
MGR	NUMBER(4)
HIREDATE	DATE
SAL	NUMBER(7,2)
COMM	NUMBER(7,2)
DEPTNO	NUMBER(2)

```
SQL> describe DEPT
```

Name	Type
------	------

DEPTNO	NUMBER(2)
DNAME	VARCHAR2(14)
LOC	VARCHAR2(13)

Najlepiej zarabiający pracownik (1)

Emp	
Ename	Sal
דוד	30
שרה	40
נתן	NULL

Ename
שרה

```
SELECT Ename FROM Emp
WHERE Sal =
      (SELECT MAX(Sal)
       FROM Emp);
```

- MAX ignoruje NULL
- Lista pracowników, którzy zapewne zarabiają najwięcej

Najlepiej zarabiający pracownik (2)

Emp	
Ename	Sal
דוד	30
שרה	40
נתן	NULL

Ename
(pusto)

```
SELECT Ename FROM Emp
WHERE Sal >= ALL
      (SELECT Sal
       FROM Emp);
```

- Porównania z NULL
- Lista pracowników, którzy na pewno zarabiają najwięcej

Najlepiej zarabiający pracownik (3)

Emp	
Ename	Sal
דוד	30
שרה	40
נתן	NULL

Ename
שרה
נתן

```
SELECT Ename FROM Emp E
WHERE NOT EXISTS
  (SELECT 'x' FROM Emp I
   WHERE I.Sal > E.Sal)
```

- Wewnętrzny WHERE traktuje U jako F, a potem jest NOT
- Lista pracowników, którzy mogą zarabiać najwięcej

Sprawdzenie, czy NULL

- Zawsze TRUE lub FALSE
 - Sal IS NULL
 - Sal IS NOT NULL
- Możliwość programowania defensywnego
- Sprowadzenie zapytania (2) do (1):

```
SELECT Ename FROM Emp
WHERE Sal >= ALL (SELECT Sal
                  FROM Emp
                  WHERE Sal IS NOT NULL);
```

Konsekwencje właściwości NULL

- Te zapytania nie są równoważne:
SELECT SUM(Sal)+SUM(Comm) FROM Emp;
SELECT SUM(Sal+Comm) FROM Emp;
- Funkcja NVL pozwala obronić się przed NULL
 - $NVL(x, y) = x$, gdy x IS NOT NULL
 - $NVL(x, y) = y$, gdy x IS NULL
- Lepiej uważać na NULL i agregacje, np.
SELECT SUM(Sal+NVL(Comm,0)) FROM Emp;

GROUP BY – *nest*

Osoby		
Imię	Płeć	Pies
Anna	K	As
Anna	K	Bara
Bolek	M	Florek

nest _{Imię} (Osoby)		
Imię	GROUP	
Anna	Płeć	Pies
	K	As
	K	Bara
Bolek	Płeć	Pies
	M	Florek

Operator *nest*

- GROUP BY wymaga przejścia do modelu zagnieżdżonych relacji
- Powstaje kolumna, która jako wartości ma zbiory wierszy

$X \subseteq \text{kol}(r)$ – pewien podzbiór kolumn

$\text{nest}_X(r) = \{ t : t \upharpoonright X \in r \wedge t(\text{GROUP}) =$

$\{ u \upharpoonright (\text{kol}(r) - X) : u \in r \wedge u \upharpoonright X = t \upharpoonright X \} \}$

$\text{kol}(\text{nest}_X(r)) = X \cup \{ \text{GROUP} \}$

GROUP BY – nadużycie semantyczne

- Na tak przetworzonej relacji oblicza się GROUP BY
- Wynik musi być jednak relacją nie zagnieżdżoną
- Właśnie dlatego wynik z obliczeń na kolumnach nie grupujących musi być agregowany

... NOT A GROUP BY EXPRESSION

Samo-złączenie

```
SELECT E.Ename, M.Ename  
FROM Emp E, Emp M  
WHERE E.Mgr = M.Empno;
```

- W ALG do zapisania tego zwykle przydaje się operator zmiany nazw kolumn ρ

Zapytania trudniejsze

- Poszukiwanie ścieżek w grafie
- Poszukiwanie domknięcia przechodniego relacji
- Da się zrobić w SQL-92, o ile znamy maksymalną długość ścieżki lub maksymalną długość „przechodniości” n
- Wymaga n -krotnego samo-złączenia

CONNECT BY (Oracle)

```
SELECT Ename, Empno, Mgr, LEVEL  
FROM Emp  
START WITH Mgr IS NULL  
CONNECT BY PRIOR Empno = Mgr;
```

- Budowa lasu drzew
 - **Korzenie**: wiersze spełniające klauzulę START WITH
 - **Wierzchołki**: wiersze tabeli z klauzuli FROM
 - **Krawędzie**: pary spełniające klauzulę CONNECT BY
- Stop gwarantowany!

Ewaluacja zapytania CONNECT BY

ENAME	EMPNO	MGR	LEVEL
KING	7839	NULL	1
JONES	7566	7839	2
SCOTT	7788	7566	3
ADAMS	7876	7788	4
FORD	7902	7566	3
SMITH	7369	7902	4
BLAKE	7698	7839	2
ALLEN	7499	7698	3
WARD	7521	7698	3
MARTIN	7654	7698	3
TURNER	7844	7698	3
JAMES	7900	7698	3
CLARK	7782	7839	2
MILLER	7934	7782	3

CONNECT BY – właściwości

- Gdy w danych jest cykl – zgłasza wyjątek
ORA-01436: CONNECT BY loop in user data
- W 10g można uodpornić się na cykle poprzez
CONNECT BY **NOCYCLE** ...
- Inne pseudo-kolumny
 - CONNECT_BY_ISLEAF
 - CONNECT_BY_ISCYCLE
- Modyfikator CONNECT_BY_ROOT (~PRIOR)

SQL-99: WITH RECURSIVE

- Jak LET REC z programowania funkcyjnego
- Definiujemy rekurencyjną perspektywę, której potem można użyć
- Tylko UNION ALL (koszt usuwania duplikatów)
- Tylko rekurencja liniowa
- Może się nie zatrzymać

SQL-99: WITH RECURSIVE

```
WITH RECURSIVE EmpHier (Ename, Empno, Mgr,  
    Level)
```

```
AS (
```

```
    SELECT B.Ename, B.Empno, NULL, 1  
    FROM EMP B WHERE B.Mgr IS NULL
```

```
    UNION ALL
```

```
    SELECT E.Ename, E.Empno, E.Mgr, H.level + 1  
    FROM EMP E, EmpHier H  
    WHERE H.Empno = E.Mgr
```

```
)
```

```
SELECT * FROM EmpHier;
```

Łatwo zapętlić

```
WITH RECURSIVE SmartLoop (Counter)
AS (SELECT 1 FROM EMP
    UNION ALL
    SELECT Counter + 1 FROM SmartLoop)
SELECT * FROM SmartLoop;

WITH RECURSIVE DummyLoop
(Counter)
AS (SELECT 1 FROM EMP
    UNION ALL
    SELECT 2 FROM DummyLoop)
```

Datalog

- Język teoretyczny (podobnie jak ALG i FOL)
- Prawie jak FOL
- Prawie jak Prolog
- Symbole relacyjne
 - intensjonalne (wyliczane)
 - ekstensjonalne (zachowane w bazie danych)
- Semantykę inflacyjną: intensjonalne puste na początku
- Potem wypełniane w miarę obliczeń

Datalog – przykład

$\text{EmpHier}(n, id, \text{NULL}) \leftarrow \text{Emp}(n, id, \text{NULL})$

$\text{EmpHier}(n, id, mgr) \leftarrow$

$\text{Emp}(n, id, mgr), \text{EmpHier}(_, mgr, _)$

- Ekstensjonalny: Emp
- Intensjonalny: EmpHier
 - Bez LEVEL, bo nie ma arytmetyki (byłoby to pogwałcenie zasady ograniczenia do dziedziny aktywnej)

Datalog – przykład obliczenia

1. $\text{EmpHier} = \emptyset$
2. $\text{EmpHier} = \{ \langle \text{KING}, 7839, \text{NULL} \rangle \}$
3. $\text{EmpHier} = \{ \langle \text{KING}, 7839, \text{NULL} \rangle, \langle \text{JONES}, 7566, 7839 \rangle, \langle \text{BLAKE}, 7698, 7839 \rangle, \langle \text{CLARK}, 7782, 7839 \rangle \}$
4. $\text{EmpHier} = \{ \dots, \langle \text{SCOTT}, 7788, 7566 \rangle, \dots \}$

Podsumowanie rekurencji w zapytaniach

- CONNECT BY przeszukuje istniejący graf, zawsze się zatrzymuje
 - Wyrażenia arytmetyczne mogą tworzyć nowe wartości ale nie węzły
- WITH RECURSIVE buduje i przeszukuje graf, więc może się nie zatrzymać
 - Np. arytmetyka pozwala tworzyć nowe węzły
- Datalog
 - Zawsze jest stop
 - Generuje nowe krotki, ale tylko w ramach (skończonej!) dziedziny aktywnej

Bazy danych



04:

Zależności funkcyjne
i postaci normalne

Krzysztof Stencel

Wadliwe schematy danych

Wykładowca	Krwawość	Wykład	Trudność	Termin
Stencel	3	BD	2	14:15
Stencel	3	SZBD	3	8:30
Jabłonowski	2	PO	2	12:15
Ciebiera	1	BD	2	10:15
Ciebiera	1	ZPP	4	8:30

Anomalie aktualizacyjne

- **Przy wstawianiu:** nie da się wstawić wykładowcy bez wykładu
- **Przy usuwaniu:** usuwając PO wykładane przez Jabłonowskiego usuwamy też jego samego
- **Przy modyfikacji:** zmiana krwawości Stencla wymaga zmiany więcej niż jednego wiersza
- **Nadmiarowość:** ta sama dana jest zapisana więcej niż jeden raz

Postulat normalizacji

- Każdy fakt jest przechowywany w bazie danych dokładnie jeden raz.
- Wtedy nie będzie też anomalii.
- Reszta wykładu, to uszczegółowienie tego postulatu.

Zależność funkcyjna

- $X, Y \subseteq \text{kol}(r)$ – pewne podzbiory kolumn
- Zależność funkcyjna $X \rightarrow Y$ zachodzi w relacji r wtedy i tylko wtedy, gdy

$$\forall t, u \in r : t \upharpoonright X = u \upharpoonright X \Rightarrow t \upharpoonright Y = u \upharpoonright Y$$

- Innymi słowy, równość wartości w kolumnach ze zbioru X implikuje równość (*determinuje*) wartości w kolumnach ze zbioru Y

Przykłady zależności funkcyjnych

- Ciekawe
 - Wykładowca \rightarrow Krwawość
 - Wykład \rightarrow Trudność
 - Wykładowca, Wykład \rightarrow Termin
- Wtórne
 - Wykładowca, Wykład \rightarrow Krwawość, Trudność
- Trywialne
 - Wykład \rightarrow Wykład
 - Wykładowca, Krwawość \rightarrow Krwawość

Pierwsza postać normalna (1NF)

- Dane są atomowe.
- Każdy element danych (komórka tabeli) jest niepodzielna.
- Dozwolone są tylko skalarne typy danych.
- Nic bardziej skomplikowanego nie: wektory, listy, drzewa, tablice, krotki, relacje itd.

Demagogia pro-1NF

- Pierwsza postać normalna pozwala rozważać zależności funkcyjne całości atrybutów od całości atrybutów
- Gdyby była kolumna *Adres* złożona z ulicy, nr domu, kodu oraz kolumna *Poczta*, to trzeba by zapisać zależność funkcyjną jakoś tak:
$$\frac{1}{3} \text{ Adres} \rightarrow \text{Poczta}$$
- Bo *Poczta* zależy od kodu, czyli kawałka *Adresu*

Schemat relacji z zależnościami funkcyjnymi

- Postacie normalne są związane z projektowaniem baz danych
- Trzeba więc oderwać się od konkretnych relacji, bo ich jeszcze nie ma (jesteśmy w fazie projektowania)
- Schemat tabeli to para:

$$\langle R, \Sigma \rangle$$

gdzie R - kol(r), a Σ to zbiór zależności funkcyjnych nad R .

Konsekwencja semantyczna

- Oznaczenie: $r \models X \rightarrow Y$ (w r zachodzi zależność funkcyjna $X \rightarrow Y$)
- $r \models \Sigma$, wtw. $r \models X \rightarrow Y$ dla każdego $X \rightarrow Y \in \Sigma$
- $\Sigma \models X \rightarrow Y$ wtw. dla każdej relacji r jeśli $r \models \Sigma$,
to $r \models X \rightarrow Y$
- Jeśli $\Sigma \models X \rightarrow Y$, to $X \rightarrow Y$ nazwiemy
konsekwencją semantyczną Σ

Aksjomaty Armstronga

- Reguły systemu dowodowego dla zależności funkcyjnych

$$\frac{X \supseteq Y}{X \rightarrow Y}$$

$$\frac{X \rightarrow Y}{XZ \rightarrow YZ}$$

$$\frac{X \rightarrow Y \quad Y \rightarrow Z}{X \rightarrow Z}$$

Dowód zależności funkcyjnej

- Dowodem z.f. $X \rightarrow Y$ ze zbioru z.f. Σ nazwiemy ciąg z.f. z_1, z_2, \dots, z_n o następujących właściwościach:
 - $z_n = X \rightarrow Y$
 - dla każdego $i = 1, 2, \dots, n$,
 - $z_i \in \Sigma$ lub
 - z_i można otrzymać korzystając z pewnego aksjomatu Armstronga na podstawie z.f. z_1, z_2, \dots, z_{i-1}
- Oznaczenie: Jeśli istnieje dowód $X \rightarrow Y$ z Σ , to powiemy, że $\Sigma \vdash X \rightarrow Y$ („jest dowodliwe”)

Ciekawe właściwości

- **Poprawność i pełność**

$$\Sigma \vdash X \rightarrow Y \Leftrightarrow \Sigma \models X \rightarrow Y$$

- **Rozstrzygalność** – oczywiste – wszystko jest skończone, można więc wygenerować domknięcie zbioru z.f.

$$\Sigma^* = \{ X \rightarrow Y : \Sigma \vdash X \rightarrow Y \}$$

- **Jednoznaczność** – dla każdego Σ^* istnieje relacja r , w której spełnione są tylko z.f. z Σ^*

Domknięcie zbioru kolumn

- Domknięcie zbioru kolumn (względem Σ):

$$Z^{*(\Sigma)} = \{ k \in R : \Sigma \vdash Z \rightarrow * \}$$

- Obliczanie (wymaga dobrej struktury danych)

$$Z^* := Z$$

powtarzaj w kółko

weź $X \rightarrow Y$ z Σ , takie, że $X \subseteq Z^*$

$$Z^* := Z^* \cup Y$$

$$\Sigma := \Sigma - \{ X \rightarrow Y \}$$

jeśli nie było takiego $X \rightarrow Y$, to koniec

Klucz – kluczowe pojęcie

- **Nadkluczem** schematu $\langle R, \Sigma \rangle$ nazwiemy każdy zbiór kolumn X , taki, że:

$$X^{*(\Sigma)} = R$$

- **Klucz** schematu $\langle R, \Sigma \rangle$ to nadklucz, który jest minimalny (w sensie \subseteq) w tym schemacie
- Jedna relacja może mieć wiele kluczy
- Klucz jest centralnym pojęciem teorii postaci normalnych

Przykład klucza i nadkluczy

- $R = \{\text{Wykładowca}, \text{Krwawość}, \text{Wykład}, \text{Trudność}, \text{Termin}\}$
 - $\text{Wykładowca} \rightarrow \text{Krwawość}$
 - $\text{Wykład} \rightarrow \text{Trudność}$
 - $\text{Wykładowca}, \text{Wykład} \rightarrow \text{Termin}$
- Nadklucze
 - $\{\text{Wykładowca}, \text{Wykład}, \text{Krwawość}\}$
 - – $\{\text{Wykładowca}, \text{Wykład}\} \leftarrow \text{klucz}$
 - $\{\text{Wykładowca}, \text{Wykład}, \text{Krwawość}, \text{Trudność}, \text{Termin}\}$
 - $\{\text{Wykładowca}, \text{Wykład}, \text{Trudność}\}$

Druga postać normalna (2NF)

- Schemat $\langle R, \Sigma \rangle$ jest 2NF, wtw. jest 1NF oraz dla każdej z. f. $X \twoheadrightarrow A \in \Sigma^*$ zachodzi co najmniej jeden z warunków:
 - $A \in X$ (zależność trywialna)
 - X nie jest właściwym podzbiorem żadnego klucza (brak *zależności częściowych*)
 - A należy do pewnego klucza (atrybut główny)
- Zależność częściowa od klucza $X \twoheadrightarrow A$:
dla pewnego klucza K , A zależy od części K

Trzecia postać normalna (3NF)

- Schemat $\langle R, \Sigma \rangle$ jest 3NF, wtw. jest 1NF oraz dla każdej z. f. $X \twoheadrightarrow A \in \Sigma^*$ zachodzi co najmniej jeden z warunków:
 - $A \in X$ (zależność trywialna)
 - X jest nadkluczem (brak *zależności przechodnich*)
 - A należy do pewnego klucza (atrybut główny)
- Zależność przechodnia od klucza $X \twoheadrightarrow A$:
dla pewnego klucza K zachodzi $K \twoheadrightarrow X \twoheadrightarrow A$

Postać normalna Boyce'a-Codda (BCNF)

- Schemat $\langle R, \Sigma \rangle$ jest BCNF, wtw. jest 1NF oraz dla każdej z.f. $X \twoheadrightarrow A \in \Sigma^*$ zachodzi co najmniej jeden z warunków:
 - $A \in X$ (zależność trywialna)
 - X jest nadkluczem (brak *zależności przechodnich*)
- W stosunku do 3NF brak „furtki” na atrybut główny
- $1NF \Leftarrow 2NF \Leftarrow 3NF \Leftarrow BCNF \Leftarrow \text{cdn} \dots$

Analiza przykładowego schematu

- $R = \{\text{Wykładowca}, \text{Krwawość}, \text{Wykład}, \text{Trudność}, \text{Termin}\}$
 - $\text{Wykładowca} \rightarrow \text{Krwawość}$
 - $\text{Wykład} \rightarrow \text{Trudność}$
 - $\text{Wykładowca}, \text{Wykład} \rightarrow \text{Termin}$
- Jedyny klucz: $\{\text{Wykładowca}, \text{Wykład}\}$
- Dwie pierwsze z.f. są więc częściowe
- Nie ma więc 2NF (a więc i 3NF, BCNF, itd.)
- Brak 2NF/3NF jest przyczyną anomalii
- Brak BCNF gdy 3NF nie powoduje anomalii

Bazy danych



05:
Normalizacja

Krzysztof Stencel

Normalizacja

- = Sprowadzenie tabel do odpowiedniej postaci normalnej
- Sprowadzenie dokonuje się poprzez podział tabel na mniejsze części:
- $R = \text{kol}(r)$, podziałem nazwiemy ciąg zbiorów P_1, P_2, \dots, P_n , taki że:

$$P_1 \cup P_2 \cup \dots \cup P_n = R$$

- Nie jest i (nie może być!) rozłączny ze względu na zachowanie informacji.

Zachowywanie informacji

- Podział P_1, P_2, \dots, P_n schematu $\langle R, \Sigma \rangle$ zachowuje informacje wtw. dla każdej relacji r o schemacie $\langle R, \Sigma \rangle$ zachodzi warunek:

$$\pi_{P_1}(r) \bowtie \pi_{P_2}(r) \bowtie \dots \bowtie \pi_{P_n}(r) = r$$

- Warunek konieczny sensowności normalizacji.
- Można z niego zrezygnować, gdy w tabeli połączono niezwiązane ze sobą dane.

Zachowywanie zależności funkcyjnych

- $\Sigma \upharpoonright P$ to podzbiór Σ zawierający wszystkie zależności funkcyjne, które nie zawierają odwołań do kolumn spoza P
- Podział P_1, P_2, \dots, P_n schematu $\langle R, \Sigma \rangle$ zachowuje zależności funkcyjne wtw.

$$(\Sigma^* \upharpoonright P_1 \cup \Sigma^* \upharpoonright P_2 \cup \dots \cup \Sigma^* \upharpoonright P_n)^* = \Sigma^*$$

- Dobrze jest zachować zależności funkcyjne (bo stanowią semantykę danych), ale nie jest to konieczne.

Właściwości normalizacji 3NF i BCNF

- Każdy schemat można sprowadzić do 3NF z zachowaniem zależności funkcyjnych i informacji
- Każdy schemat można sprowadzić do BCNF z zachowaniem informacji
- Przejście od 3NF do BCNF nie usuwa żadnych anomalii
- Za to jest prostsze

Tabela w 3NF bez „taniego” przejścia do BCNF

- $R = \{ \text{Kino, Miasto, Film} \}$
 - $\text{Kino} \rightarrow \text{Miasto}$
 - $\text{Miasto, Film} \rightarrow \text{Kino}$
- Klucze: $\{ \text{Kino, Film} \}, \{ \text{Miasto, Film} \}$
- Wszystkie atrybuty są kluczowe, więc 3NF
- Nie da się podzielić (przejsć do BCNF) bez utraty drugiej zależności funkcyjnej
- Takie tabele są na szczęście rzadkie

Normalizacja BCNF

- Weź zależność funkcyjną $X \rightarrow Y$ ($X \cap Y = \emptyset$), która powoduje „problemy” i podziel tabelę R na dwie części:
 - $X \cup Y$
 - $R - Y$
- Rób tak aż do chwili, w której wszystkie fragmenty są BCNF
- Informacje są zachowane, bo obie te tabele połączone naturalnie dają wyjściową tabelę (klejem jest X występujący w obu tabelach)

Przykład normalizacji BCNF

- **$R = \{\text{Wykładowca}, \text{Krwawość}, \text{Wykład}, \text{Trudność}, \text{Termin}\}$**
 - **$\text{Wykładowca} \rightarrow \text{Krwawość}$**
 - **$\text{Wykład} \rightarrow \text{Trudność}$**
 - **$\text{Wykładowca}, \text{Wykład} \rightarrow \text{Termin}$**
- **Krok 1: $\text{Wykładowca} \rightarrow \text{Krwawość}$**
 - **$R_1 = \{ \text{Wykładowca}, \text{Krwawość} \}$**
 - **$R_2 = \{ \text{Wykładowca}, \text{Wykład}, \text{Trudność}, \text{Termin} \}$**
- **Krok 2: $\text{Wykład} \rightarrow \text{Trudność}$**
 - **$R_{21} = \{ \text{Wykład}, \text{Trudność} \}$**
 - **$R_{22} = \{ \text{Wykładowca}, \text{Wykład}, \text{Termin} \}$**

Wynik normalizacji BCNF

- Podział schematu $R = \{\text{Wykładowca}, \text{Krwawość}, \text{Wykład}, \text{Trudność}, \text{Termin}\}$
 - $R_1 = \{\text{Wykładowca}, \text{Krwawość}\}$
 - $R_{21} = \{\text{Wykład}, \text{Trudność}\}$
 - $R_{22} = \{\text{Wykładowca}, \text{Wykład}, \text{Termin}\}$
- Zachowano też zależności funkcyjne (widać!)
- Każda powstała tabelka jest już w BCNF
 - Zależności funkcyjne w niej obowiązujące mają reprezentację złożoną z jednej z.f. z wszystkimi kolumnami
- Przy normalizacji BCNF konieczna jest staranność, żeby nie „zrywać” zależności funkcyjnych.

Minimalna reprezentacja

Zbiór z.f. Γ to minimalna reprezentacja zbioru z.f. Σ wtw.

- Dla każdej z.f. $X \rightarrow Y \in \Gamma$ zbiór Y ma jeden element
 - Dla każdej z.f. $X \rightarrow Y \in \Sigma$, $\Gamma \vdash X \rightarrow Y$
 - Jeśli $X \rightarrow A \in \Gamma$, to nie jest prawdą, że
$$\Gamma - \{X \rightarrow A\} \vdash X \rightarrow A$$
 - Jeśli $XZ \rightarrow A \in \Gamma$ i $Z \neq \emptyset$, to nie jest prawdą, że
$$\Gamma \vdash X \rightarrow A$$
- Minimalna reprezentacja jest potrzebna do normalizacji 3NF, bo ta polega na wygenerowaniu tabeli dla każdej zależności funkcyjnej

Normalizacja 3NF

- Mając dany schemat $\langle R, \Sigma \rangle$ wyznacz jego minimalną reprezentację Γ .
- Dla każdej z.f. $X \rightarrow A \in \Gamma$, utwórz tabelę o kolumnach $X \cup A$
 - Dla zachowania zależności funkcyjnych
- Wybierz K – jeden z kluczy schematu $\langle R, \Sigma \rangle$ i utwórz tabelę o kolumnach K
 - Dla zachowania informacji

Przykład normalizacji 3NF

- $R = \{ \text{Wykładowca}, \text{Krwawość}, \text{Wykład}, \text{Trudność} \}$
 - $\text{Wykładowca} \rightarrow \text{Krwawość}$
 - $\text{Wykład} \rightarrow \text{Trudność}$
 - Jest to też minimalna reprezentacja
- Krok 1: tabela dla każdej zależności funkcyjnej
 - $R_1 = \{ \text{Wykładowca}, \text{Krwawość} \}$
 - $R_2 = \{ \text{Wykład}, \text{Trudność} \}$
- Krok 2: Tabela dla jednego z kluczy
 - $R_3 = \{ \text{Wykład}, \text{Wykładowca} \}$
 - Bez kroku 2 nie zachowujemy informacji o tym, kto prowadzi wykłady

Normalizacja 3NF: Optymalizacja 1

- Usuń każdą tabelę, której zbiór kolumn jest podzbiorem zbioru kolumn jakiejś innej tabeli
 - $R = \{ \text{Wykładowca}, \text{Krwawość}, \text{Wykład}, \text{Trudność}, \text{Termin} \}$
 - $\text{Wykładowca} \rightarrow \text{Krwawość}$
 - $\text{Wykład} \rightarrow \text{Trudność}$
 - $\text{Wykład}, \text{Wykładowca} \rightarrow \text{Termin}$
-
- $R_1 = \{ \text{Wykładowca}, \text{Krwawość} \}$ (z.f.)
 - $R_2 = \{ \text{Wykład}, \text{Trudność} \}$ (z.f.)
 - $R_3 = \{ \text{Wykład}, \text{Wykładowca}, \text{Termin} \}$ (z.f.)
 - $R_4 = \{ \text{Wykład}, \text{Wykładowca} \}$ (klucz)

Normalizacja 3NF: Optymalizacja 2

- Połącz tabelę pochodzące od zależności funkcyjnych z tym samymi lewymi stronami
- $R = \{ \text{Wykładowca}, \text{Krwawość}, \text{Zasięg}, \text{Wykład}, \text{Trudność} \}$
 - $\text{Wykładowca} \rightarrow \text{Krwawość}$
 - $\text{Wykładowca} \rightarrow \text{Zasięg}$
 - $\text{Wykład} \rightarrow \text{Trudność}$
- $R_{12} = \{ \text{Wykładowca}, \text{Krwawość}, \text{Zasięg} \}$ (z.f.)
- $R_3 = \{ \text{Wykład}, \text{Trudność} \}$ (z.f.)
- $R_4 = \{ \text{Wykład}, \text{Wykładowca} \}$ (klucz)

Optymalizacja 2 jest poprawna

- Weźmy tabele o schemacie $R = \{ A, B, C, \dots \}$ i minimalnej reprezentacji zawierającej
 - $A \rightarrow B$
 - $A \rightarrow C$
- Czy $\{ A, B, C \}$ może nie być 3NF? Wtedy musiałoby być, np. $B \rightarrow C$ i też jakoś musiałoby wynikać z minimalnej reprezentacji
- Ale wtedy $A \rightarrow C$ i $A \rightarrow B$ nie mogą być razem w minimalnej reprezentacji
- Porządny dowód: przez indukcję

Inne redundancje

Imię	Lubi	Uprawia
Adam	Spać	Brydż
Andrzej	Jeździć	Tenis
Andrzej	Pływać	Tenis
Andrzej	Jeździć	Konopie
Andrzej	Pływać	Konopie

To było BCNF

- Nie ma tu żadnej nietrywialnej zależności funkcyjnej
- Klucz składa się ze wszystkich kolumn
- Generują się jednak lokalnie iloczyny kartezyjańskie na kolumnach *Lubi* i *Uprawia*
- Takie redundancje występują jednak niezwykle rzadko i zdarza się to tylko wyjątkowo przemęczonym projektantom

Zależność wielowartościowa

- $X, Y \subseteq R$ – podzbiory kolumn
- Oznaczenie $Z := R - X - Y$
- Zależność wielowartościowa $X \twoheadrightarrow Y$ zachodzi w relacji r wtedy i tylko wtedy, gdy

$\forall t, u \in r:$

$$t \upharpoonright X = u \upharpoonright X \Rightarrow$$

$$\exists w \in r (w \upharpoonright X = t \upharpoonright X \wedge w \upharpoonright Y = t \upharpoonright Y \wedge w \upharpoonright Z = u \upharpoonright Z)$$

$X \rightarrow Y$ – wyjaśnienie

$\forall t, u \in r:$

$t \upharpoonright X = u \upharpoonright X \Rightarrow$

$\exists w \in r (w \upharpoonright X = t \upharpoonright X \wedge w \upharpoonright Y = t \upharpoonright Y \wedge w \upharpoonright Z = u \upharpoonright Z)$

$t : \text{XXXXXXX } \textcolor{red}{YYYYYYY} \textcolor{red}{ZZZZZZ}$

$u : \text{XXXXXXX } \textcolor{green}{YYYYYYY} \textcolor{green}{ZZZZZZ}$

\Rightarrow

$w : \text{XXXXXXX } \textcolor{red}{YYYYYYY} \textcolor{green}{ZZZZZZ}$

Zależność funkcyjna jest też wielowartościowa

- Zakładamy, że zachodzi $X \rightarrow Y$ ($X, Y \subseteq R$)
- Weźmy dwie krotki t i u , takie, że $t \upharpoonright X = u \upharpoonright X$
 t : XXXXXX YYYYYYY ZZZZZZ
 u : XXXXXX YYYYYYY ZZZZZZ
- Czy istnieje w ?
 w : XXXXXX YYYYYYY ZZZZZZ
- Tak, jest nią u
- Bo przecież $X \rightarrow Y$, a więc YYYYYYY = YYYYYYY

Czwarta postać normalna (4NF)

- Schemat jest 4NF, wtw. jest 1NF oraz każda zależność wielowartościowa $X \twoheadrightarrow A$ jest również zależnością funkcyjną od nadklucza
- Normalizacja jak do BCNF:
- Weź z.w. $X \twoheadrightarrow Y$ ($X \cap Y = \emptyset$), która powoduje „problemy” i podziel tabelę R na dwie części:
 - $X \cup Y$
 - $R - Y$

W 4NF też mogą być redundancje

- Schemat $R = \{ \text{Projekt, Towar, Dostawca} \}$
- Jeśli:
 - Dostawca D dostarcza towar T dla jakiegoś projektu
 - Dostawca D dostarcza jakiś towar dla projektu P
 - Jakiś dostawca dostarcza towar T dla projektu P
- To:
 - Dostawca D dostarcza towar T dla projektu P

Bazowe fakty

- Fakty biznesowe sklejone w tej tabeli to
 - Dostawca D jest w stanie dostarczyć towar T
 - Dostawca D pracuje dla projektu P
 - Projekcie P zużywa towar T

Dostawca	Towar	Projekt
<u>Parys</u>	<u>Jabłka</u>	Eris
<u>Parys</u>	Wiśnie	<u>Afrodyta</u>
Hektor	<u>Jabłka</u>	<u>Afrodyta</u>
Parys	Jabłka	Afrodyta

Wiersz nadmiarowy

Zależność łączeniowa

- P_1, P_2, \dots, P_n – podział R
- Zależność łączeniowa $\bowtie(P_1, P_2, \dots, P_n)$ zachodzi w relacji r wtedy i tylko wtedy, gdy spełniony jest warunek:

$$\pi_{P_1}(r) \bowtie \pi_{P_2}(r) \bowtie \dots \bowtie \pi_{P_n}(r) = r$$

- Tzn. istnienie zależności łączeniowej jest równoważne z zachowywaniem informacji przez podział

Piąta postać normalna (5NF)

- Schemat jest 5NF, wtw. jest 1NF oraz każda zależność złączeniowa $\bowtie(P_1, P_2, \dots, P_n)$ jest indukowana przez nadklucz, tj.

$P_1 \cap P_2 \cap \dots \cap P_n$ jest nadkluczem

- Jeśli tabela jest 5NF, to żaden jej podział zachowujący informacje nie ma sensu (bo i tak w każdym fragmencie będzie ten sam klucz)

Normalizacja 5NF

- Jeśli $\bowtie(P_1, P_2, \dots, P_n)$ nie jest indukowana przez nadklucz, to podziel R na P_1, P_2, \dots, P_n .
- 5NF jest ostateczną postacią normalną
- Dalej już się normalizować się nie da (gdy zakładamy normalizację przez podział!)

Przykład zależności łączeniowej i normalizacji

- Schemat $R = \{ \text{Projekt}, \text{Towar}, \text{Dostawca} \}$
 $\bowtie(\{ \text{Projekt}, \text{Towar} \},$
 $\{ \text{Projekt}, \text{Dostawca} \},$
 $\{ \text{Towar}, \text{Dostawca} \} \quad)$
- Podziel R na:
 $\{ \text{Projekt}, \text{Towar} \},$
 $\{ \text{Projekt}, \text{Dostawca} \},$
 $\{ \text{Towar}, \text{Dostawca} \}$

Obiecanka relacyjna (przełożył Lech Banachowski)

Bez powtórzeń,
dane zależą od klucza,
od całego klucza
i tylko od klucza.

Tak mi dopomóż Codd!

- The Key, the whole Key, and nothing but the Key, **so help me Codd.**

Bazy danych

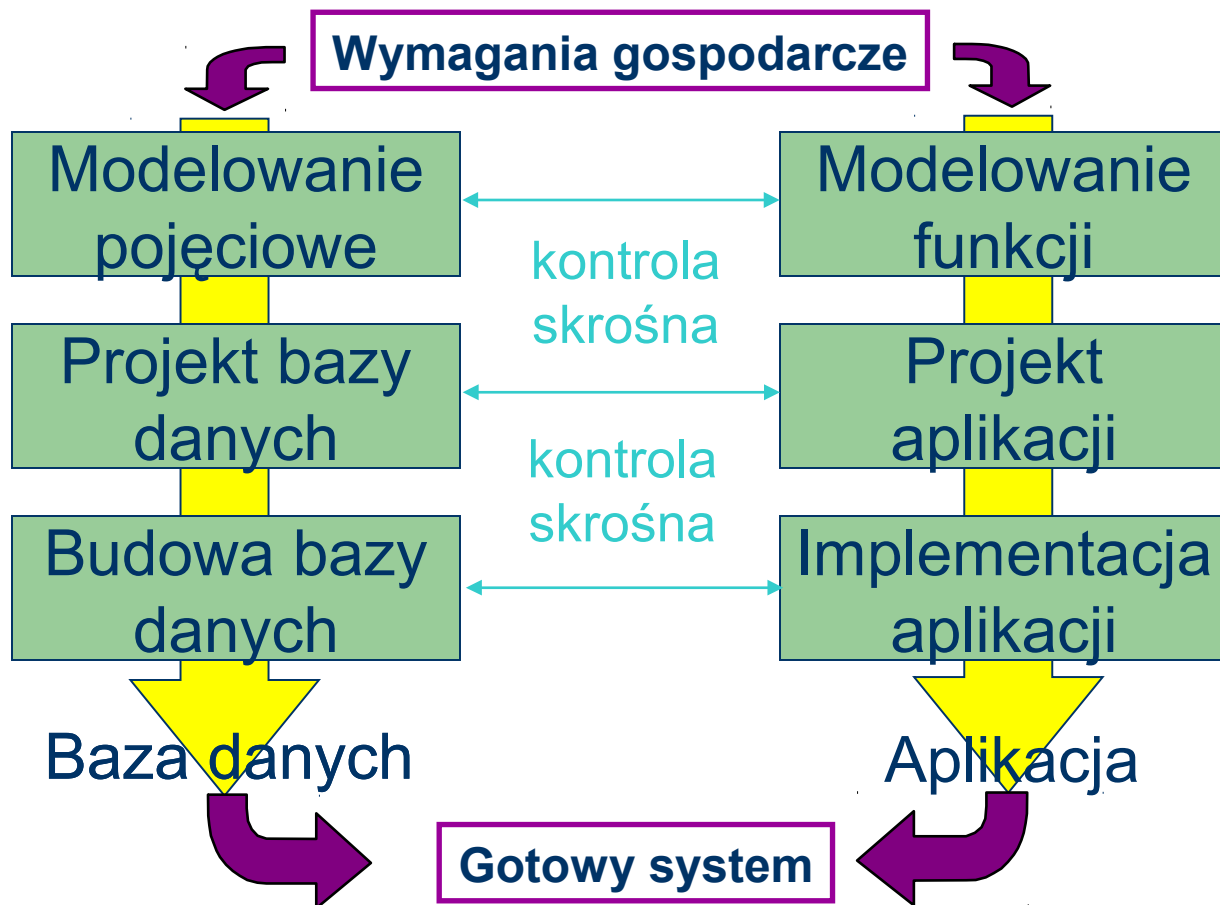


06:

Modelowanie związków
encji

Krzysztof Stencel

Proces wytwarzania aplikacji



Proces wytwarzania bazy danych

Wymagania informacyjne

Modelowanie
pojęciowe

Modelowanie
logiczne

Modelowanie
fizyczne

Gotowa baza danych

Cel modelowania informacji

- Uporządkowanie procesu myślowego
- Precyzyjne modelowanie danych gospodarczych
- Komunikacja z uczestnikami, udziałowcami, sponsorami, użytkownikami itd.
- Analiza zakresu
- Położenie solidnego fundamentu pod projekt systemu

Encja

- Obiekt interesujący gospodarczo
- Klasa (kategoria) rzeczy
- Rzecz nazwana
- Rzeczownik
- Rzecz istotna, o której firma musi przechowywać informacje

Encja – krótka weryfikacja

- Znalazłeś rzeczownik w wymaganiach gospodarczych?
- Czy jest on istotny?
- Czy jakieś informacje o nim musi firma przechowywać?
- Czy to grupa czy jeden element?

Encja – zbiór egzemplarzy

- Zbiór egzemplarzy encji jest to kolekcja podobnych encji w sensie posiadania wspólnych atrybutów
- Encja to liczba mnoga. Prawie jak klasa. Prawie robi wielką różnicę (czasem).

Entia non sunt multiplicanda praeter necessitatem

- Entia (łac.) = byty [l.poj. ens]
- Myśl przypisywana Wilhelmowi Ockhamowi (informatykom znany jest on lepiej jako Occam → zob. PW), ale sformułowana w ten sposób później. Tzw. brzytwa Occama
- Nie należy mnożyć bytów ponad potrzebę
- Każda encja winna być ens necessarius

Atrybut

- Atrybut to właściwość wystąpień encji, jest pewną wartością np. liczbą, napisem.
- Atrybut powinien opisywać egzemplarze encji, przy której się go umieszcza (a nie związki z innymi encjami).
- 1NF. Dla każdego egzemplarza encji każdy jej atrybut powinien przyjmować pojedynczą, atomową wartość.
- Należy pomijać atrybuty wyliczane tj. takie których wartości dadzą się wyliczyć z innych

Atrybut – znaczenie gospodarcze

- Rzeczownik służący do opisu encji
- Konkretny kwant informacji, którą trzeba znać
- Encja powinna mieć atrybuty (jak inaczej rozróżnić jej egzemplarze? Albo, po co to robić?)

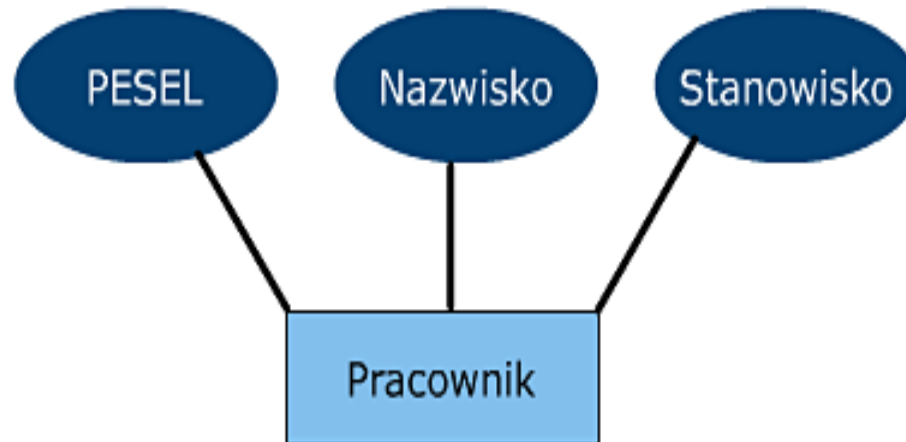
Przykładowy tekst wymagań informacyjnych

Jestem kierownikiem firmy szkoleniowej, która prowadzi kursy komputerowe. Każdy z kursów ma kod, nazwę i opłatę. Wstęp do Uniksa i Programowanie w C to najpopularniejsze kursy. Kursy mają różną długość od 1 do 4 dni. Krwawy MundeK to nasz najlepszy wykładowca. Znamy nazwisko i telefon każdego wykładowcy. Studenci mogą uczęszczać na kilka kursów naraz i wielu to robi. Bolek i Lolek wzięli wszystkie! Zapisujemy nazwisko i telefon każdego studenta.

Model danych dla przykładu



Encja z atrybutami



- Jest masa notacji diagramów związków encji
- Tu używam najbardziej klasycznej i najbogatszej, tj. notacji Petera Chena.

Identyfikator jednoznaczny

- Jednoznaczny identyfikator to niepusty zbiór (być może jednoelementowy) atrybutów danej encji, których wartości jednoznacznie identyfikują każdy egzemplarz tej encji. Jedna encja może mieć wiele identyfikatorów.
- Jeden identyfikator jest wskazywany jako główny, pozostałe jako alternatywne.
- Klucze obce w zasadzie odnoszą się do identyfikatora (klucza) głównego.

Związek – znaczenie gospodarcze

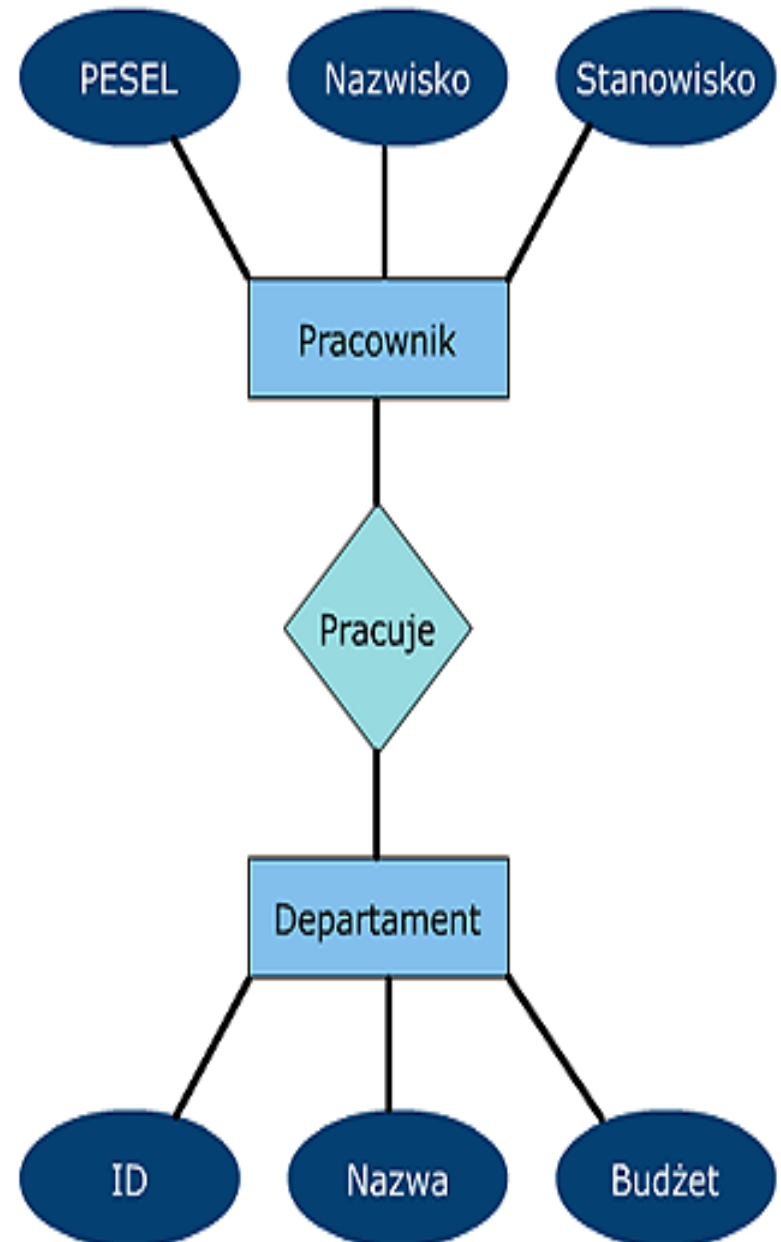
- Sposób, w jaki encje odnoszą się do siebie
- Reguły gospodarcze kojarzące ze sobą wymagania gospodarcze
- Czynność, którą wykonuje jeden byt z drugim
- Nazwana asocjacja między encjami

Związek

- Związek to uporządkowana lista encji
- Poszczególne encje mogą występować wielokrotnie w jednym związku.
- Każdy związek określa pewną relację między zbiorami egzemplarzy encji wchodzących w skład związku – zbiór egzemplarzy związku.
- Związek można formalnie zapisać przy użyciu notacji relacyjnej $Z(E_1, \dots, E_n)$ co oznacza: encje E_1, \dots, E_n są w związku Z .

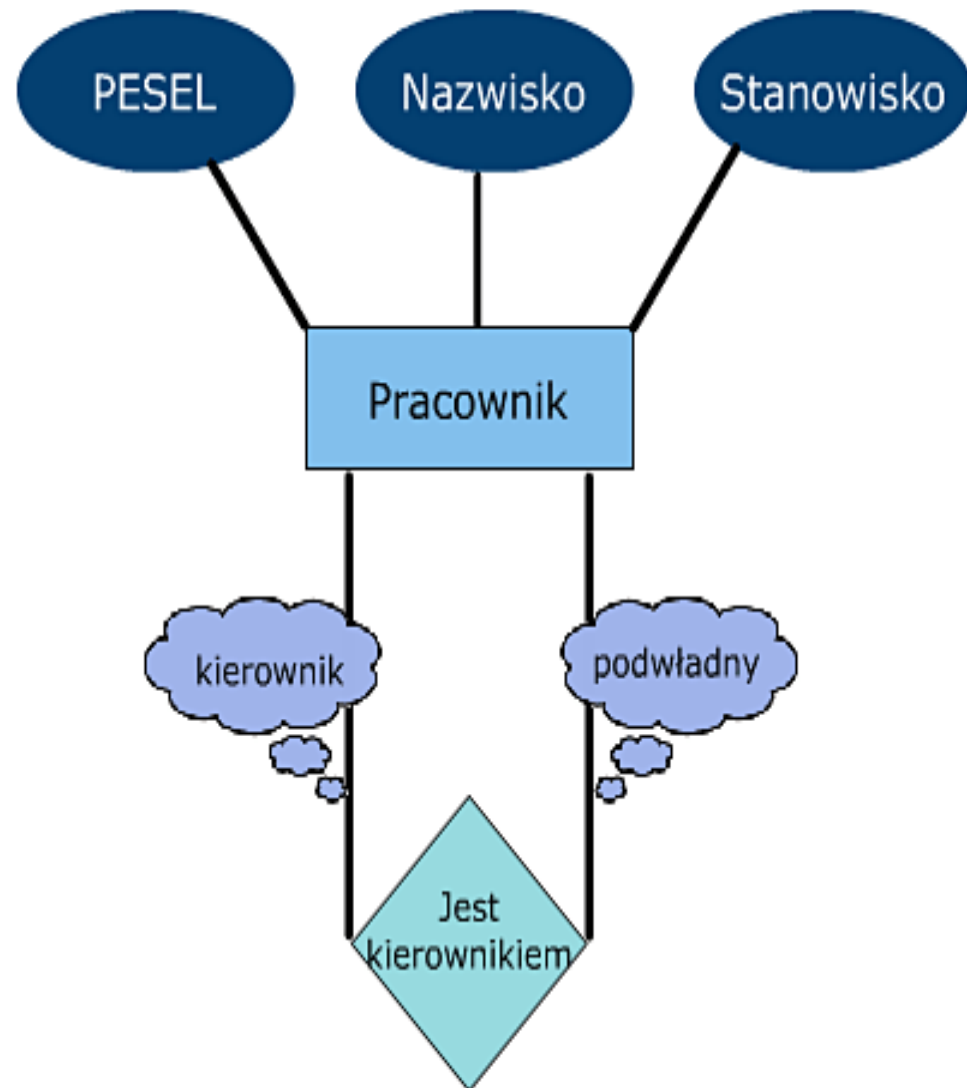
Związek dwuargumentowy

- Także: związek binarny
- Pracownik pracuje w Departamencie



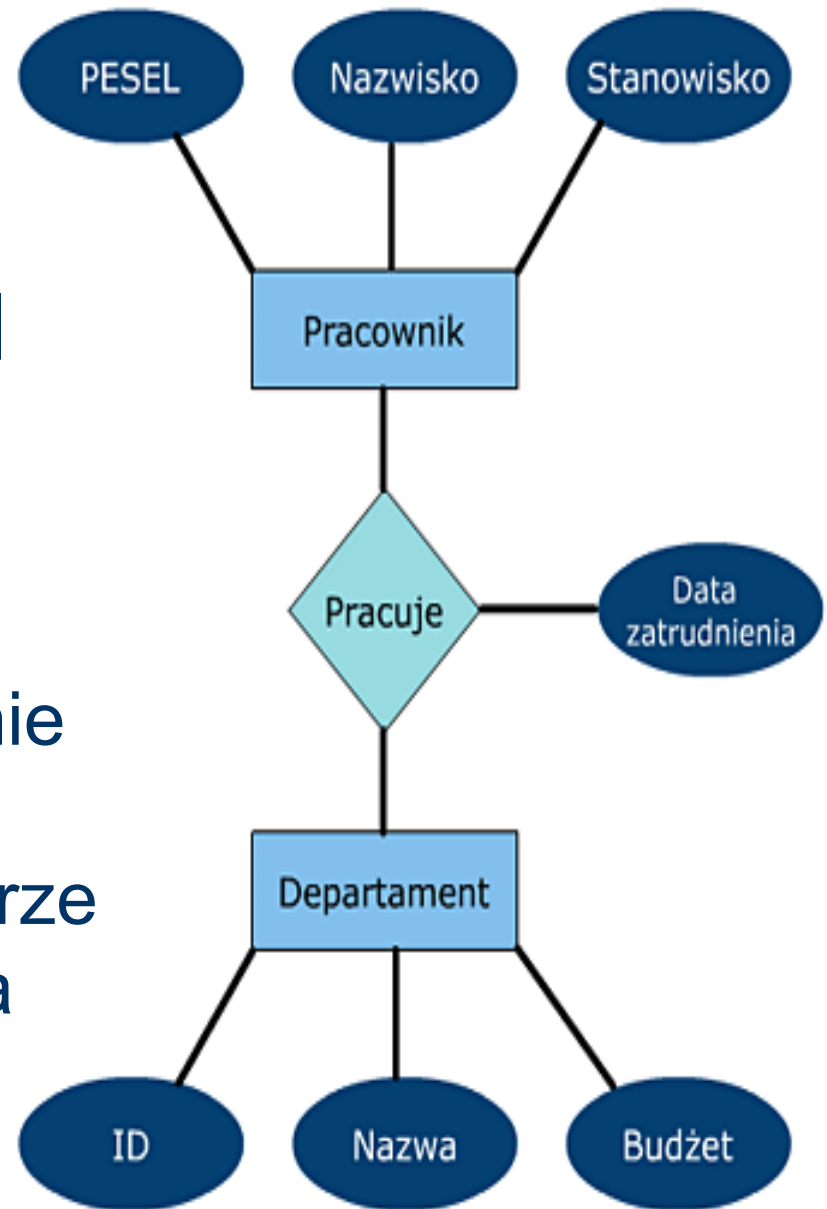
Związek rekurencyjny

- Związek, w którym ta sama encja jest dwa razy
- Role w związku: w jaką rolę odgrywa dane wystąpienie w tym związku?



Atrybut związku

- Atrybut, który opisuje związek (a nie encję).
- Egzemplarz związku może być jednoznacznie identyfikowany przez tworzące go egzemplarze encji (bez odwoływania się do jego atrybutów)

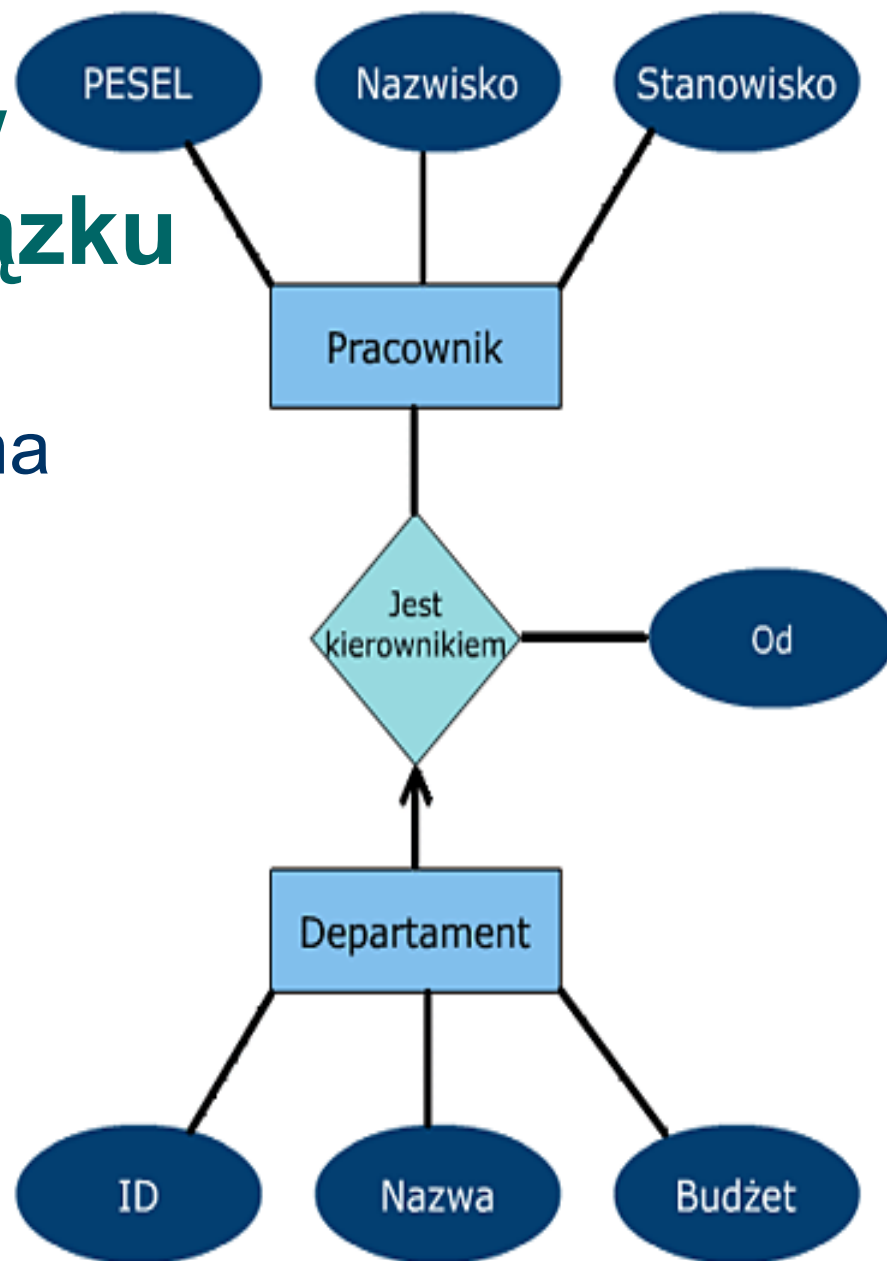


Więzy kluczowe związku

- Więzy kluczowe związku oznaczają, że dla każdego układu egzemplarzy encji istnieje co najwyżej jeden egzemplarz związku zawierający te egzemplarze encji.
- Inaczej mówiąc, między zbiorami egzemplarzy encji jest określona funkcja częściowa.
- W szczególności, więzy kluczowe związku binarnego oznaczają, że jest to związek wiele-do-jeden czyli związek jednoznaczny.
- Gdy egzemplarz związku binarnego jest funkcją częściową 1-1 mamy do czynienia ze związkiem jeden-do-jeden czyli związkiem jedno-jednoznaczny.

Przykład więzów kluczowych związku

- Każdy departament ma dokładnie jednego kierownika
- Pracownik może być kierownikiem wielu departamentów

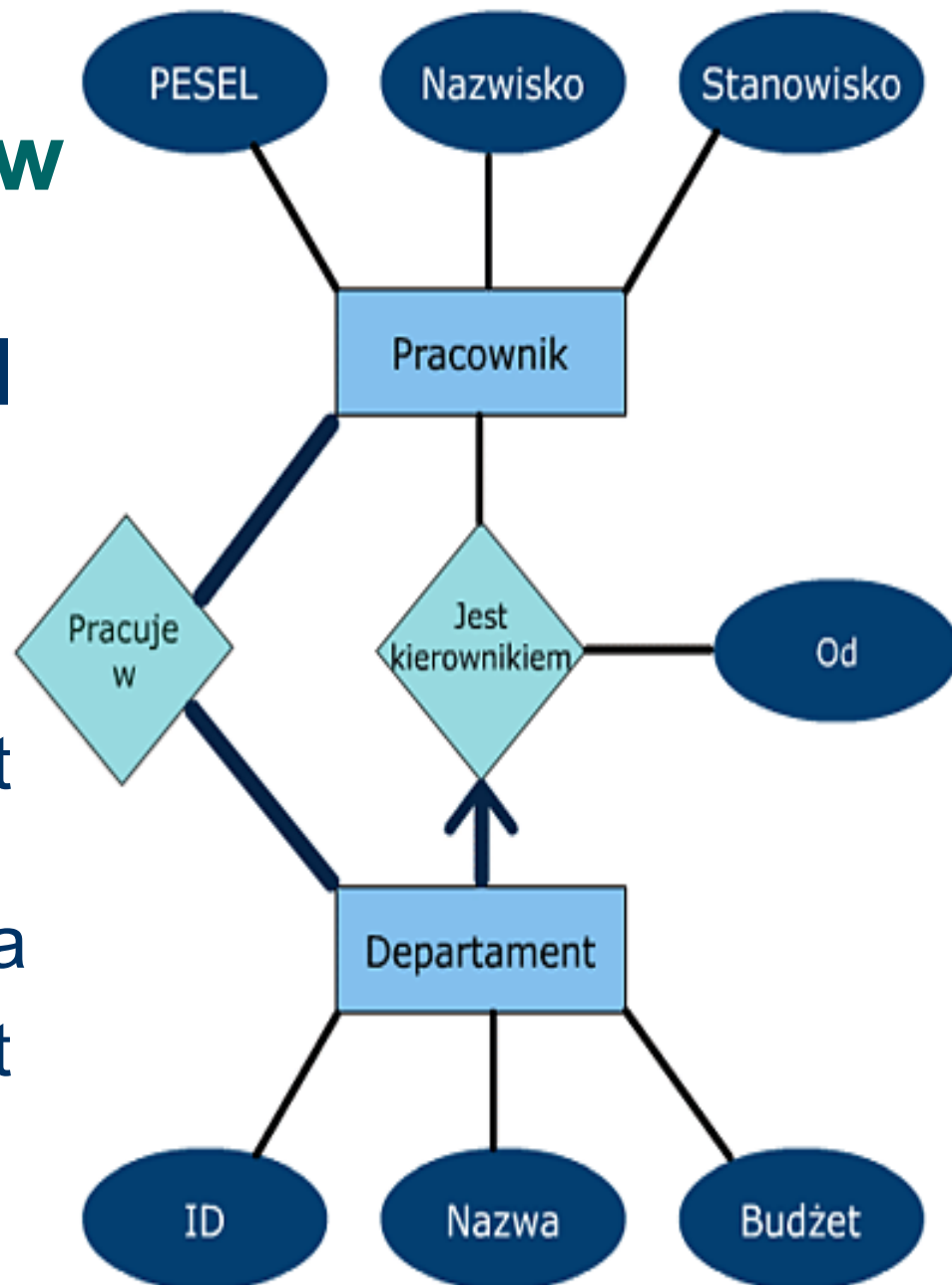


Więzy uczestnictwa

- Więzy uczestnictwa encji w związku oznaczają, że dla każdego egzemplarza encji istnieje egzemplarz związku go zawierający
- Związek jest więc obowiązkowy dla wszystkich egzemplarzy encji.
- Związek jest opcjonalny, gdy nie jest skojarzony z więzami uczestnictwa

Przykład więzów uczestnictwa

- Każdy pracownik pracuje w jakimś departamencie
- Każdy departament ma co najmniej jednego pracownika
- Każdy departament ma dokładnie jednego kierownika

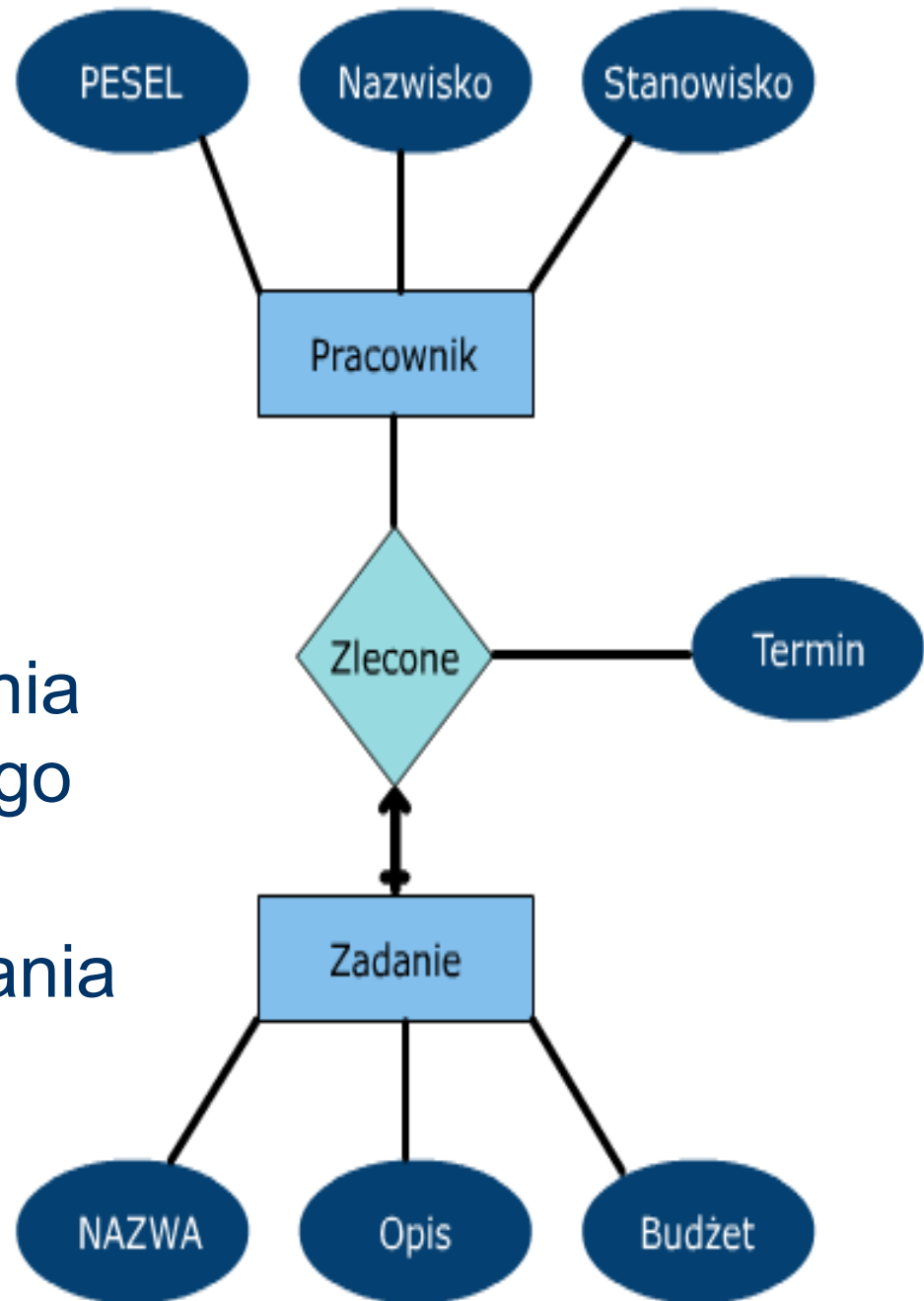


Encja zależna (słaba)

- **Encja zależna** to encja, której atrybuty nie wystarczają do jednoznacznej identyfikacji jej egzemplarza
- Do jego zidentyfikowania potrzebne są dodatkowo egzemplarze jednego lub więcej związków
- Inaczej mówiąc, encja *jest zależna* od związku lub związków albo, jeszcze inaczej, związek bądź związki identyfikują encję.

Przykład encji słabej

- Zadanie to encja zależna
- Sama *Nazwa* zadania nie wystarcza do jego pełnej identyfikacji
- Mogą być dwa zadania o tej samej nazwie zlecone różnym pracownikom.

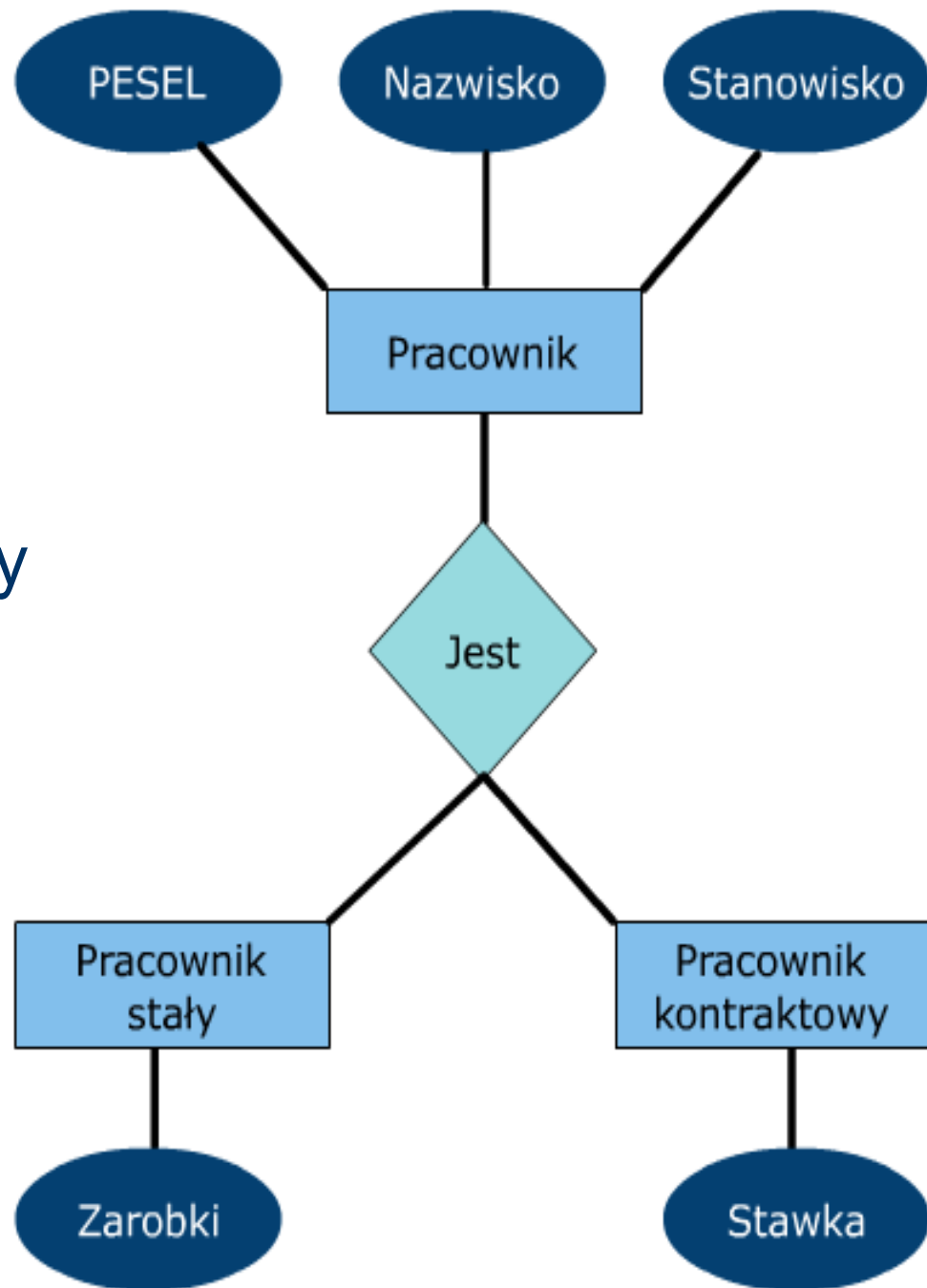


Hierarchia encji (klas)

- Hierarchia encji jest to zbiór encji i związków jedno-jednoznacznych tworzący hierarchię
- W dół hierarchii są to związki *specjalizacji*;
- W górę hierarchii są to związki *generalizacji*
- Hierarchia encji jest tworzona przez wielokrotne użycie związku jedno-jednoznacznego jest
- Nadencja/podencja (nadrzędna/podrzędna)
- Więzy pokrycia, więzy rozłączności

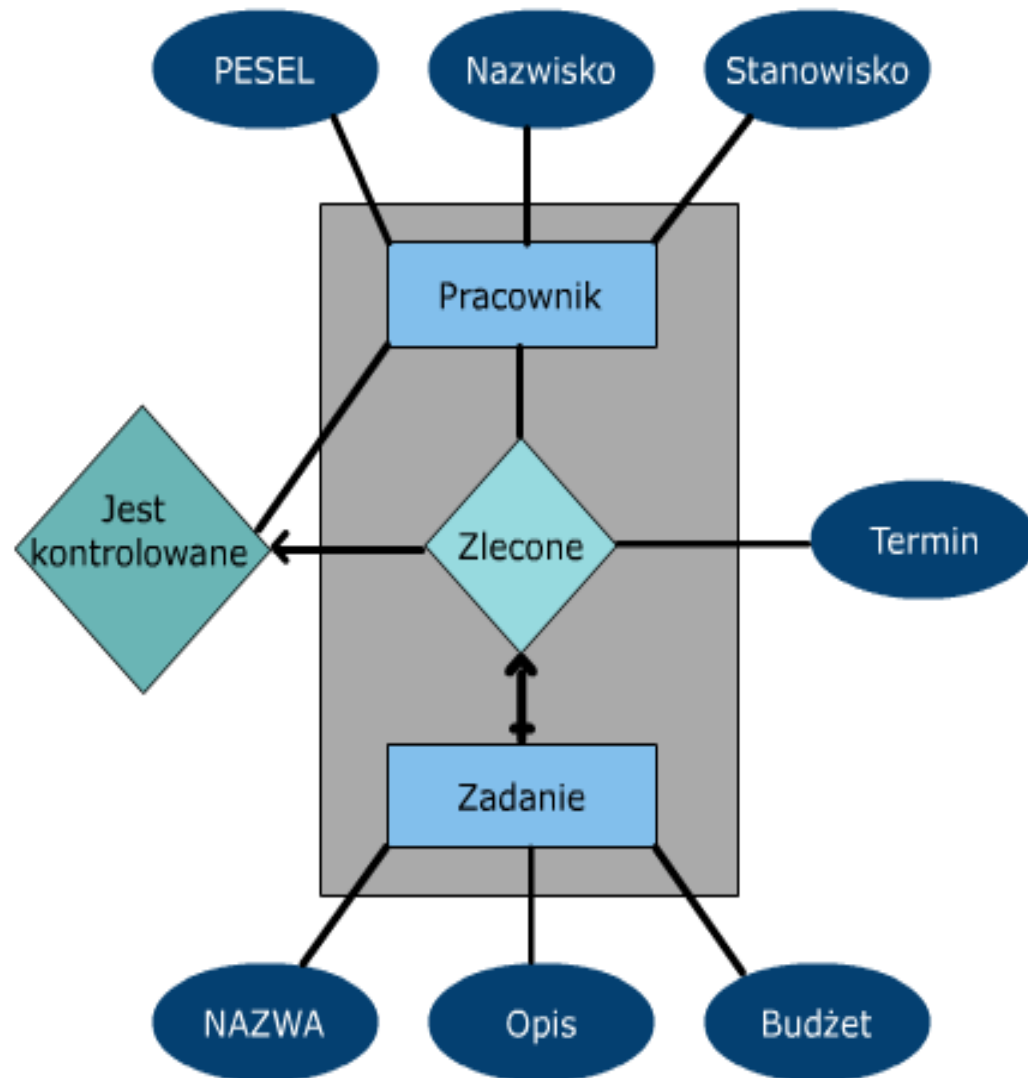
Przykład hierarchii

- Pokrycie: każdy Pracownik jest stały albo kontraktowy
- Rozłączność: nie ma pracowników jednocześnie stałych i kontraktowych



Agregacja = związek związków

- **Agregacja** oznacza sytuację, w której jeden związek jest argumentem innego związku.
- Nic dziwnego, skoro związek może mieć atrybut



Wiele pojęć można wyeliminować

- Związki wieloargumentowe: zastąpić encjami i związkami dwuargumentowymi
- Atrybuty związków: zastąpić związek encją i jej przypisać atrybut
- Związki związków: zastąpić związek encją i ją powiązać związkiem
- Każdy gracz ma więc encje i łączy nimi przeciwnika po związku

Bogactwo modelu Chena

- Precyzyjniejsze modelowanie pojęć semantycznych niż notacje wykorzystujące tylko związki binarne.
- Notacje oparte na związkach binarnych są bliższe schematom rzeczywistych baz danych,
- W szczególności umożliwiają automatyczne generowanie schematu bazy danych.
- Stanowią krok pośredni między precyzyjnymi modelami semantycznymi dziedzin zastosowań a schematami baz danych.

Bazy danych



07:

Uproszczone ERD –
związki binarne

Krzysztof Stencel

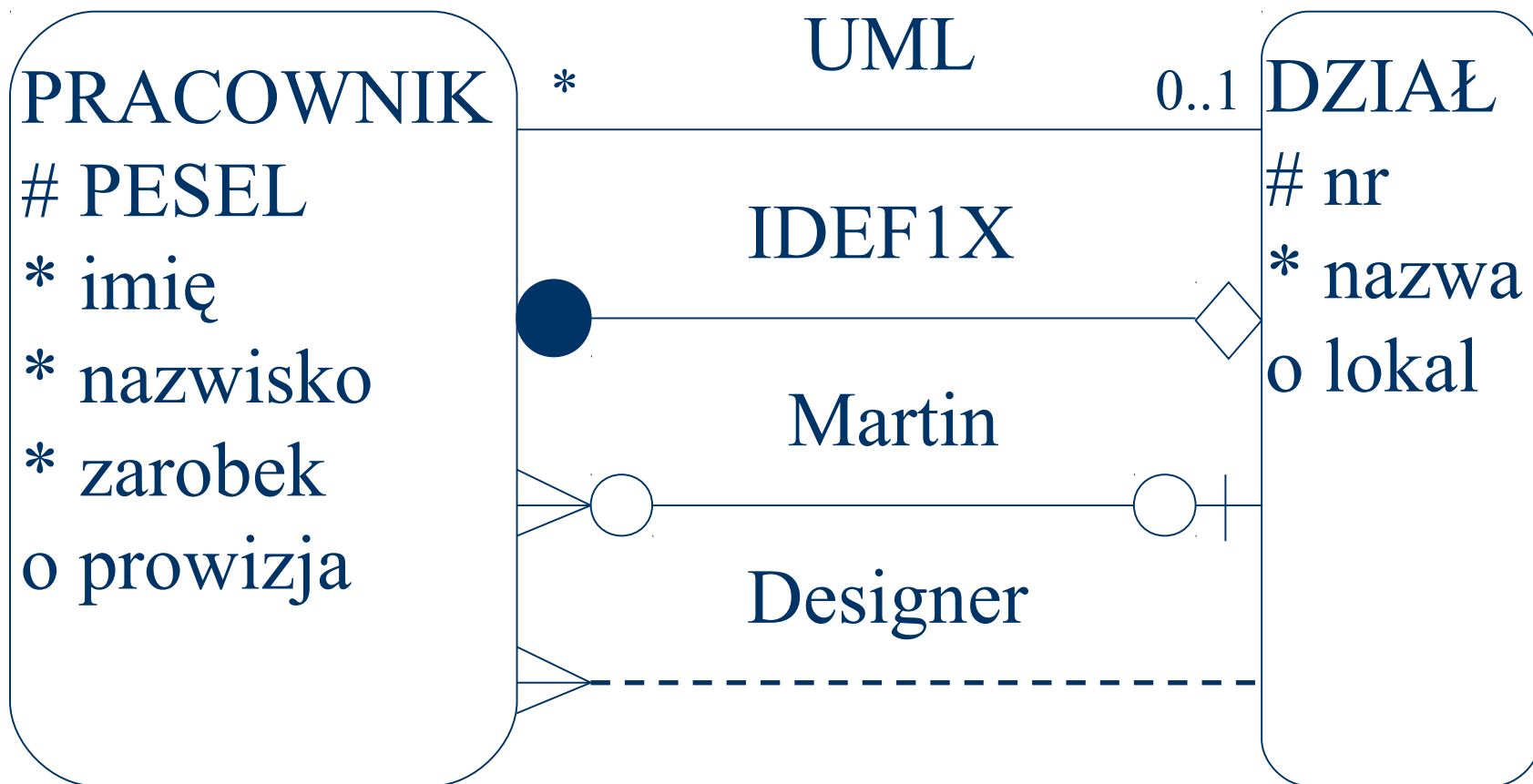
Model P. Chena jest super, ale...

- Niestety jest zbyt bogaty i zbyt skomplikowany
- Nie rozumieją go niektórzy studenci informatyki, nie mówiąc już o przemyśle:
 - Pamiętamy bowiem, że jest on przeznaczony też dla czytelników biznesowych
- W praktyce stosuje się więc uproszczoną metodykę modelowania związków encji
 - są tylko związki dwuargumentowe między encjami nie mające atrybutów.

Dobrze się to sprawdza w praktyce

- Prostszy model pojęciowy jest lepiej przyswajalny przez wszystkich
- Notacji jest cała masa, ale wszystkie mają:
 - Symbol encji w postaci zamkniętego kształtu
 - Atrybuty zapisywane we wnętrzu symbolu encji
 - Związki jako kreski łączące encje z różnymi dekoracjami
- Są one jednak bardzo podobne.
- Z badań archeologicznych wynika, że wszystkie rozumiał już *Homo erectus*.

Bogactwo symboliki



Wybór notacji

- Podyktowany narzędziem – często tak jest
- Główna trudność to związki (ale do obejścia)

- Opcjonalny

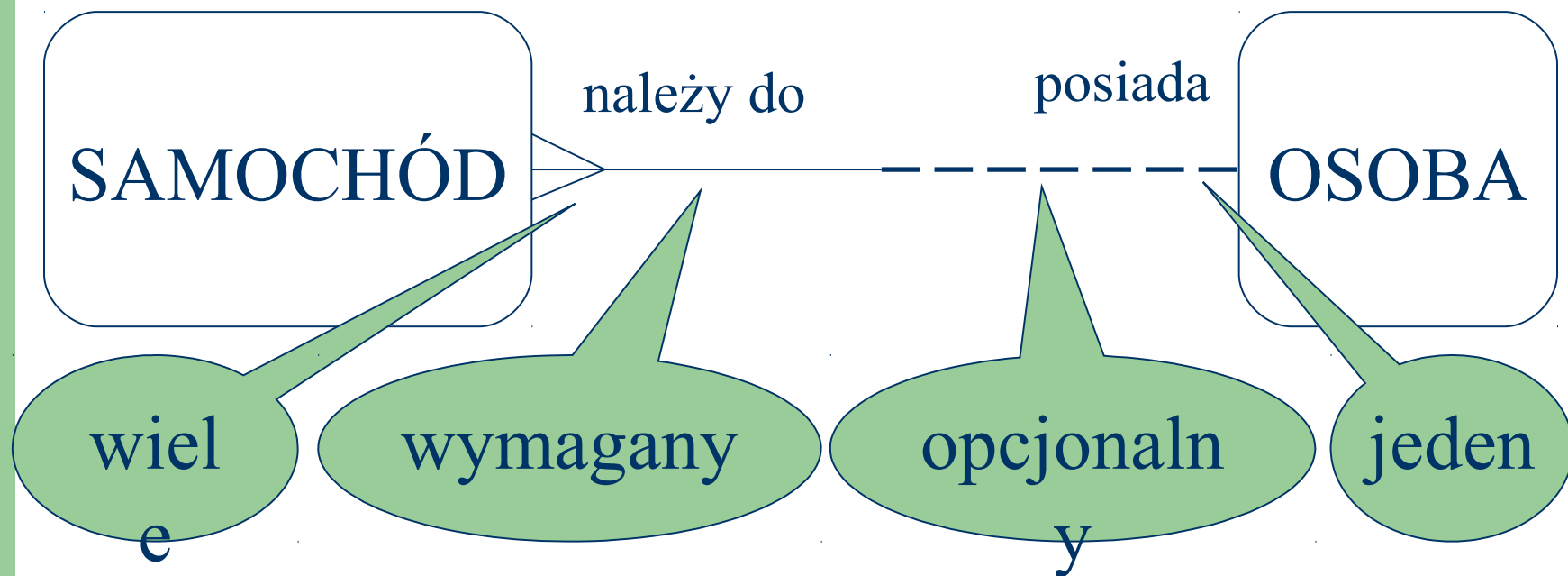
- Wymagany

- Jeden

- Wiele

 _____

Przykład związku

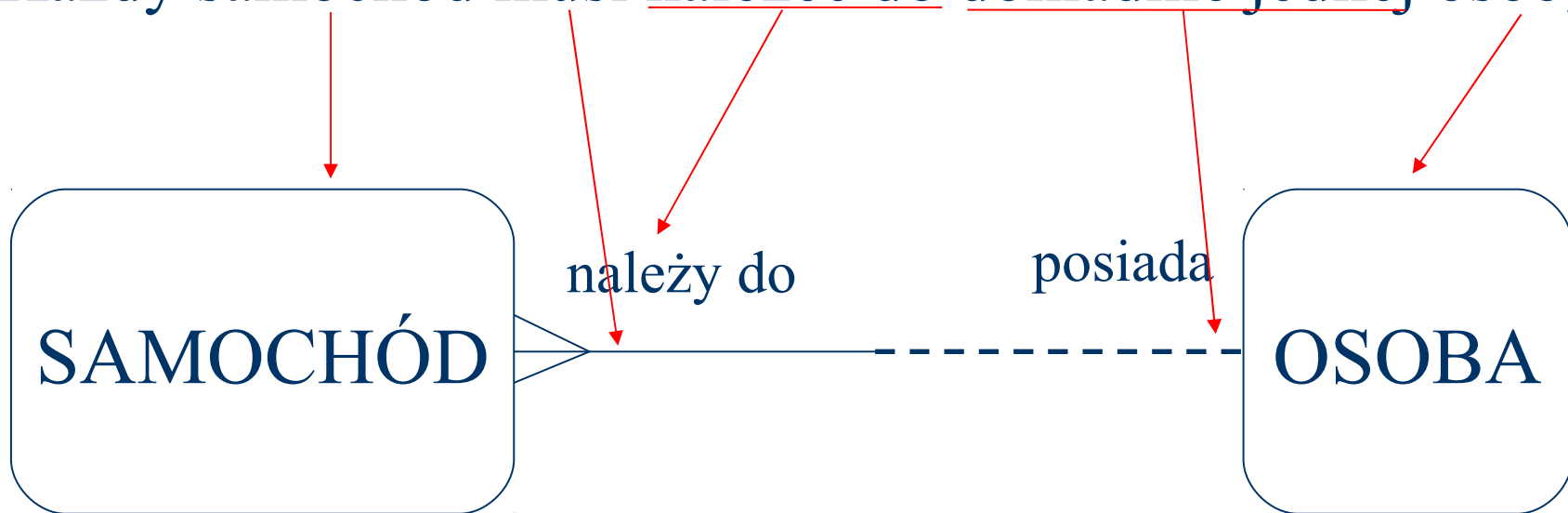


Każdy samochód należy do dokładnie jednej osoby.

Każda osoba może posiadać dowolnie wiele samochodów.

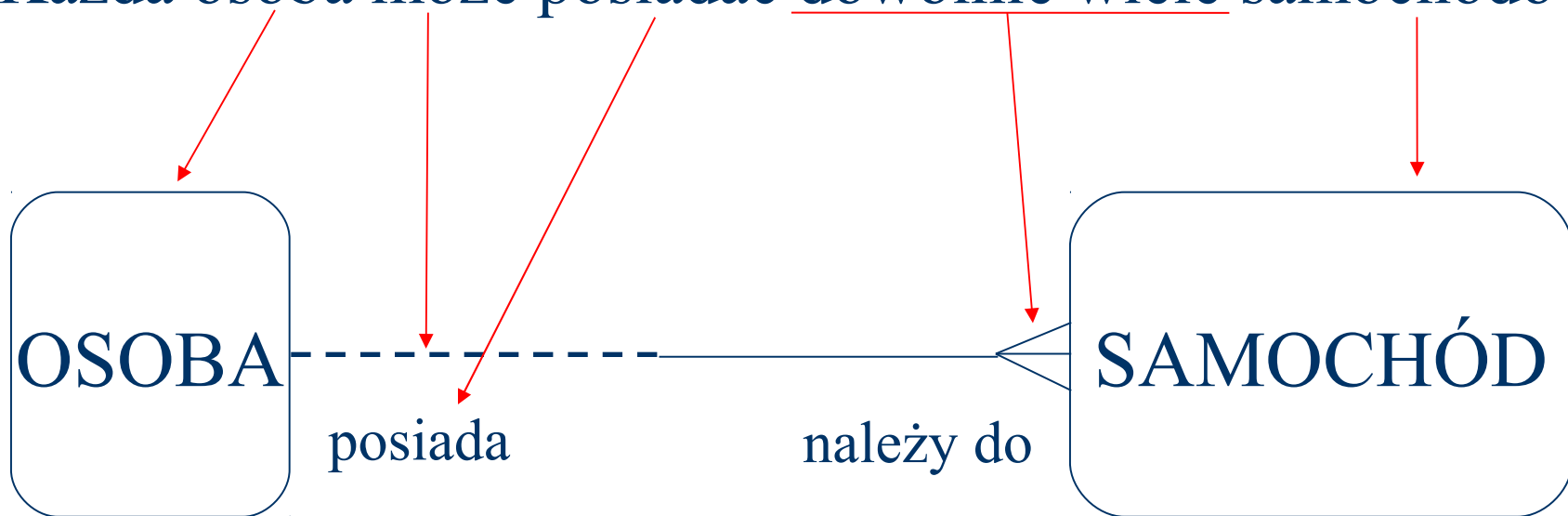
Czytanie związku dla opornych

Każdy samochód musi należać do dokładnie jednej osoby.



Czytanie związku dla opornych

Każda osoba może posiadać dowolnie wiele samochodów.



Uproszczona klasyfikacja związków



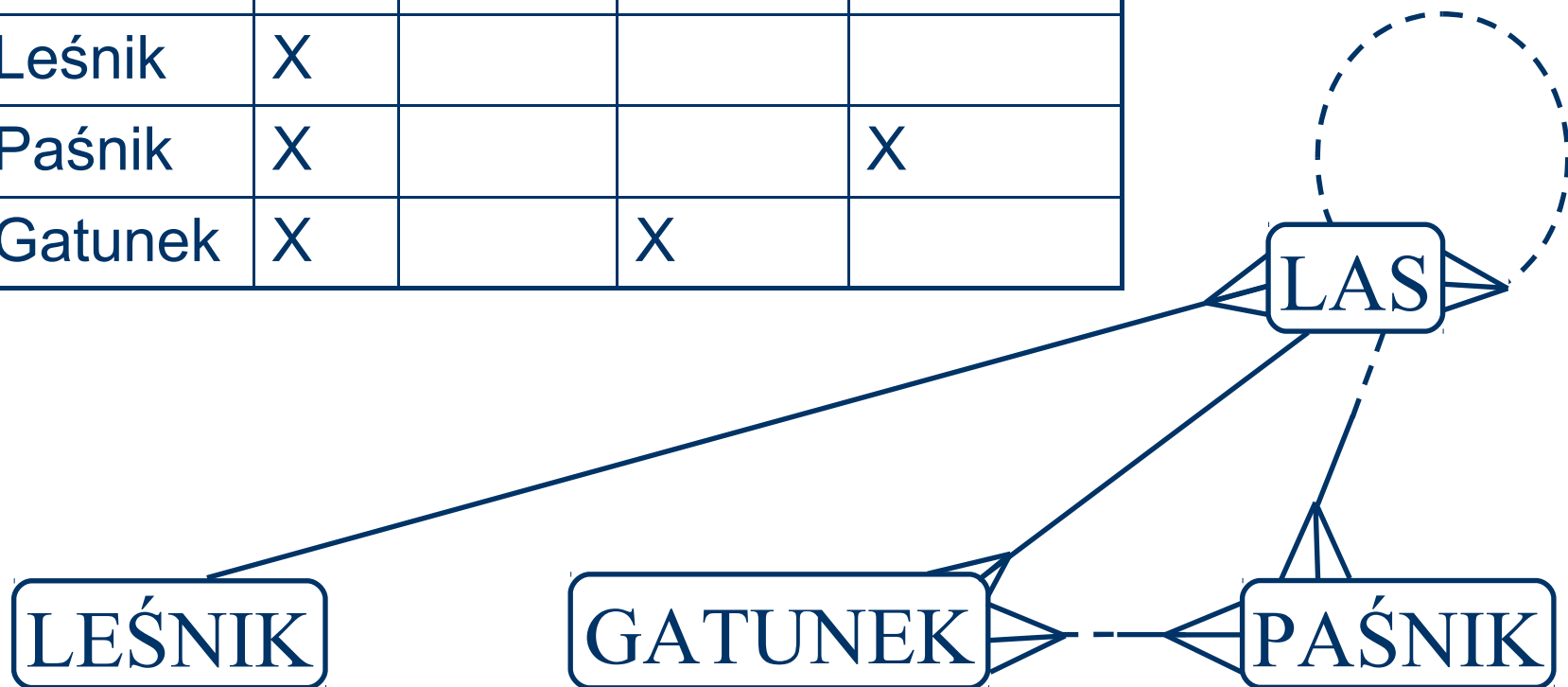
Jeden-do-jeden to zwykle zapis chwilowej sytuacji

Dodawanie związku

1. Stwierdź jego istnienie
2. Nazwij go (u nas: daj dwie nazwy)
3. Określ liczebność każdego końca
4. Określ wymagalność każdego końca
5. Przeczytaj go głośno w celu weryfikacji
6. Każ też czytać innym (zwłaszcza uczestnikom biznesowym)

Przydatna technika: macierz encja-encja

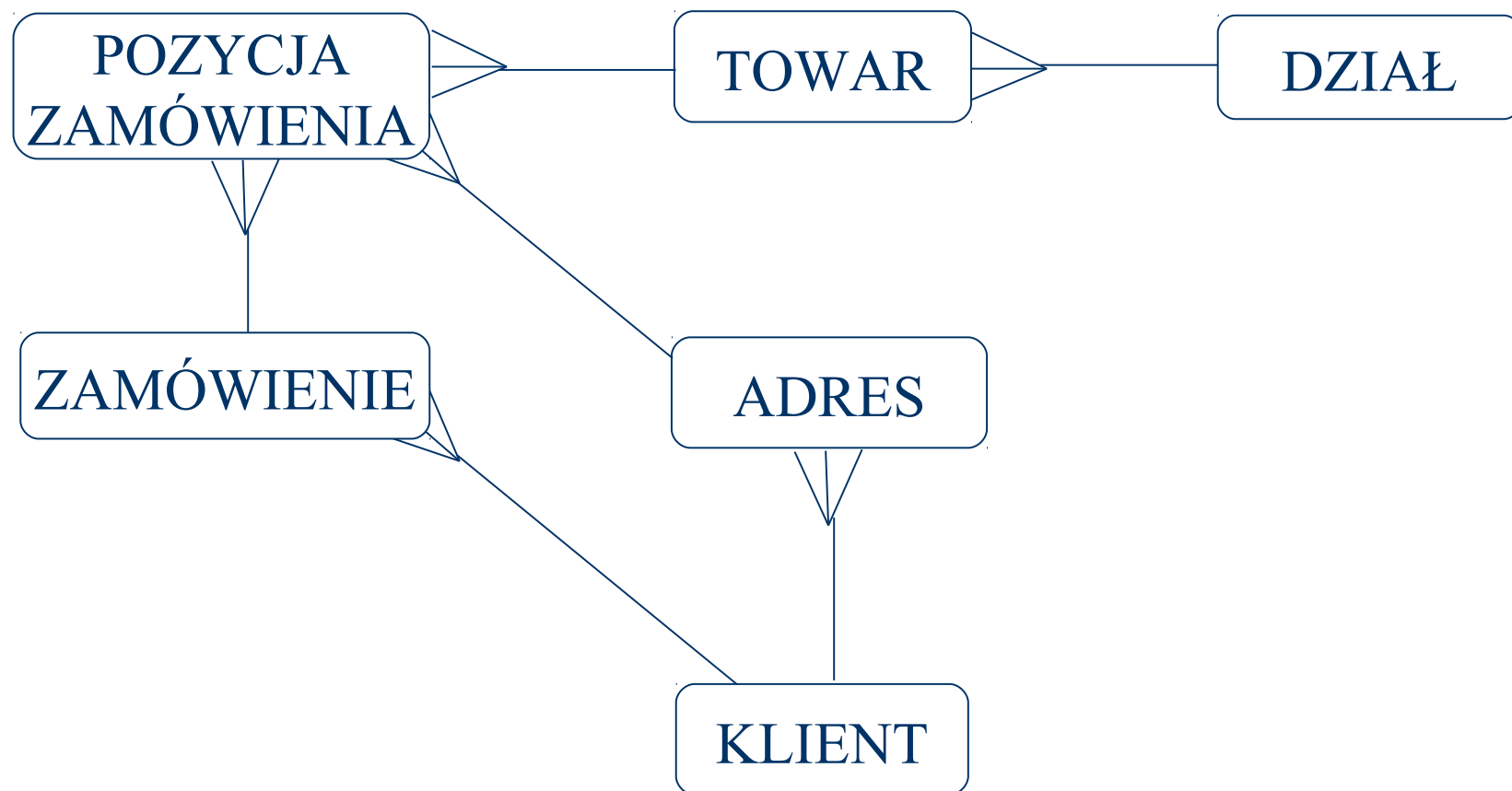
	Las	Leśnik	Paśnik	Gatunek
Las	X	X	X	X
Leśnik	X			
Paśnik	X			X
Gatunek	X		X	



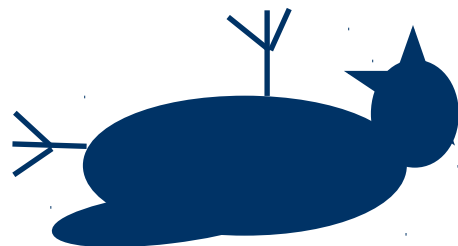
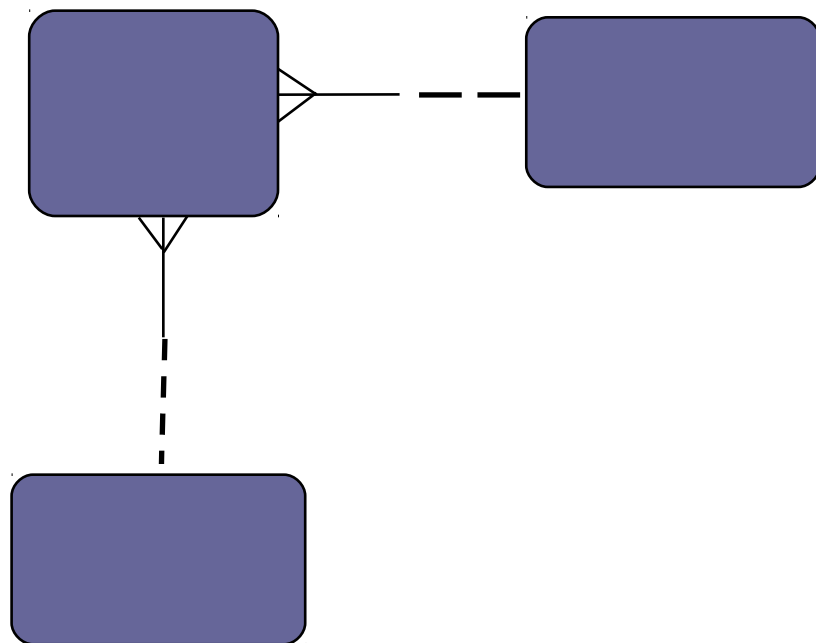
Układ diagramu

- Strony „wiele” związków zwrócone w jedną stronę, np. do góry i lewo lub skosem do góry i lewo
- To pomaga czytać diagram, bo wskazuje tzw. *encje referencyjne*, czyli po stronie „najbardziej jeden”, czyli tych najważniejszych
- Związek jeden-do-wiele bywa bowiem też nazywany *master-detail* (ogół-szczegół)
- Tworzą się też łatwe do zapamiętania wzorce

Układ diagramu – przykład



Układ diagramu – wyjaśnienie



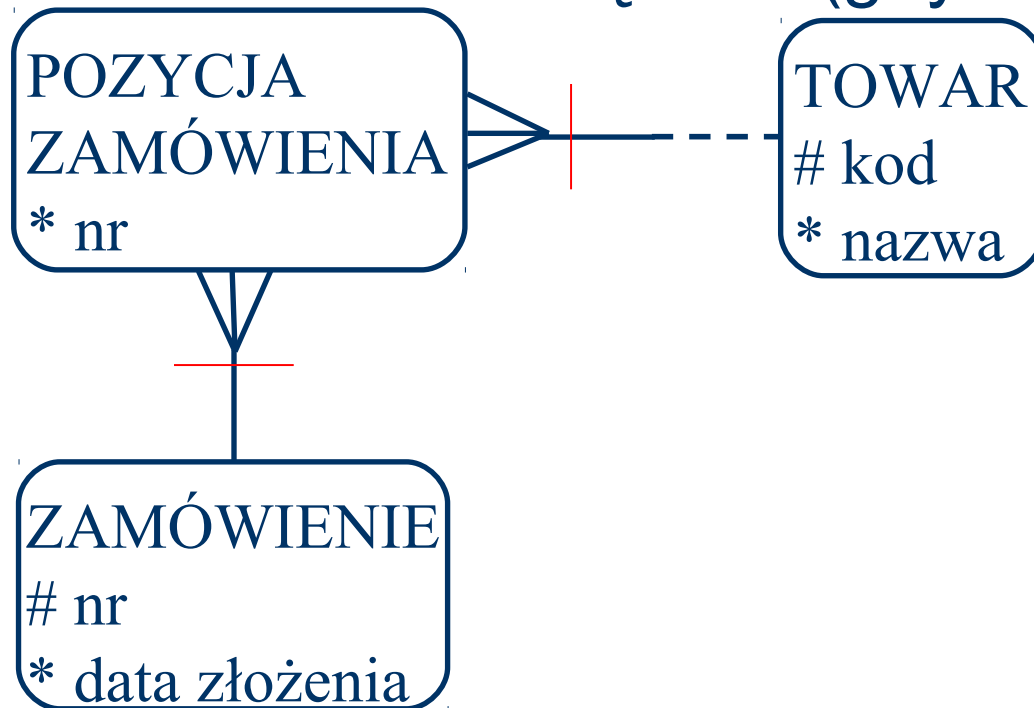
Zdechłe ptaszki lecą na wschód!

Atrybuty

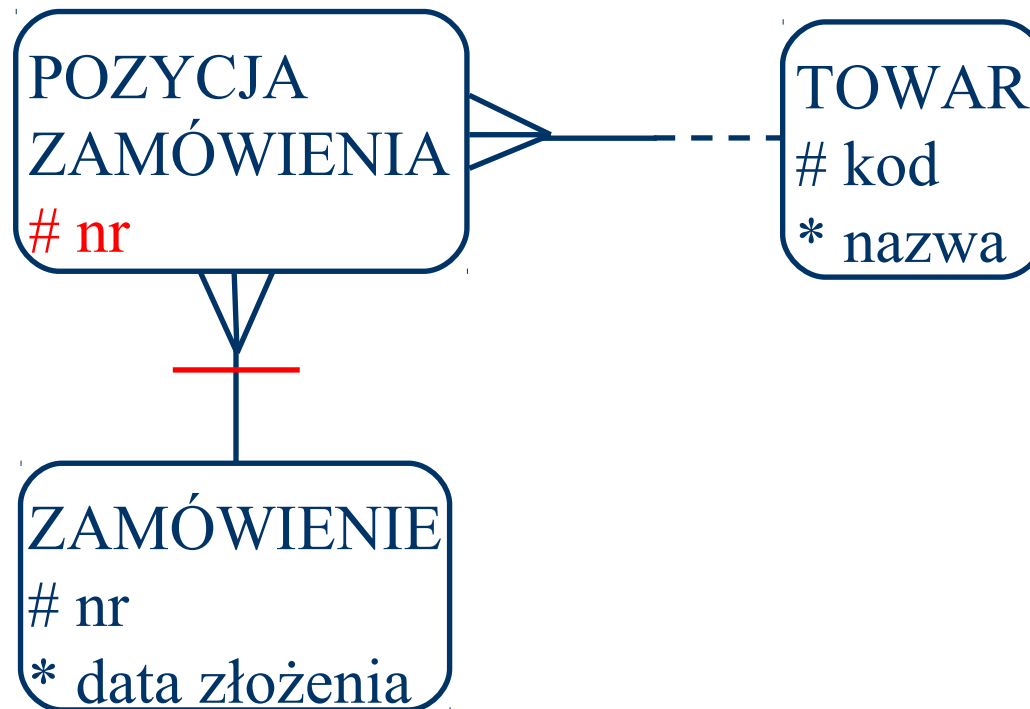
- Opcjonalne/wymagane/kluczowe (o/*/#)
- Czy są nierozkładalne?
 - Adres → Ulica, NrDomu, Miejscowość,...
- Czy są jednokrotne?
 - Pozycja zamówienia → nowa encja
- Czy mają atrybuty?
 - Recenzja filmu → nowa encja z treścią, autorem,...
- Czy są wyliczane?
 - Wartość zamówienia → wyliczana z pozycji i rabatu

Identyfikator

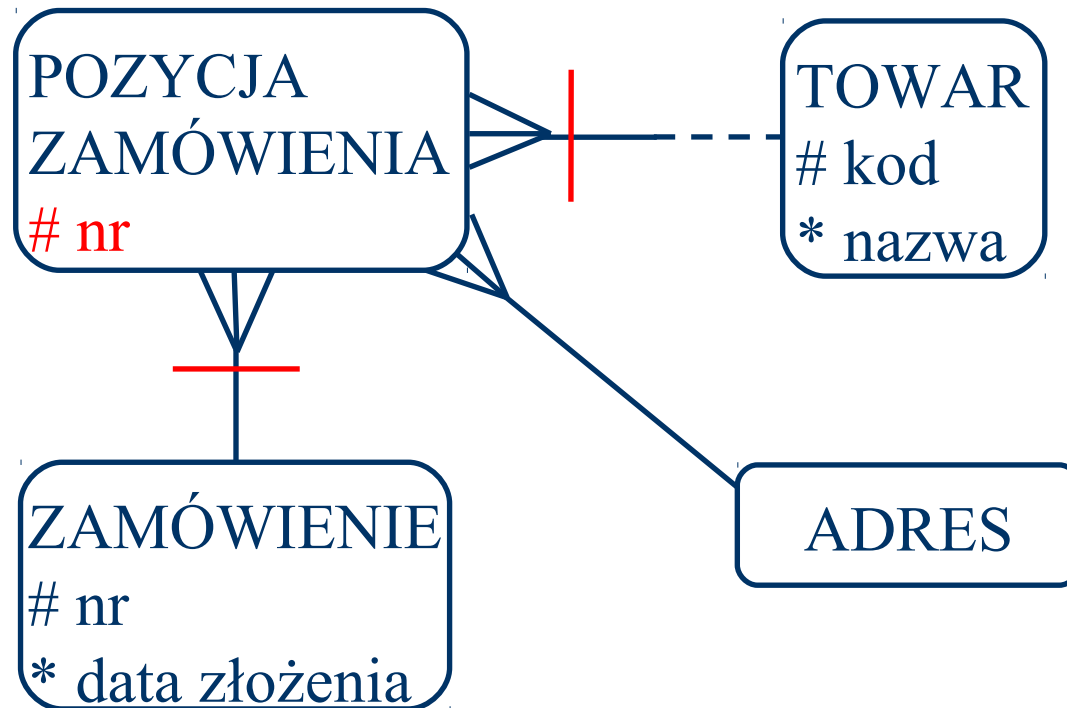
- Pewien podzbiór atrybutów
- Plus ewentualnie związków (gdy encja słaba)



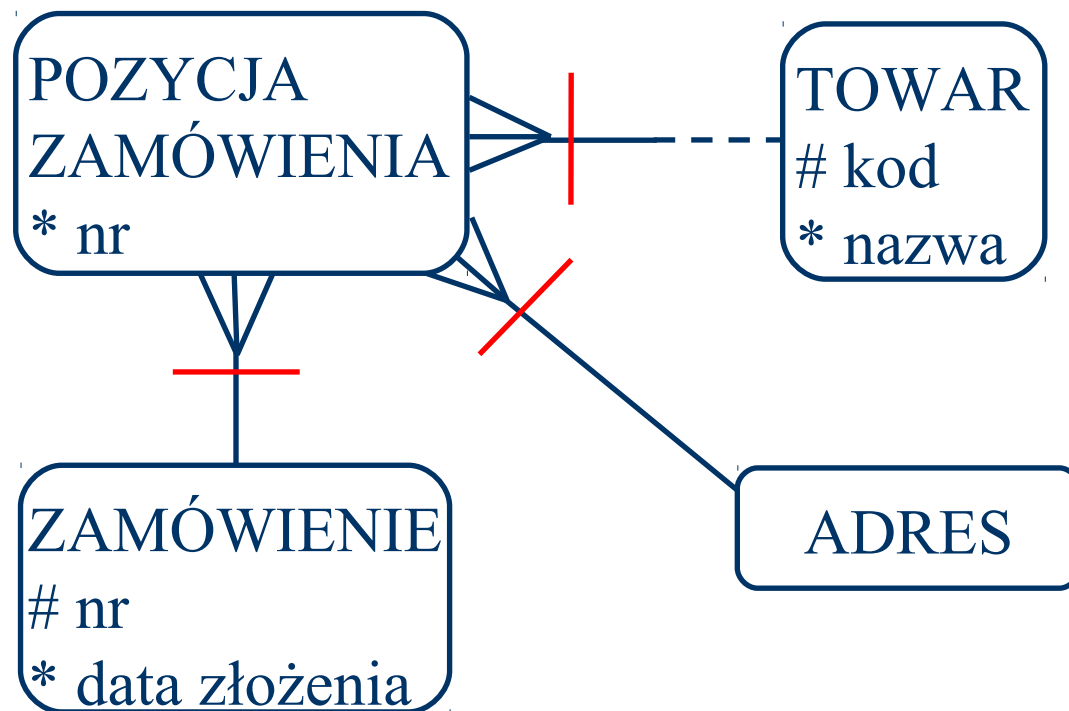
Identyfikator mieszany



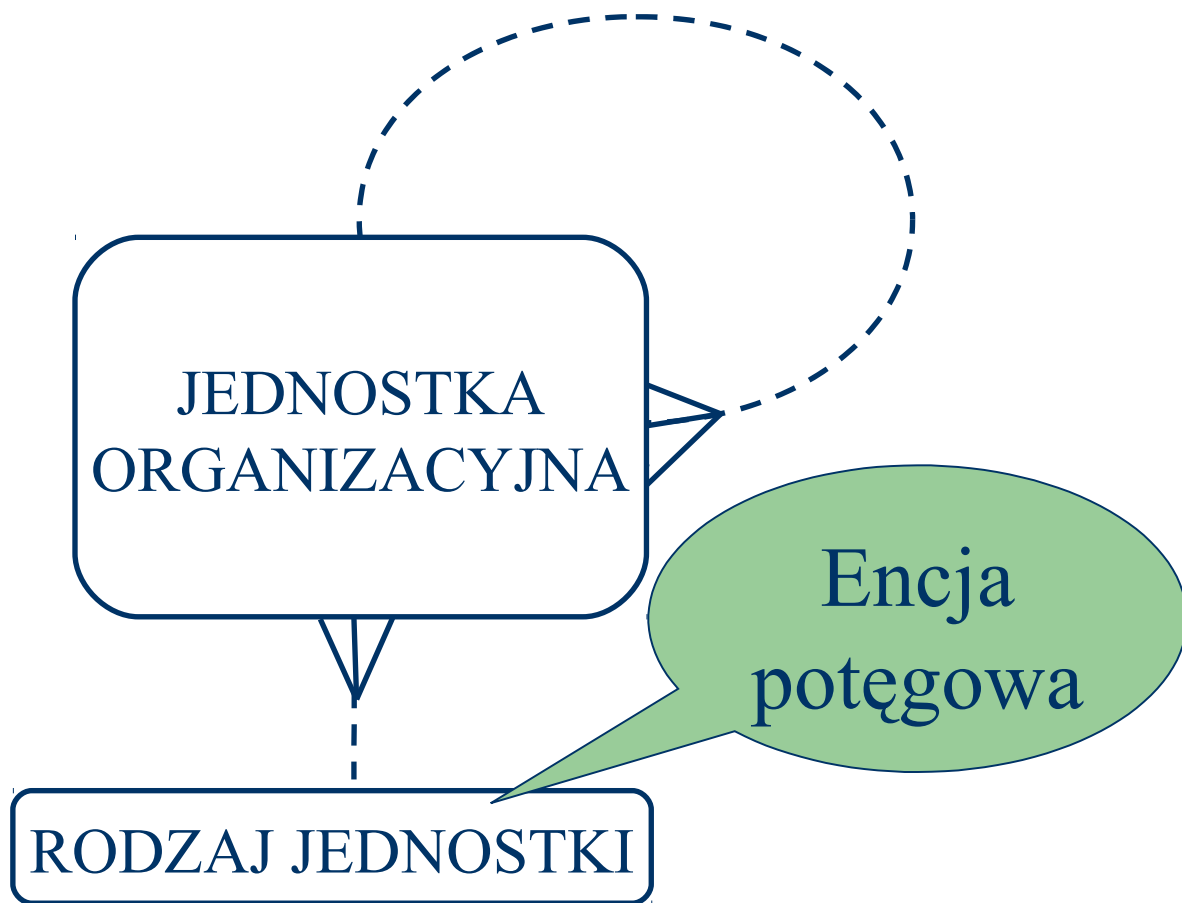
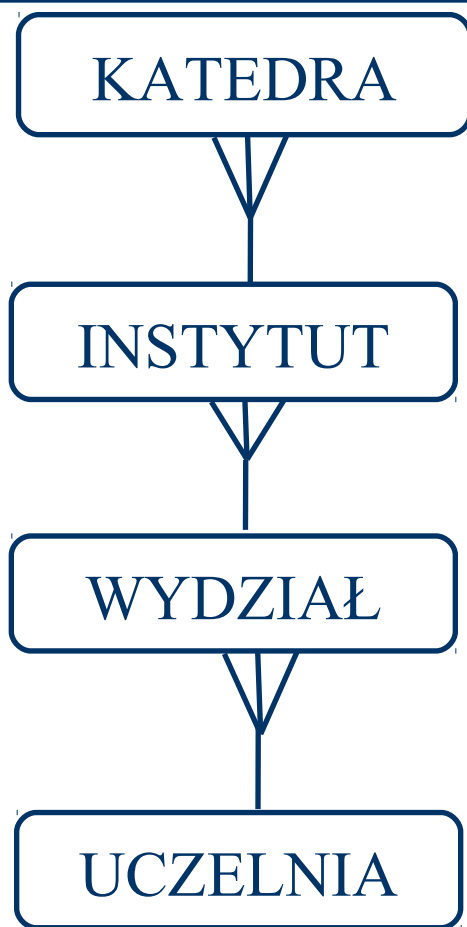
Gdy pozycji na jeden towar jest wiele...



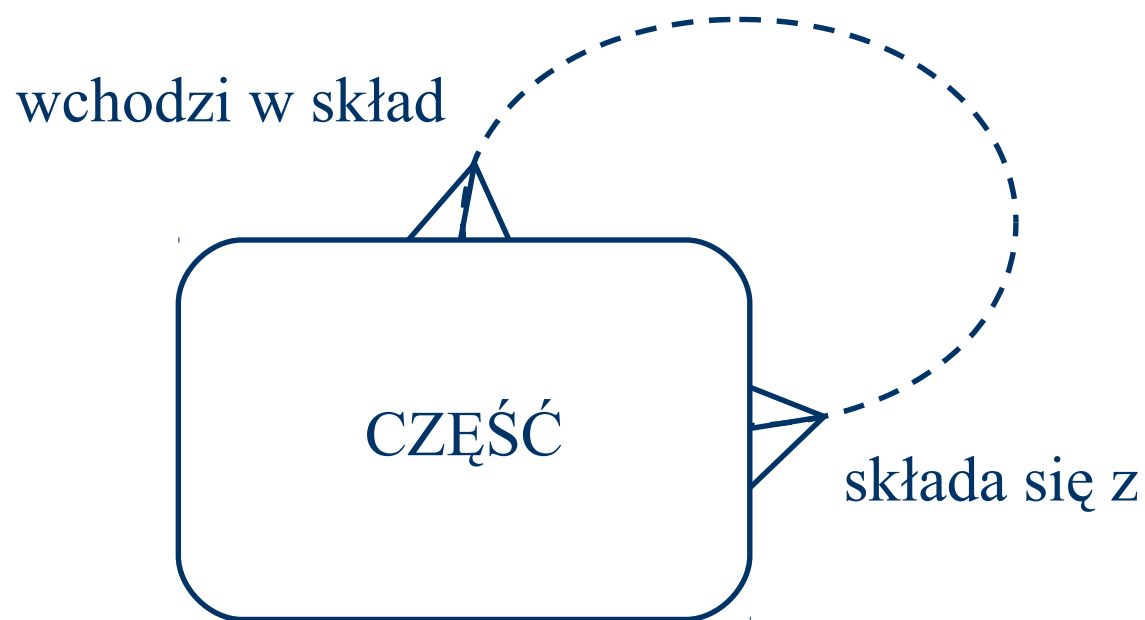
Lub też...



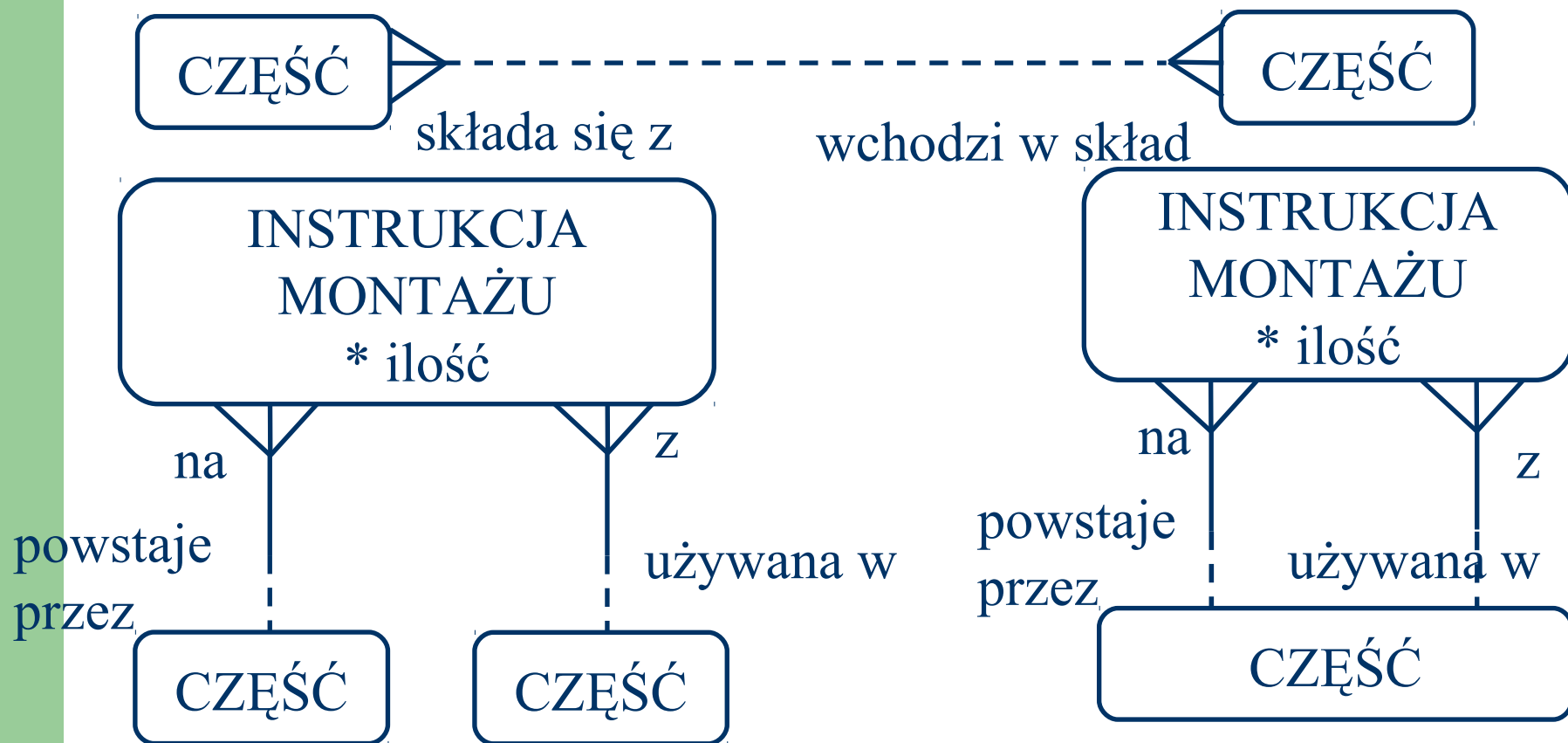
Hierarchie



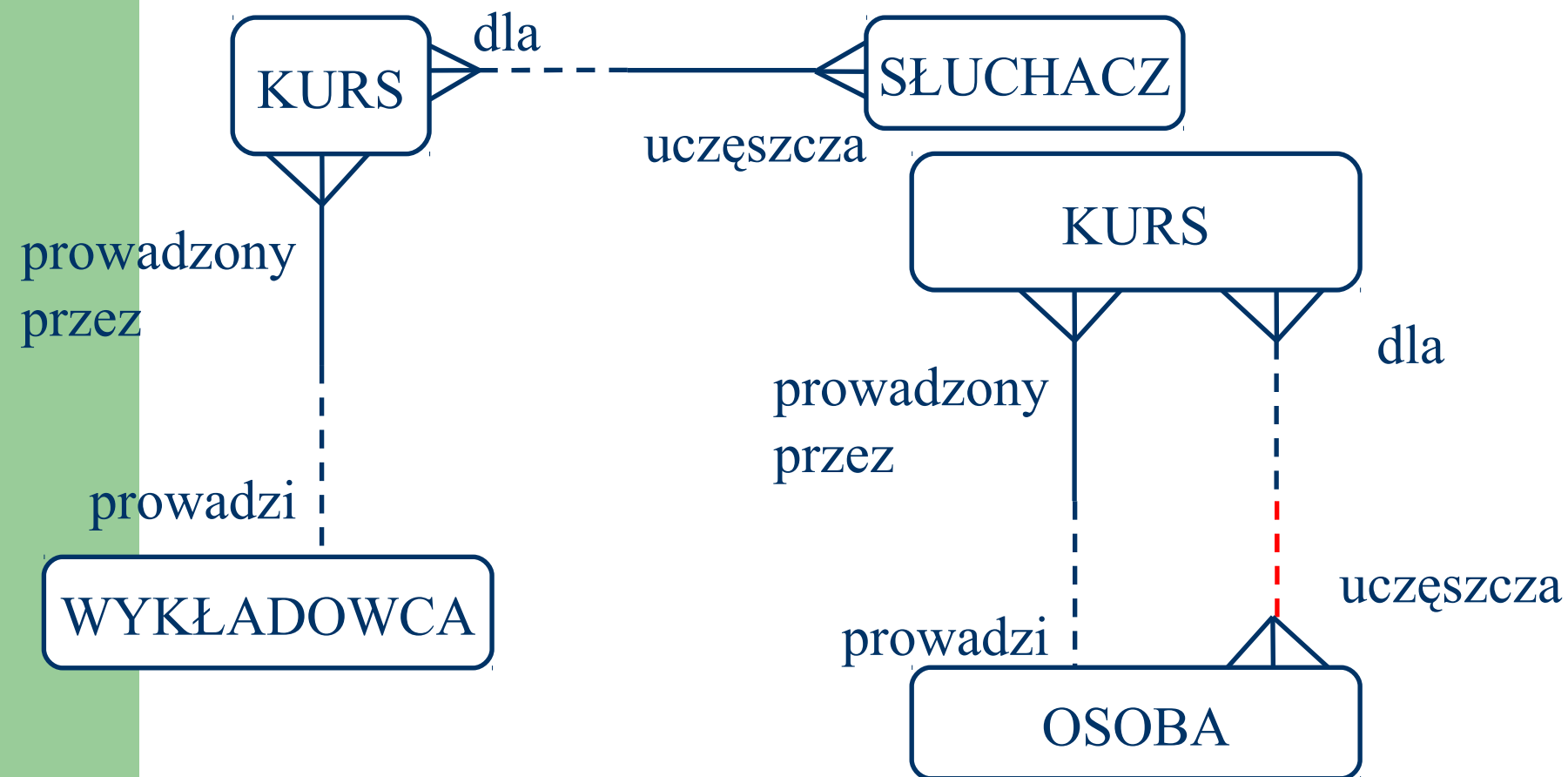
Graf (sieć)



Związek w grafie ma zwykle atrybuty



Role



Bazy danych



08:

ERD – podencje, łuki i
pułapki

Krzysztof Stencel

Hierarchia encji

OSOBA

PESEL

* imię

* nazwisko

* data urodzenia

PRACOWNIK

* zarobek

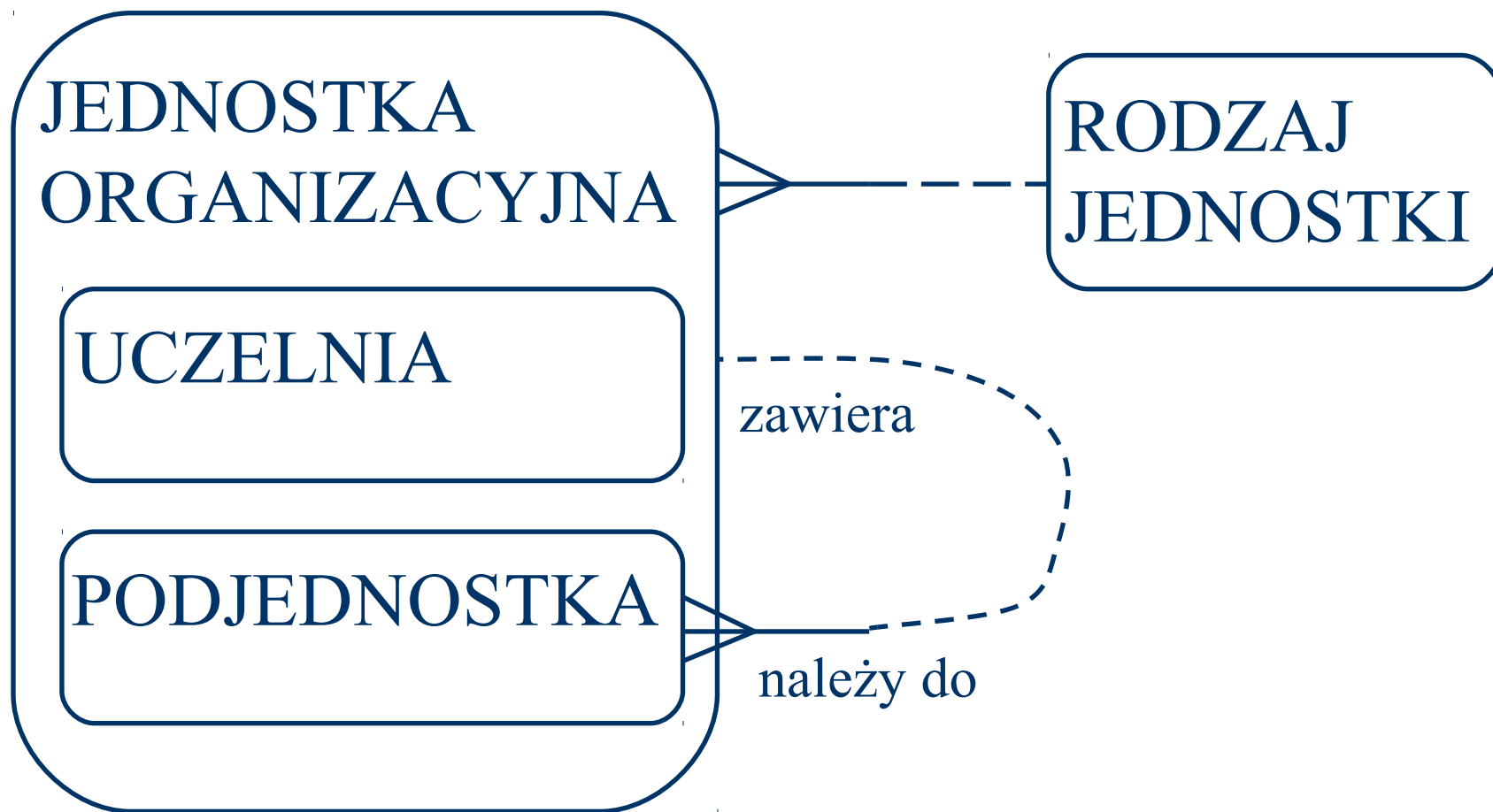
* stanowisko

STUDENT

* nr albumu

* rok studiów

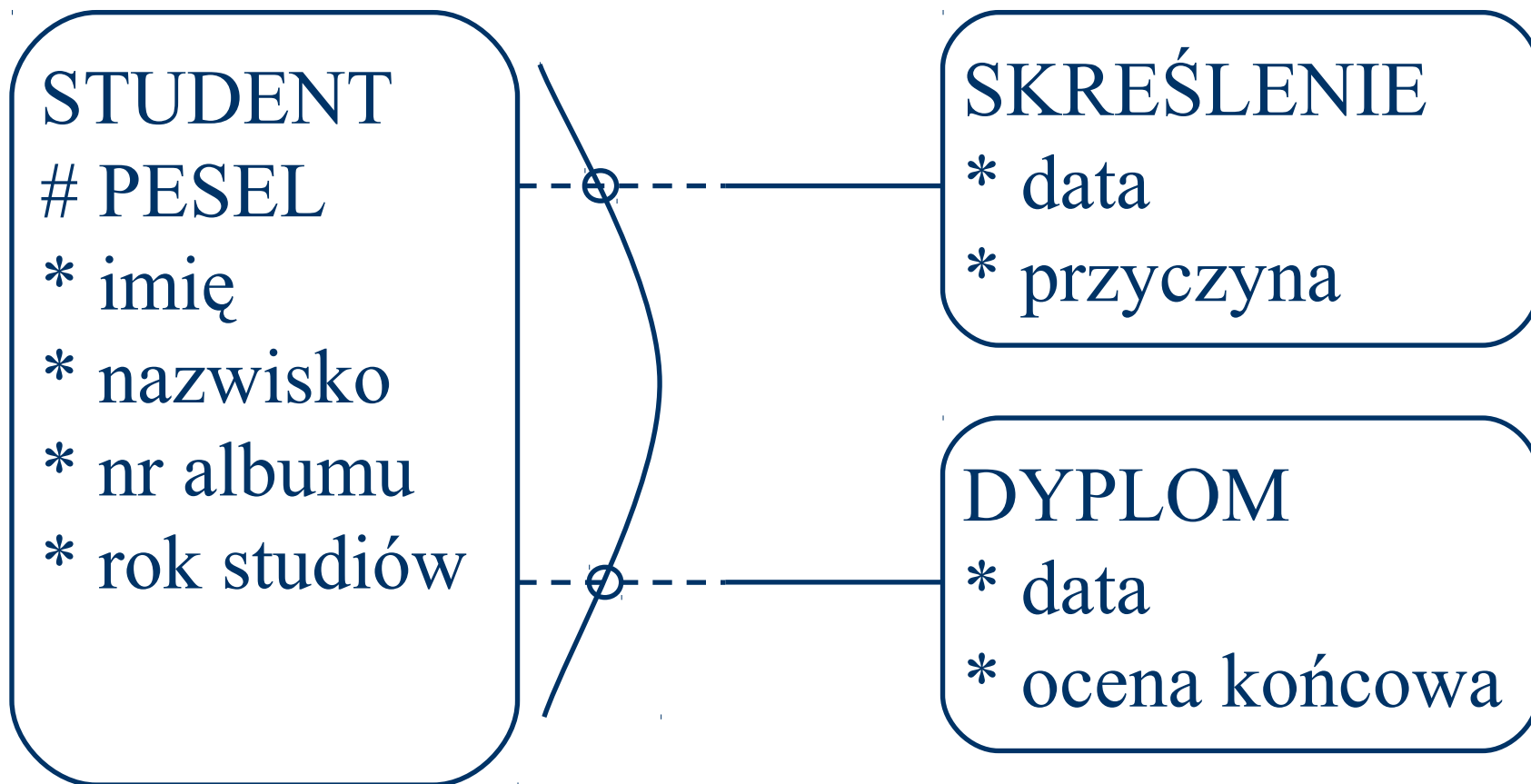
Związki idą tam, gdzie trzeba (inny sposób na drzewo)



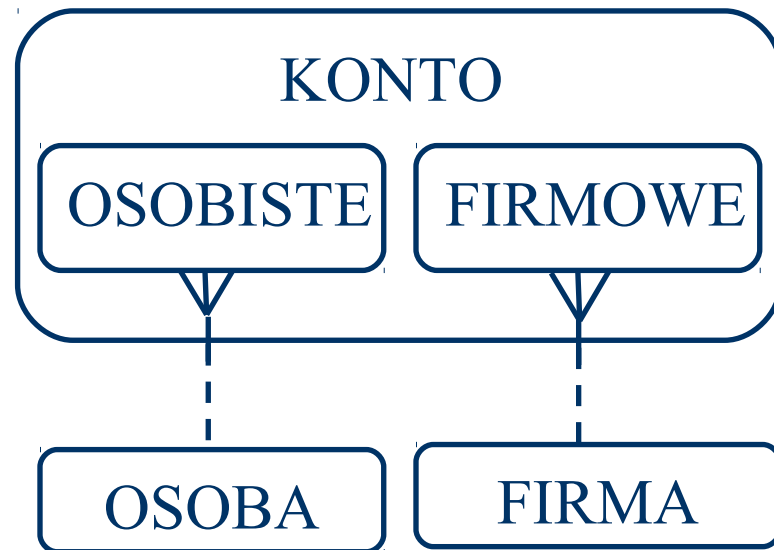
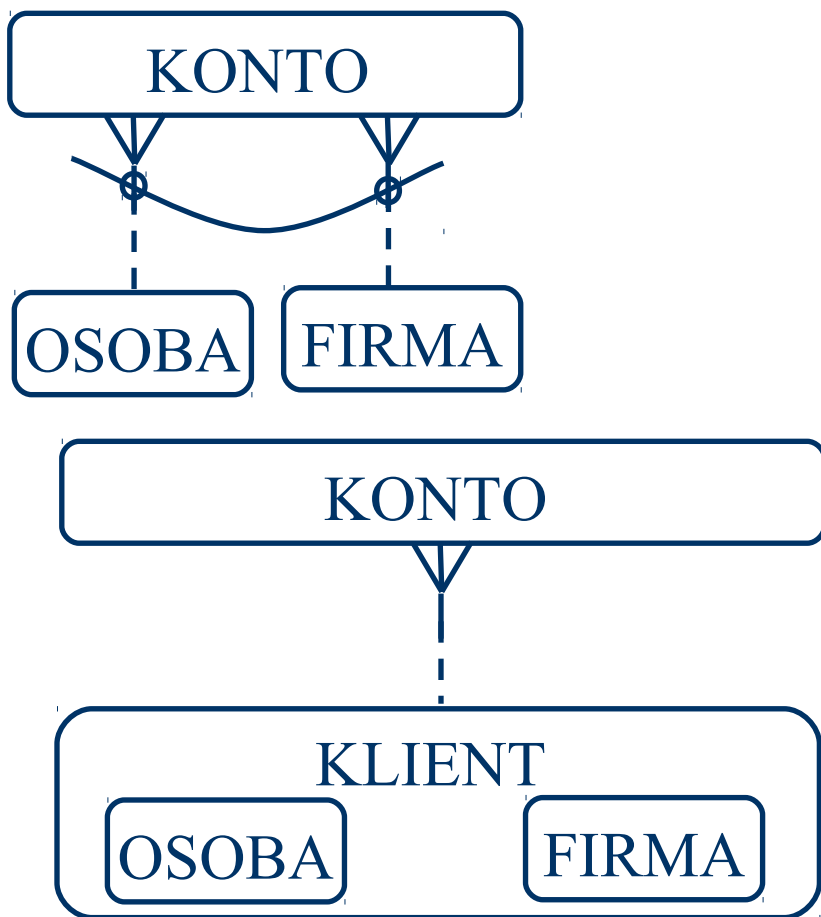
Elastyczność?

- Brak: tu zawsze pokrycie, zawsze rozłączność
- A przecież jest (brak pokrycia i rozłączności):
 - Asystent stażysta (student i pracownik w jednym)
 - Recenzent zewnętrzny (ani student ani pracownik)
 - Doktorant (i może jednocześnie pracownik)
- Jeszcze trudniej (tzw. *role wielokrotne*)
 - Student dwóch wydziałów i pracownik trzeciego
 - Pracownik dwóch wydziałów (częste)
 - Były student trzech wydziałów
- Itd.

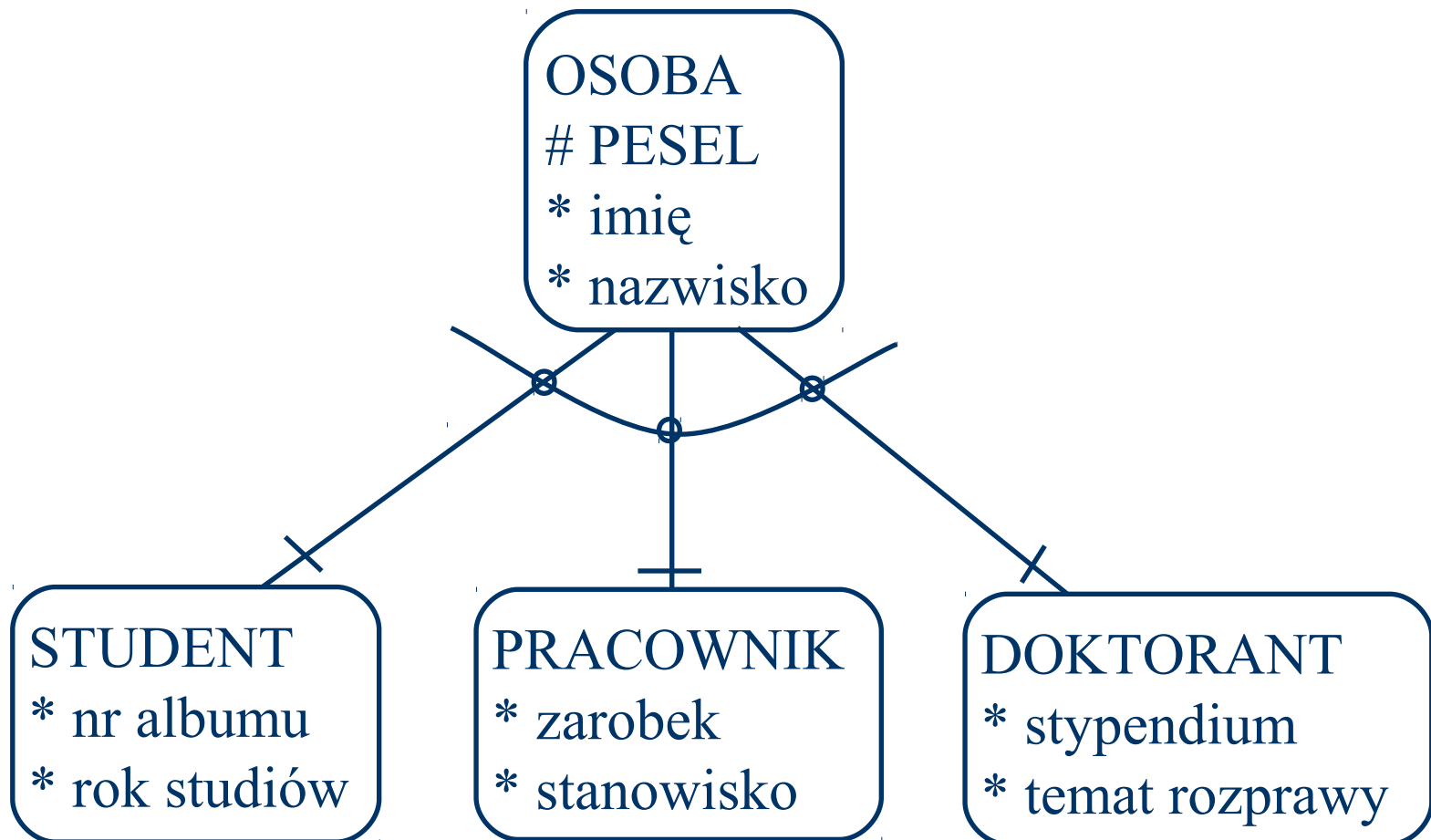
Łuki wykluczające



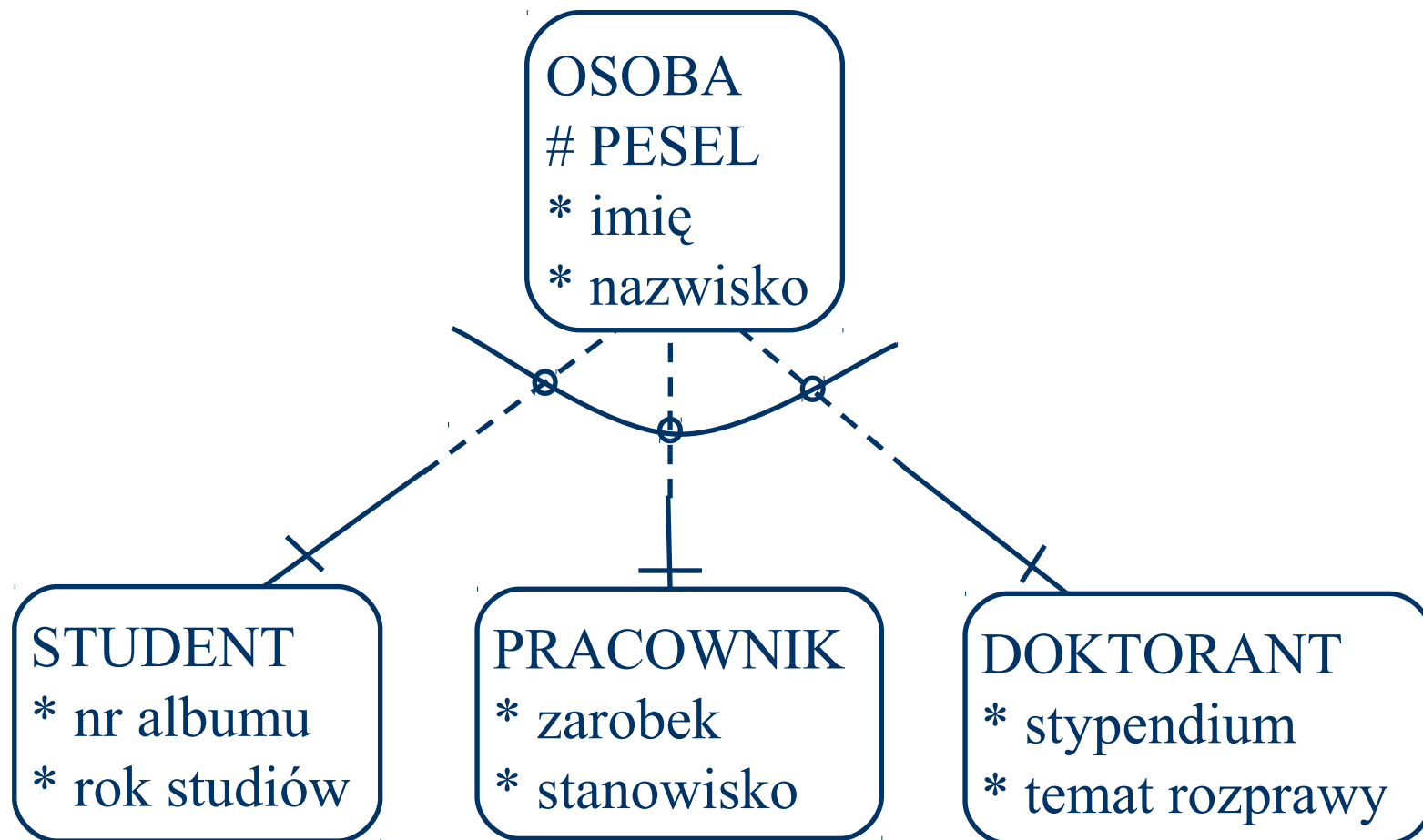
Wiele możliwości modelowania wykluczania



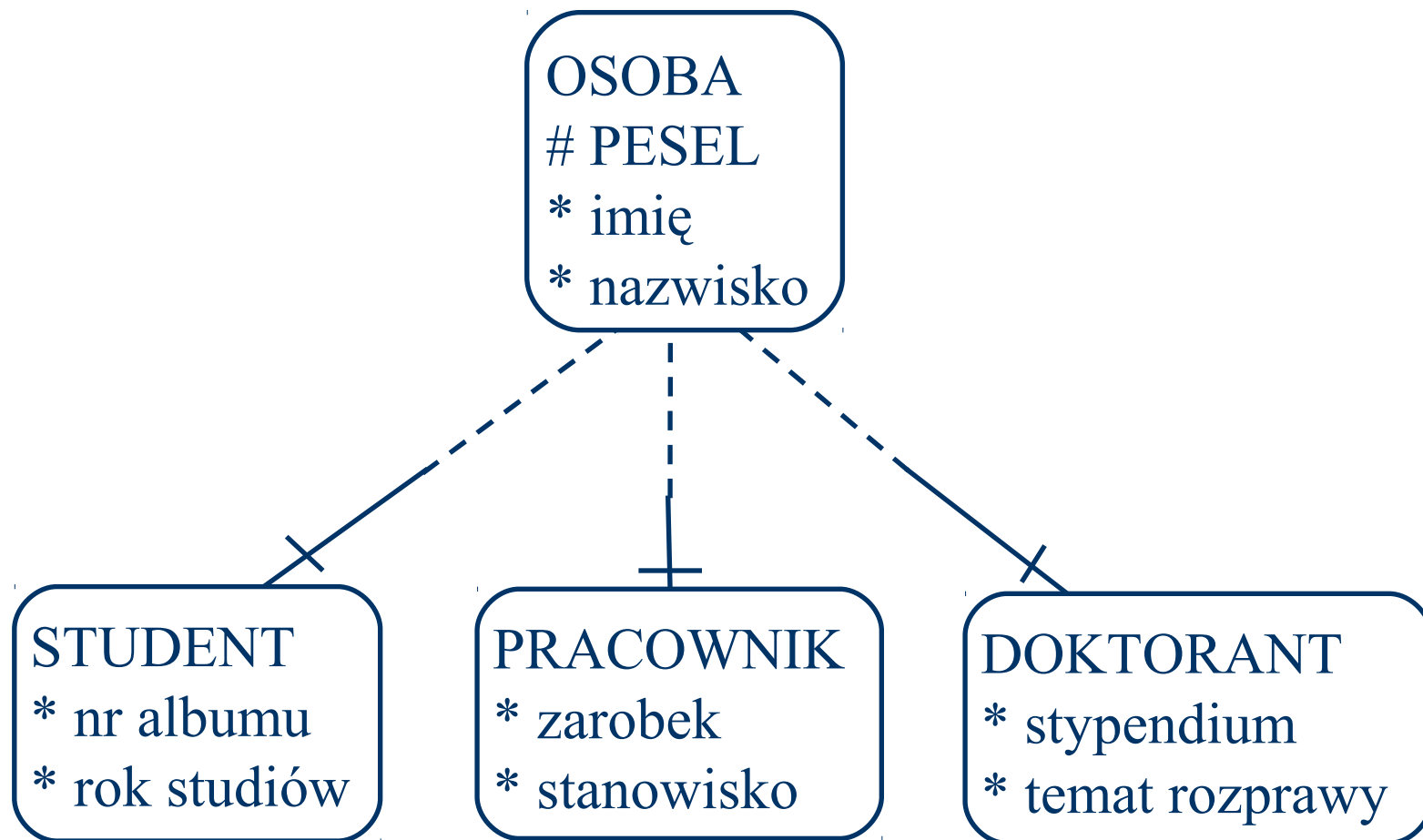
Łuki prowadzą do elastyczniejszej hierarchii... (na razie to samo)



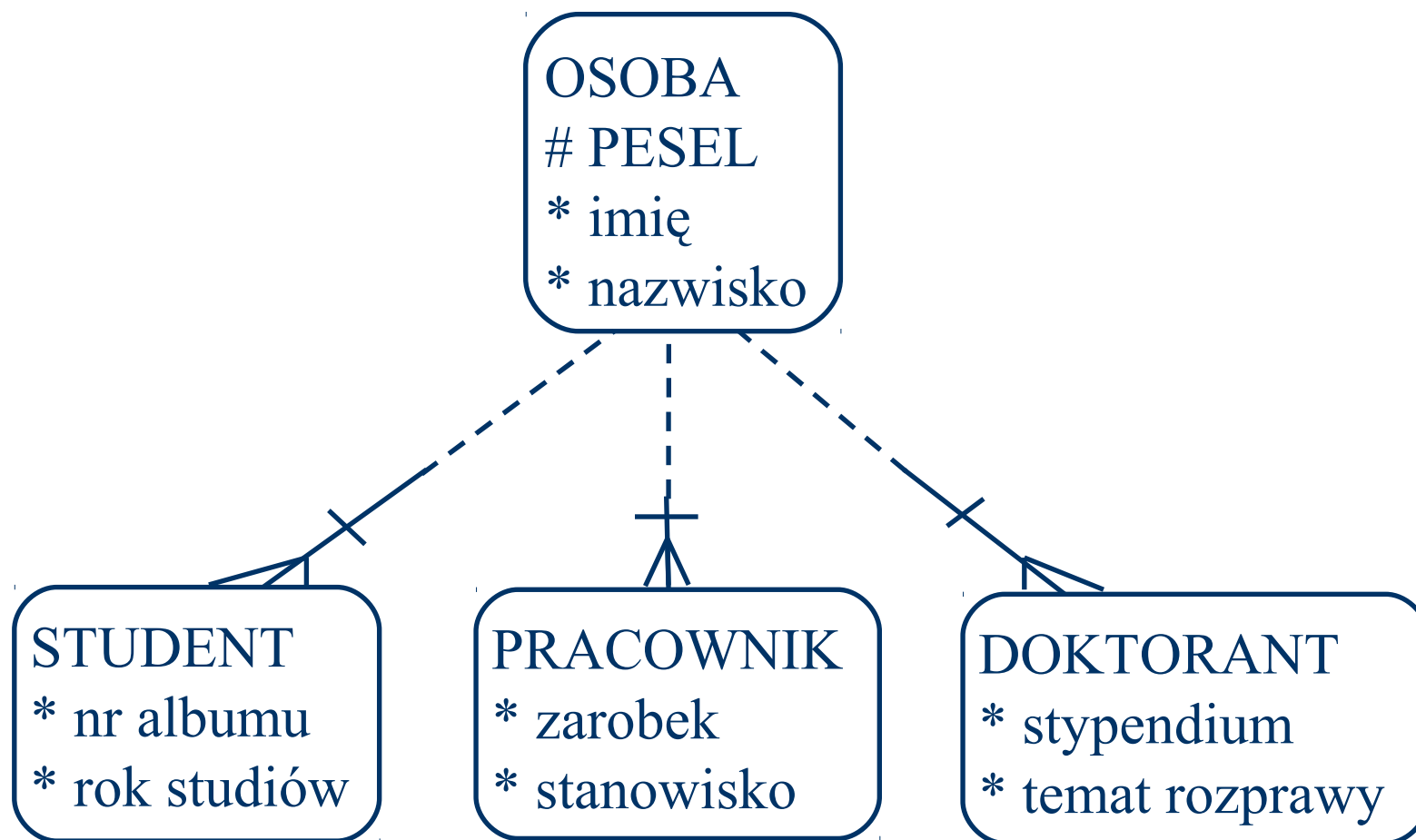
Bez więzów pokrycia



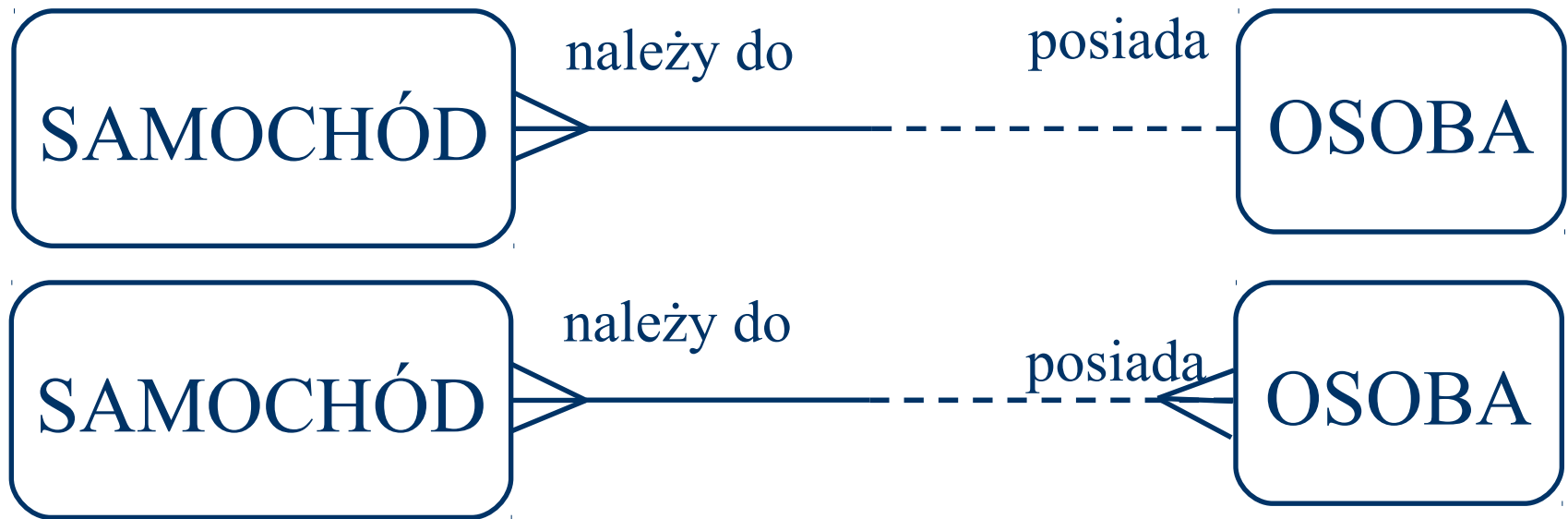
Bez więzów rozłączności i pokrycia



Role wielokrotne



Modelowanie zmian w czasie



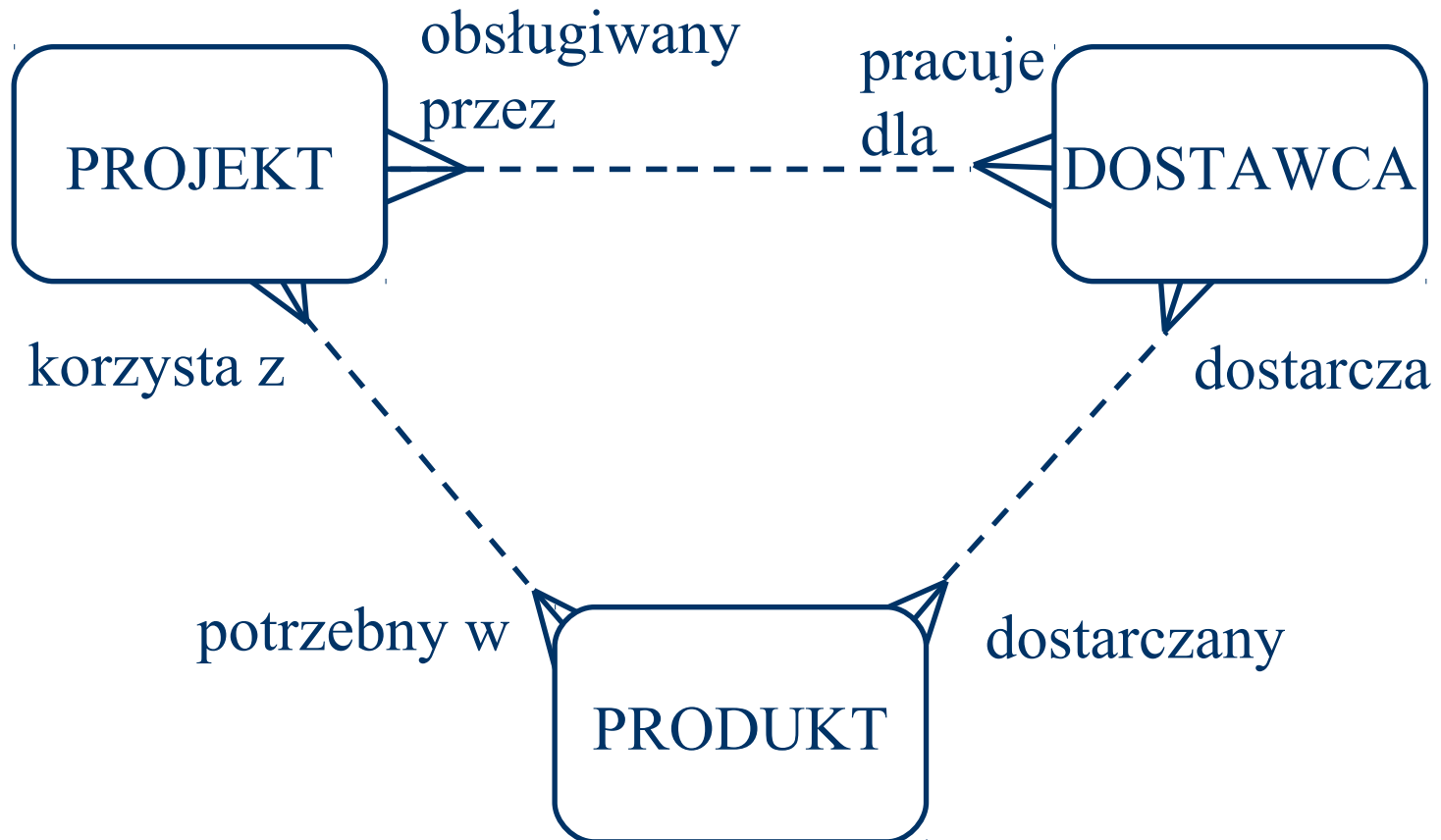
- Jeśli chcemy pamiętać historię własności?
 - Atrybut związku
 - Związek wiele-do-wiele

Modelowanie zmian w czasie

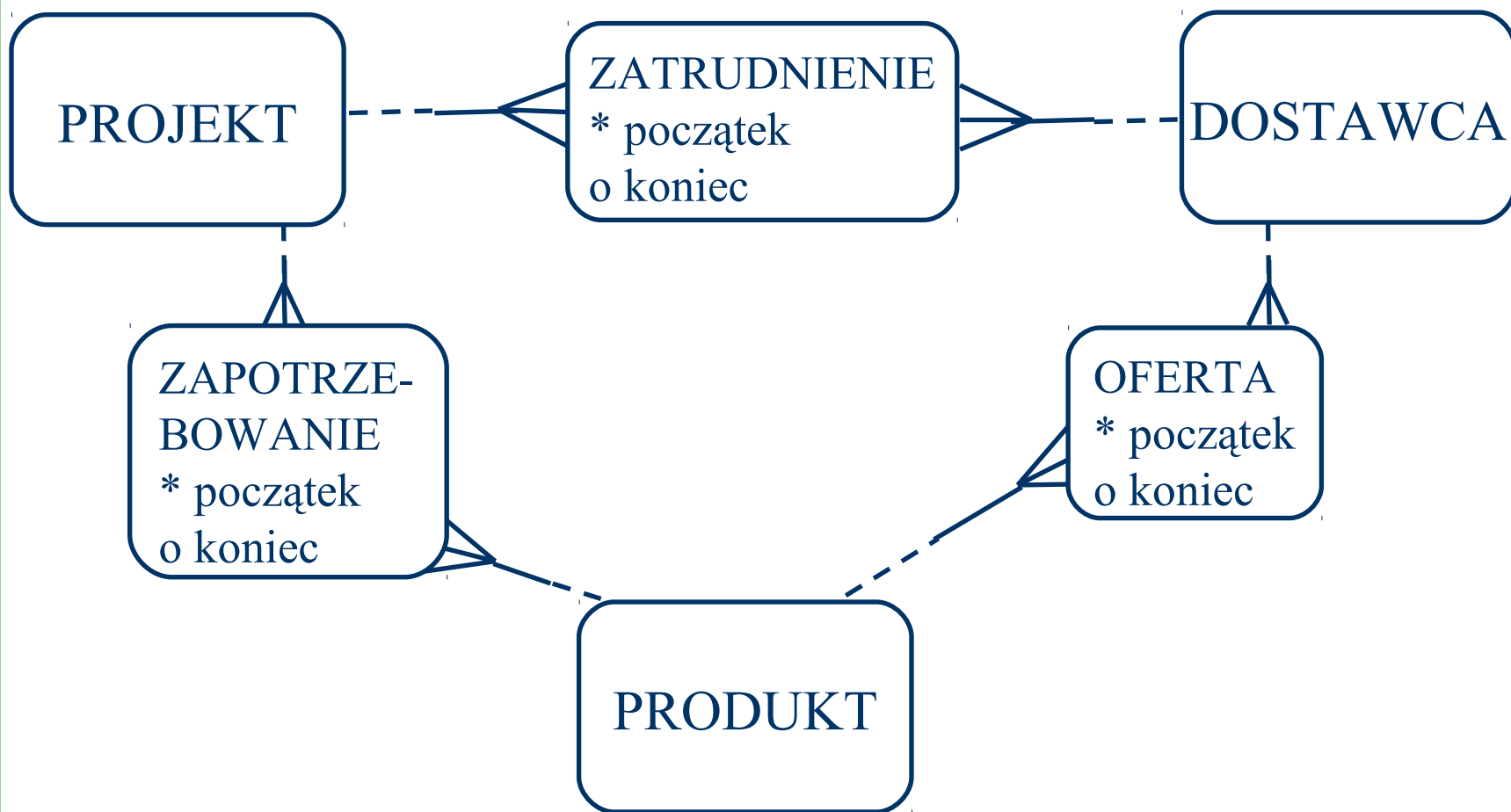


- Charakterystyczna nazwa związku „dotyczy” i „podlega”
- Często nazywanie takich związków wymaga inwencji
- Dużo atrybutów związku

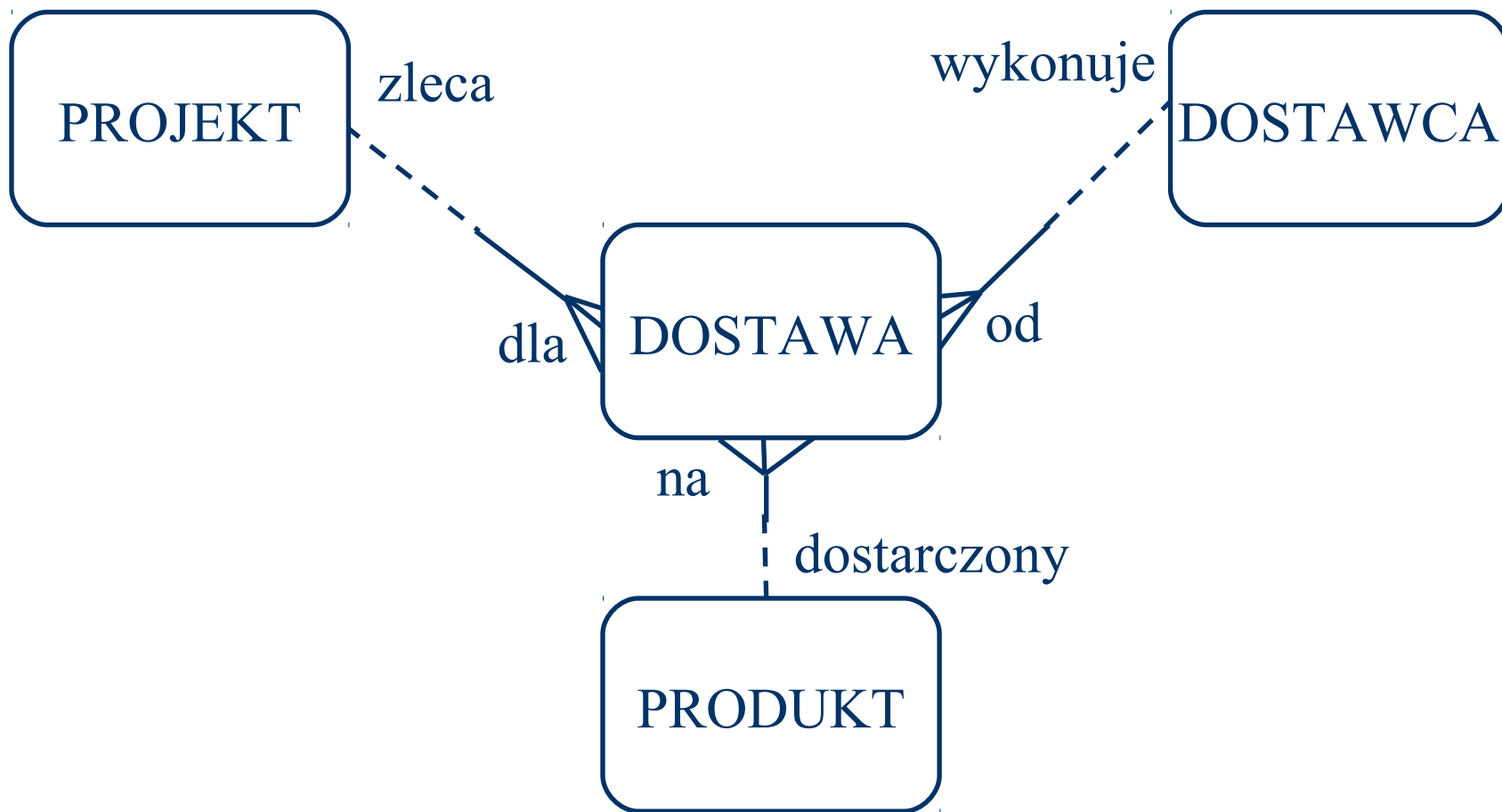
Wentylator (kojarzyć z 5NF)



Szalejący wentylator



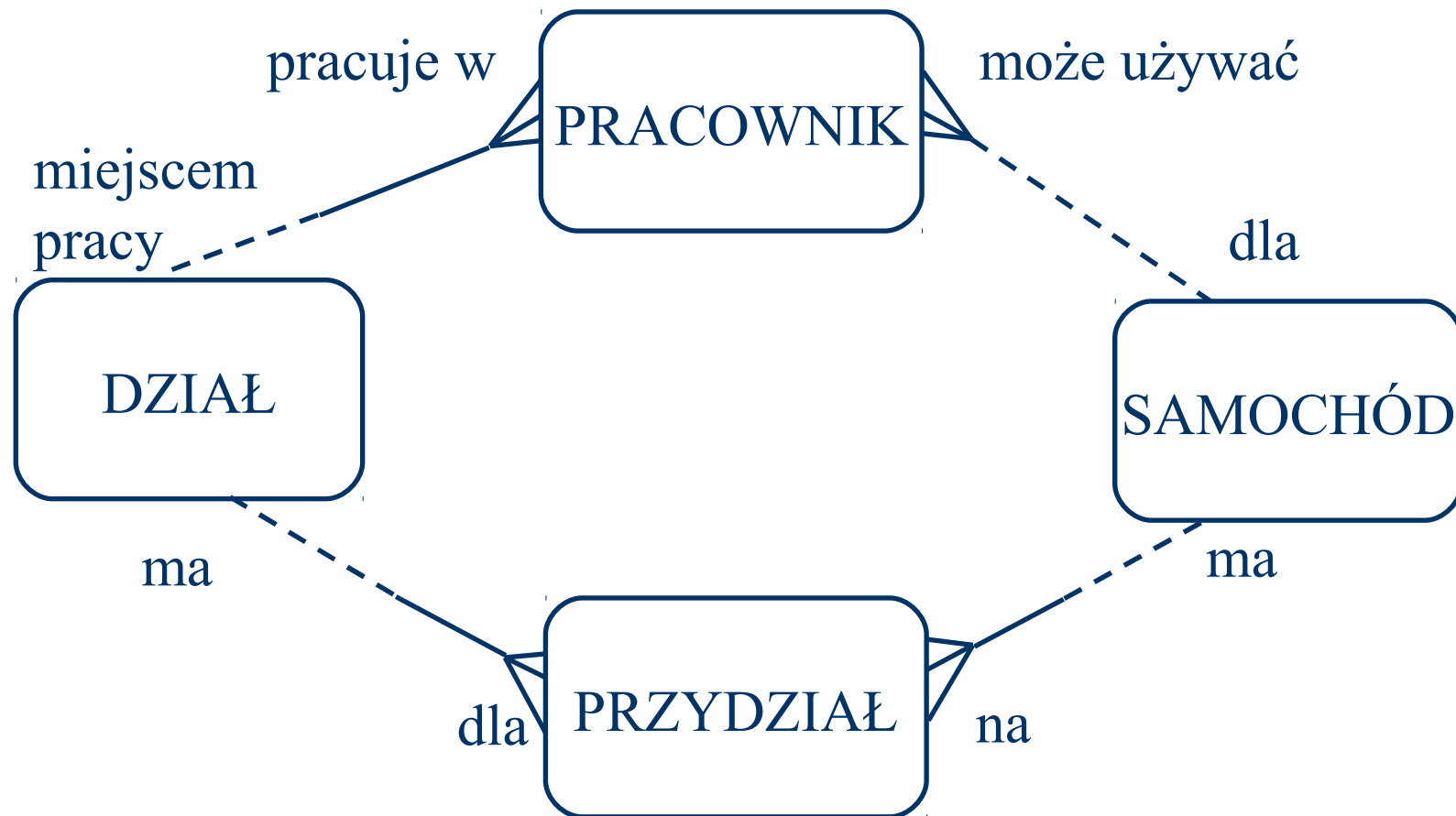
Wentylator wywiał dostawy



Łopaty i oś wentylatora

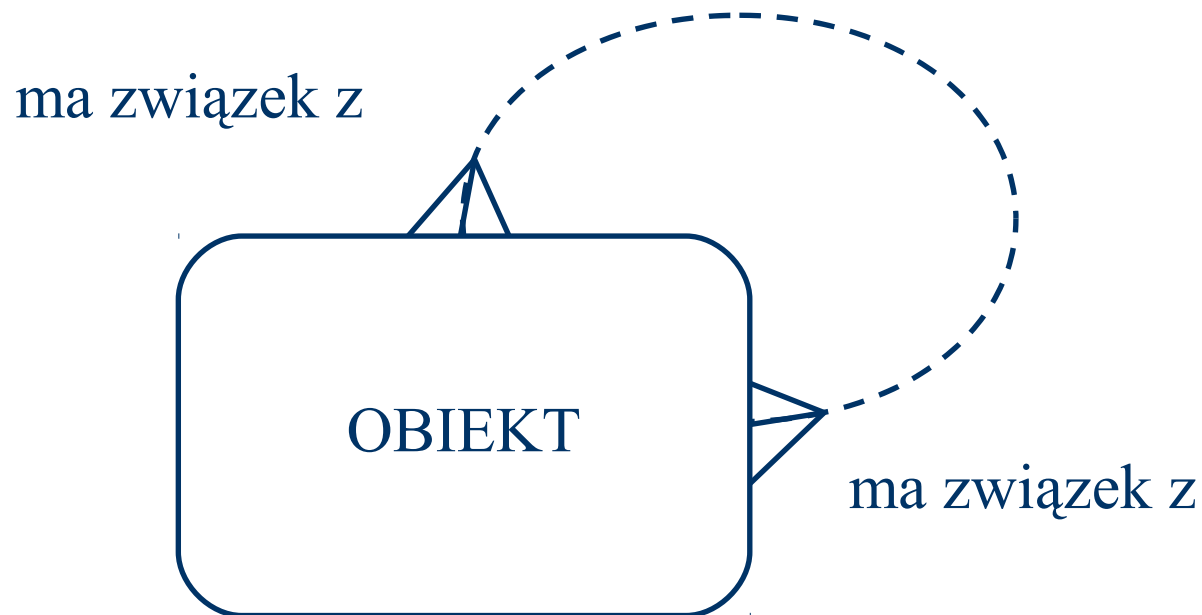
- Oś wentylatora (prawie) zawsze ma sens biznesowy
- Łopaty wentylatora też mogą mieć sens biznesowy i mogą znaleźć się w modelu
- Tu tak jest:
 - Oferta
 - Zatrudnienie
 - Zapotrzebowanie

Przepaść



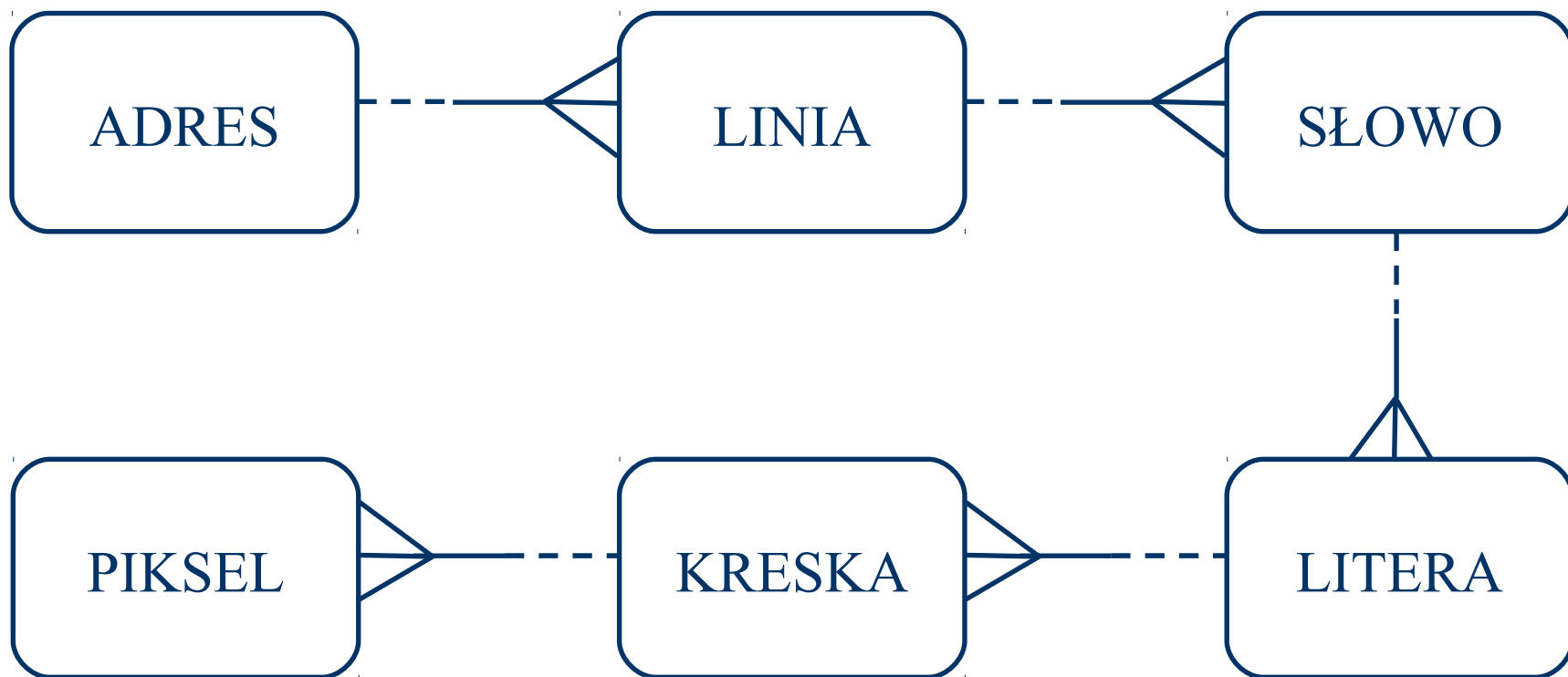
Abstrakcja jest dobra, ale...

- Uważaj na model wszystkiego:



Szczegółowość jest dobra, ale...

- Uważaj na model totalny:



Związki nietransferowalne (Oracle*Method)



- Jak już dyplom jest czyjś, to nie będzie już należał do nikogo innego

Diagram i model

- Model zawiera wszystkie informacje, również takie, których nie pokazuje się na diagramie
 - Typ danych atrybutu
 - Jednostka miary
 - Informacje ilościowe (ile wystąpień encji? Ile pustych atrybutów?)
- Diagram jest pewnym obrazem (przekrojem) przez model. Każdy diagram może mieć inny poziom abstrakcji
- Na diagramie widać tylko niektóre encje i tylko niektóre o nich informacje
- Encja może być na dowolnej liczbie (też zerze) diagramów

Oglądanie modelu

Repository Object Navigator - HR(1): Entity Properties

File Edit View Properties Utilities Application Tools Options Window Help

HR(1): Navigator

- HR (1)
 - Reference Data Definition
 - Enterprise Modeling
 - Entity/Relationship Modeling
 - Entities
 - DEPARTMENT**
 - EMPLOYEE
 - Attributes
 - ENAME
 - SAL
 - HIREDATE
 - JOB
 - EMPNO
 - Relationships
 - works in DEPARTMENT
 - Synonyms
 - Sub Entities
 - Unique Identifiers
 - Usages
 - Entity Relationship Diagrams
 - CORE Entity Relationship Diagram
 - Dataflow Modeling

HR(1): Entity Properties

Application Owner	HR
Name	DEPARTMENT
Short Name	DEPT
Plural	DEPARTMENTS
Type Of	
Volumes	
Initial	
Average	
Maximum	
Annual Growth Rate (%)	
Datawarehouse	
Datawarehouse Type	
Documentation	
Description	
Notes	

DEPARTMENT

UPDATE

Gotowy model encja-związek

- Może służyć do wygenerowania wstępnego projektu tabel
- Nie należy utożsamiać pojęć encja-tabela(-plik)
- Choć pokusa jest...
- Uproszczenie małe
- A koszty mieszania poziomów abstrakcji duże

Bazy danych



09:

Projektowanie bazy
danych (logiczne)

Krzysztof Stencel

Generowanie wstępnego projektu bazy danych

- Analizujemy model związków encji i generujemy
 - Dla każdej encji: tabelę
 - Dla każdego atrybutu: kolumnę
 - Dla każdego związku 1:N: klucz obcy
 - Dla każdego związku N:M tabelę pośredniczącą
- Tabela implementująca związek N:M
 - Ma dwa klucze obce do połączonych tabel
 - Oba te klucze obce łącznie stanowią klucz główny

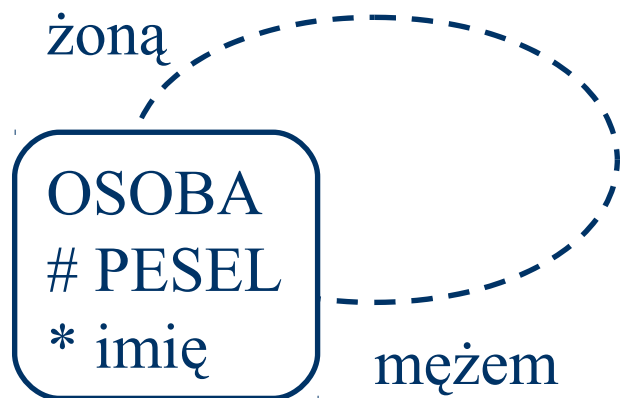
Implementacja związku wieloznacznego



<u>VIN</u>	<u>PESEL</u>
W0L0758324758	78120612345
W0L04356BVV33	75110145667
...	...

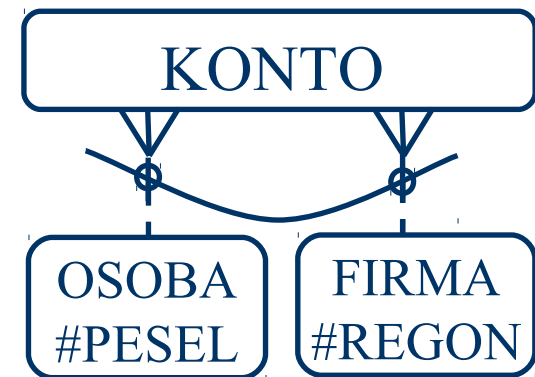
Implementacja związku 1:1

- Klucz obcy może być po dowolnej stronie
- Teoretycznie lepiej żeby był po stronie wymaganej (jeśli taka jest)
- Ciekawy przypadek związku rekurencyjnego 1:1



<u>PESEL</u>	Imię	PESEL_jakiś
75567567	Jaro	99878787
99878787	Jola	75567567
...

Łuki wykluczające



<u>Nr</u>	Saldo	PESEL	REGON
1	50.00	7606...	
2	53.98		P213...
3	-4.98	5645...	

- Dwa klucze obce
 - Kontrola wspierana w SZBD
 - Liczny NULL

<u>Nr</u>	Saldo	Własc.	Typ Wł.
1	50.00	7606...	O
2	53.98	P213...	F
3	-4.98	5645...	O

- Jeden klucz obcy
 - Kontrola nie wspierana w SZBD
 - Pole typu całkiem wygodne

Encje podrzędne

OSOBA
#PESEL

STUDENT

* rok

PRACOWNIK

* zarobek

Studenci	
<u>PESEL</u>	rok
6456...	1
4545...	2

Pracownicy	
<u>PESEL</u>	zar.
1234...	100
1236...	200

Osoby			
<u>PESEL</u>	rok	zar.	Typ
6456...	1		S
1236...		200	P

- Dwie tabele
 - Podwójny Koszt szukania osoby
 - Więzy pokrycia i rozłączności nie do przejścia

- Jedna tabela
 - Liczny NULL
 - Niezgodność z modelem pojęciowym

Trzy tabele

OSOBA
#PESEL

STUDENT

* rok

PRACOWNIK

* zarobek

Studenci	
<u>PESEL</u>	rok
6456...	1
4545...	2

Pracownicy	
<u>PESEL</u>	zar.
1234...	100
1236...	200

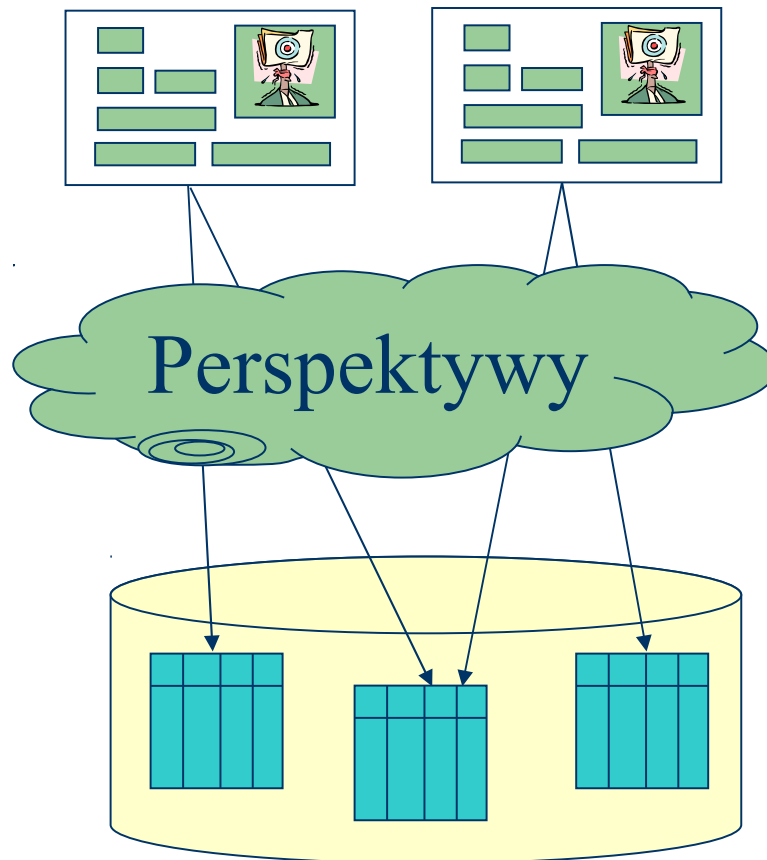
Osoby	
<u>PESEL</u>	
6456...	
1236...	

- Najbardziej elastyczne
 - Łatwo przejść na role dynamiczne
- Najbliższe modelowi pojęciowemu
- Ale jest koszt przetwarzania – złączanie tabel

To było generowanie wstępnego projektu...

- Lepiej nie traktować go jako ostatecznego
- Punkt 1: wydajność
- Punkt 2: ewolucja schematu

Logiczna niezależność danych



- Aplikacje korzystają z danych z tabel
- Schemat tabel może się zmienić
- Musi być coś pośrodku, co zapewni logiczną niezależność danych
- Perspektywy lub API

Perspektywy

- Zapamiętana w bazie danych definicja zapytania do późniejszego użycia
- Perspektywa jest “wirtualną” tabelą
- Można jej używać tak jak tabeli
- Jej wiersze nie są przechowywane w bazie danych. Są wyliczane na żądanie

```
CREATE VIEW Clerks (Empno, Ename, Sal) AS  
  SELECT Empno, Ename, Sal  
    FROM Emp  
   WHERE Job = 'CLERK';
```

Perspektywy - cel

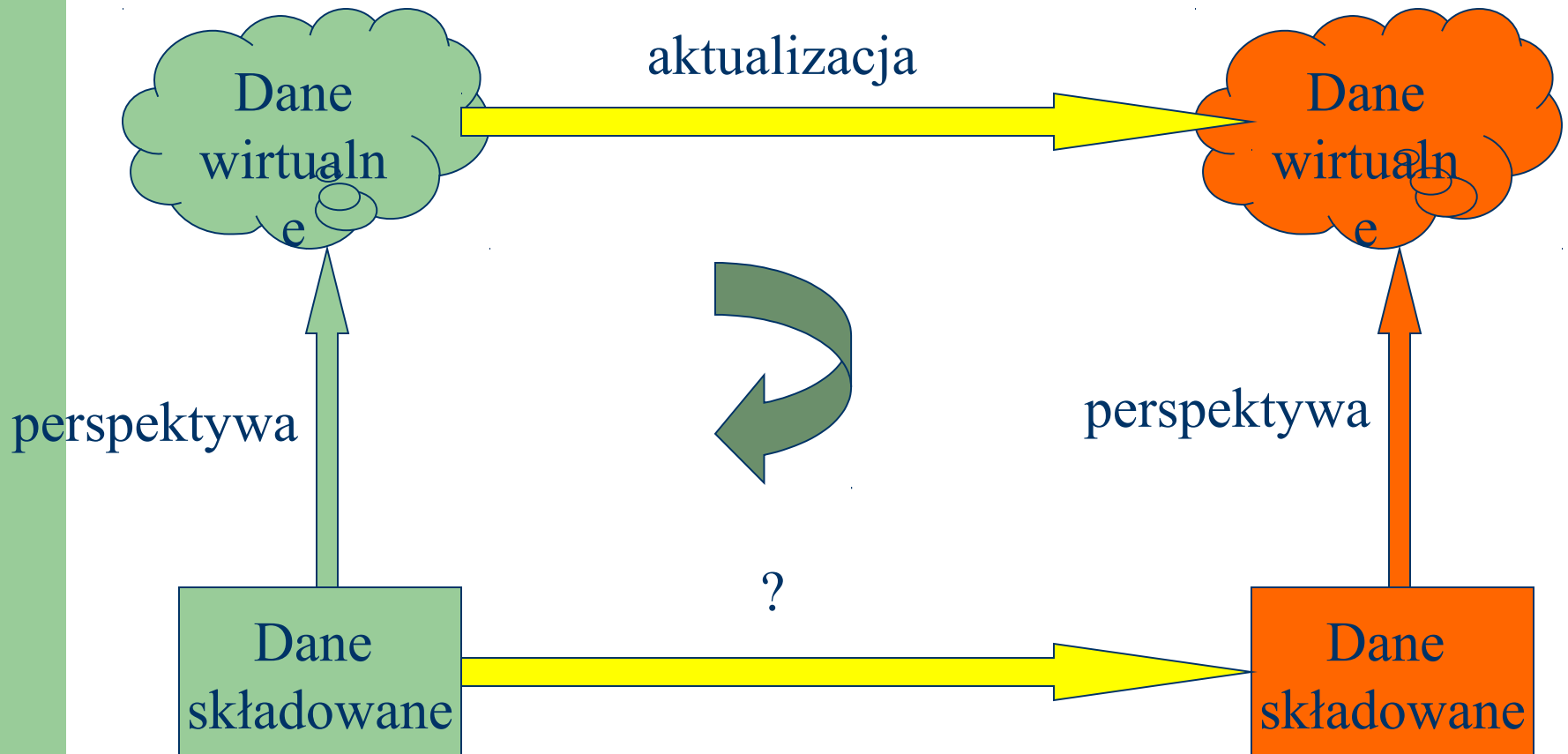
- Perspektywy dostosowują zawartość bazy danych do potrzeb rozmaitych użytkowników
- Dają broń na wypadek ewolucji schematu
- Pozwalają ukryć pewne dane
- Pozwalają zdenormalizować dane
- Muszą pozwalać na aktualizację (INSERT, UPDATE, DELETE), jeśli mają stanowić jedyny interfejs do bazy danych, np.

```
UPDATE Clerks SET Sal = Sal * 1.1;
```

Aktualizacja perspektyw

- Użytkownicy nie tylko czytają dane, ale też aktualizują
- Aktualizacja danych wirtualnych musi zostać odwzorowana na aktualizację danych składowanych
- Jeśli tego nie ma, trzeba szukać jakichś „obejść”, co niweczy przezroczystość perspektyw

Aktualizacja perspektyw – formalizacja



Aktualizacja perspektyw w SQL-92

- Bardzo duże ograniczenia na zapytanie definiujące aktualizowalną perspektywę:
 - Zwraca tylko wartości kolumn
 - Bez GROUP BY i DISTINCT
 - Bez podzapytań
 - **Bez złączeń!**
- To powoduje praktyczne wykluczenie aktualizacji perspektyw
- W implementacjach SZBD znacznie lepiej, bo są możliwości aktualizacji złączeń

Nieaktualizowalna perspektywa

```
CREATE View DeptBudget
  SELECT Dept.Deptno,
         NVL(SUM(Sal), 0) AS Budget
  FROM Dept LEFT JOIN Emp
         ON (Dept.Deptno = Emp.Deptno)
 GROUP BY Dept.Deptno;
```

Aktualizacja perspektyw w SQL-99

- Uwolniono aktualizację poprzez wprowadzenie wyzwalaczy INSTEAD OF.
- Programista sam wskazuje, co ma się stać przy aktualizacji perspektywy.

Wyzwalacz INSTEAD OF dla „nieaktualizowanej perspektywy”

```
CREATE TRIGGER DeptBudgetUpdate
  INSTEAD OF UPDATE OF Budget on
  DeptBudget
DECLARE v_vat NUMBER(4,1);
BEGIN
  UPDATE Emp
    SET Sal = Sal * :NEW.Budget / :OLD.Budget
    WHERE Deptno = :OLD.Deptno;
END;
```


Zniesienie ograniczenia na odwzorowanie danych

- Wyzwalacze INSTEAD OF pozwalają znieść ograniczenia na dopuszczalne aktualizacje danych wirtualnych
- Samo odwzorowanie jest nadal ograniczone, bo SQL jest językiem słabym
- Odwzorowanie powinno być definiowane w języku, który jest silniejszy (Datalog, XQuery)
- Inna możliwość: zastosowanie API zamiast perspektyw

Wstawić można cokolwiek

```
CREATE VIEW RichEmp AS  
  SELECT Empno, Ename, Sal  
  FROM Emp  
  WHERE Sal > 1000;
```

```
INSERT INTO RichEmp  
  VALUES (2344, 'Miller', 999);
```

1 row created;

Wymuszanie kontroli jawne

```
CREATE VIEW RichEmp AS  
  SELECT Empno, Ename, Sal  
    FROM Emp  
   WHERE Sal > 1000  
  WITH CHECK OPTION;
```

```
CREATE VIEW Clerks (Empno, Ename, Sal) AS  
  SELECT Empno, Ename, Sal  
    FROM Emp  
   WHERE Job = 'CLERK'  
  WITH READ ONLY;
```

Dane wyliczane, denormalizacja

- Liczenie na bieżąco
 - Koszt przy każdym odwołaniu
 - Brak redundancji
 - Kolumna perspektywy lub procedura/funkcja
- Materializacja

Materializacja

- Kolumna/tabela plus wyzwalacz aktualizujący
 - Redundancja
 - Koszt aktualizacji (synchroniczna/asynchroniczna)
 - Synchroniczna: dane są zawsze aktualne
- Perspektywa zmaterializowana
 - Brak redundancji w schemacie
 - Niższy koszt
 - Czasem nieaktualne dane – dostosować model biznesowy

Perspektywy zmaterializowane

- Tak samo jak perspektywa, ale dane są przechowywane i udostępniane na żądanie
- Odświeżane co pewien czas

```
CREATE SNAPSHOT DeptBudget
```

```
  REFRESH NEXT Sysdate + 1
```

```
  SELECT Dept.Deptno, NVL(SUM(Sal), 0) AS Budget
```

```
  FROM Dept LEFT JOIN Emp
```

```
    ON (Dept.Deptno = Emp.Deptno)
```

```
  GROUP BY Dept.Deptno;
```

Korzystanie

- Tak jak z normalnej tabeli/perspektywy
- Jeśli jako perspektywa byłaby aktualizowana, to może mieć możliwość aktualizacji.
- Może mieć wyzwalacze INSTEAD OF
- Trzeba to zaznaczyć:

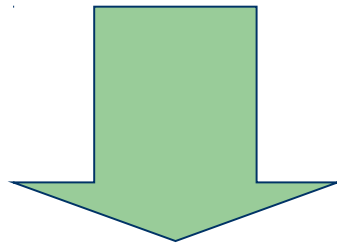
```
CREATE SNAPSHOT DeptBudget  
  REFRESH NEXT Sysdate + 1  
  FOR UPDATE  
  SELECT ...
```

Przepisywanie zapytań

- Można w konfiguracji SZBD włączyć użycie perspektyw zmaterializowanych przy optymalizacji zapytań
- Jeśli SZBD rozpozna gdzieś fragment SQL odpowiadający definicji perspektywy zmaterializowanej, to zmieni zapytanie tak, żeby jej użyć

Przykład przepisywania

```
SELECT Dept.Deptno, Dname, Loc, NVL(SUM(Sal), 0)
FROM Dept LEFT JOIN Emp
            ON (Dept.Deptno = Emp.Deptno)
GROUP BY Dept.Deptno, Dname, Loc;
```



```
SELECT Dept.Deptno, Dname, Loc, DeptBugdet.Budget
FROM Dept, DeptBugdet
WHERE Dept.Deptno = DeptBugdet.Deptno;
```

Projekt logiczny bazy danych

- Wygenerowanie projektu wstępnego
- Decyzja co do implementacji łuków i podencji
- Decyzja co do interfejsu z aplikacjami (perspektywy/API)
- Aktualizacja perspektyw
- Denormalizacja, dane wyliczane
 - Sposób realizacji (perspektywy, procedury, materializacja)
 - Aktualizacja: synchroniczna/asynchroniczna

Fizyczna organizacja danych w bazie danych. Indeksy.

Przygotował Lech Banachowski na podstawie:

- 1. Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, McGrawHill, 2002 (książka i slide'y).*
- 2. Lech Banachowski, Krzysztof Stencel, Bazy danych – projektowanie aplikacji na serwerze, EXIT, 2001.*

Model fizyczny bazy danych jest oparty na pojęciu pliku i rekordu.

- *Plik* składa się z rekordów w tym samym *formacie*.
- *Format rekordu* jest listą nazw *pól*.
- *Rekord* składa się z wartości poszczególnych *pól*.
- Niektóre pola są wyróżnione jako *klucz rekordu* – ich wartości jednoznacznie identyfikują cały rekord.

Podstawowymi operacjami na pliku są:

- *Wstawianie* - wstaw rekord do pliku.
- *Usuwanie* - usuń rekord z pliku.
- *Modyfikacja* - zmodyfikuj zawartość pól w rekordzie w pliku.
- *Wyszukiwanie* - znajdź w pliku rekord(y) z podaną wartością w danym polu lub spełniające podane warunki.

Dyski i pliki

- SZBD przechowuje dane na twardych dyskach.
- Stąd konieczność stosowania operacji We/Wy:
 - **Odczyt (READ)**: przesłanie danych z dysku do pamięci RAM.
 - **Zapis (WRITE)**: przesłanie danych z pamięci RAM na dysk.
 - Obie operacje są o rząd wielkości wolniejsze niż operacje w pamięci RAM – powinny być stosowane umiejętnie! **Koszt operacji** na bazie danych jest przedstawiany jako **liczba operacji We/Wy**.

Dlaczego nie można przechowywać danych w pamięci RAM?

- Pamięć RAM jest chwilowa.*
- Za duży koszt.*

□ Typowa hierarchia pamięci w bazie danych:

- Pamięć RAM dla danych używanych w bieżącej chwili.
- Dysk dla głównej bazy danych.
- Taśma dla archiwalnych wersji danych.

Dyski

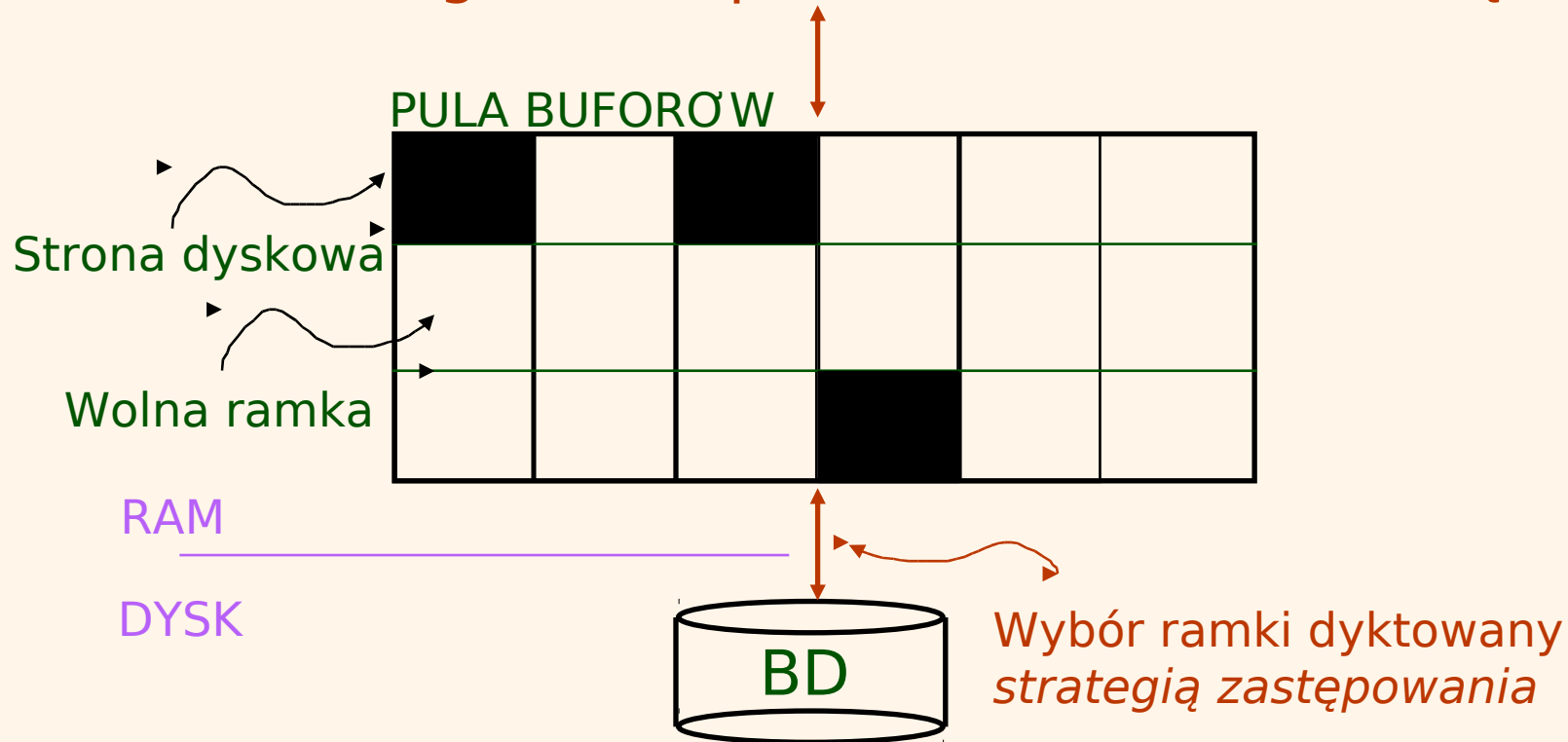
- Dostęp swobodny (random access) – w przypadku dysków; dostęp sekwencyjny – w przypadku taśm.
- Dane są przechowywane i przekazywane w jednostkach nazywanych *blokami dyskowymi* lub *stronami*.
- Inaczej niż w przypadku RAM, czas dostępu do danych na dysku zależy od ich położenia na dysku.
 - Dlatego wzajemne rozmieszczenie stron na dysku ma zasadniczy wpływ na szybkość działania SZBD! Najlepiej operować ciągami sąsiadujących ze sobą stron.

Zarządzanie miejscem na dysku

- ▣ Realizowane funkcje:
 - Alokacja/dealokacja strony.
 - Odczyt/zapis strony.
 - Sekwencyjna alokacja ciągu stron.

Zarządzanie buforami (w RAM)

Proces zgłasza zapotrzebowanie na stronę



- Dane muszą być w RAM aby SZBD mógł na nich operować!
- Tablica par $\langle \text{nr.ramki}, \text{idstrony} \rangle$.

Gdy procesorowi jest potrzebna strona...

- Gdy nie ma jej w puli buforów:
 - Wybierz ramkę o liczniku odwołań = 0 .
 - Jeśli ramka jest niezaktualizowana ("*dirty*"), zapisz ją na dysk.
 - Wczytaj potrzebną stronę w wybraną ramkę.
 - Ustaw licznik odwołań do tej strony na jeden.
 - Gdy strona jest w puli buforów, zwiększ jej licznik odwołań o jeden.
 - Przekaż procesowi wskaźnik do ramki ze stroną.
-
- *Jeśli można z góry przewidzieć (np. przeglądanie sekwencyjne) sprowadza się od razu kilka stron!*

Zarządzanie buforami – c.d.

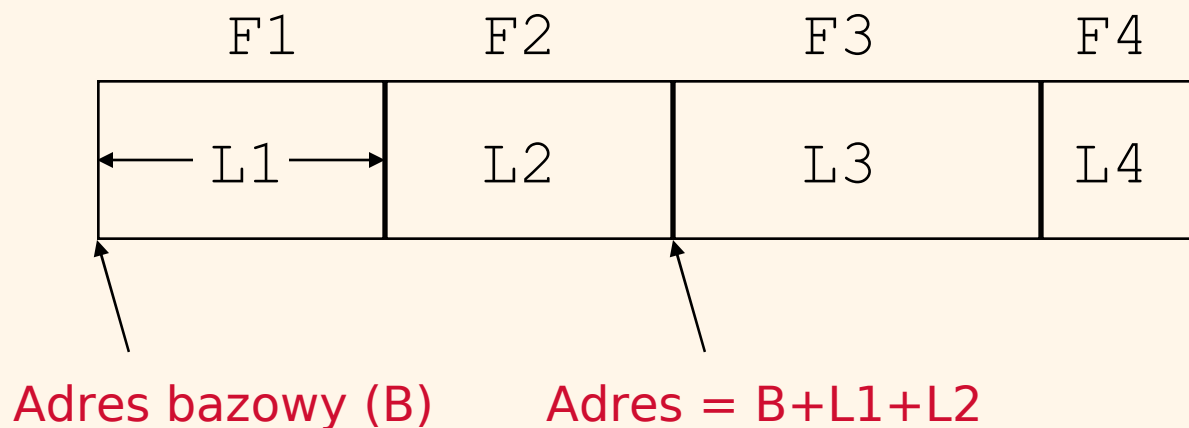
- Gdy zmienia się zawartość strony:
 - Zostaje ustawiony *bit aktualizacji* - “dirty”.
- Strona w buforze może być potrzebna dla wielu procesów:
 - Nowe zapotrzebowanie na stronę zwiększa jej *licznik odwołań* o jeden. Gdy proces zwalnia stronę, jej *licznik odwołań* zmniejsza się o jeden. Strona staje się kandydatem do zastąpienia gdy jej *licznik odwołań* = 0.

Strategie zastępowania ramek

- *LRU* – najdłużej nie używana,
- *Clock* - cyklicznie,
- *MRU* – ostatnio używana.

- *Sekwencyjne zalewanie puli ramek:* *LRU* + powtarzane sekwencyjne przeglądanie pliku.
 - *# ramek < # stron* oznacza, że każde żądanie strony powoduje operację We/Wy. *MRU* lepsze w tym przypadku.

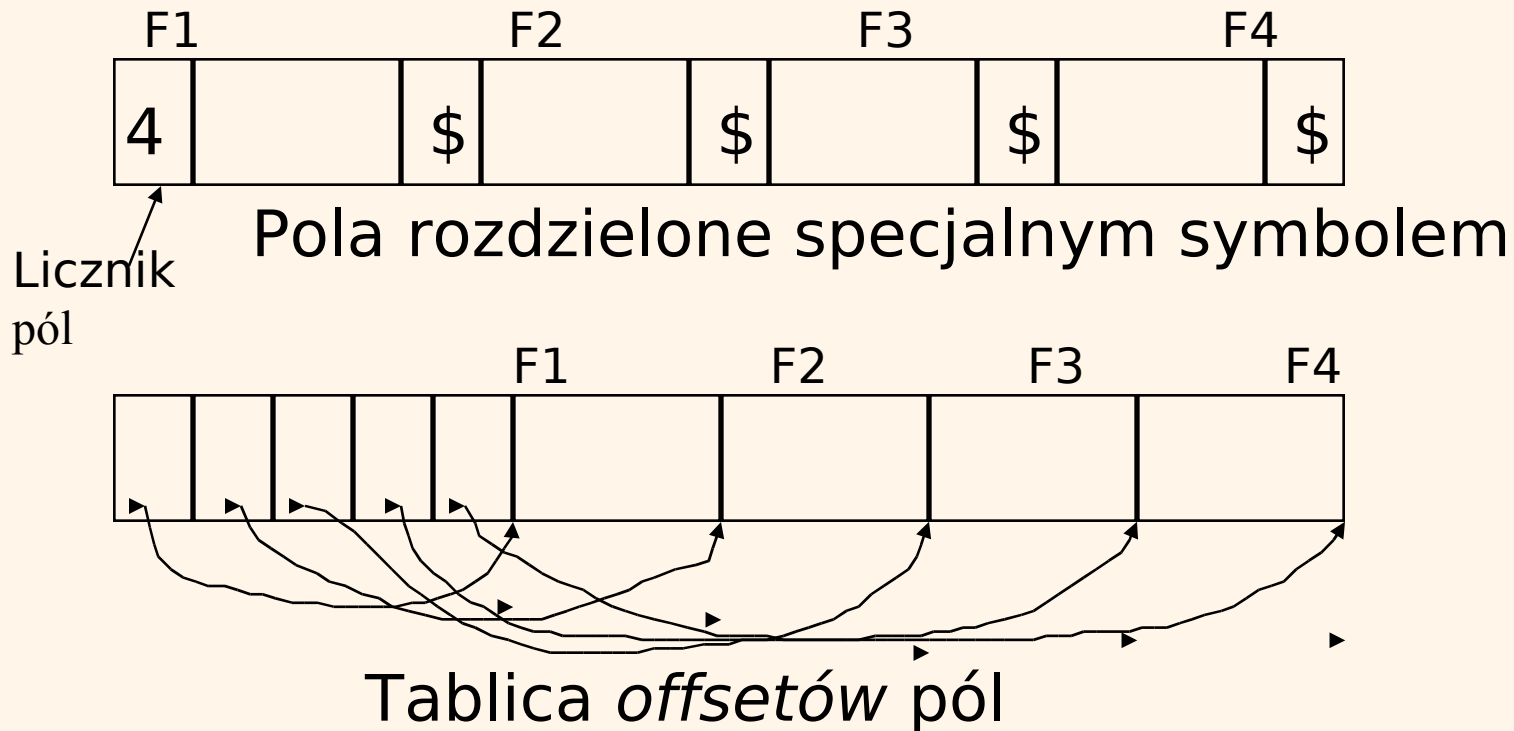
Formaty rekordów: stała długość



- Typy pól takie same dla wszystkich rekordów w pliku; zapisane w *słowniku danych* (*katalogu systemowym*).

Formaty rekordów: zmienna długość

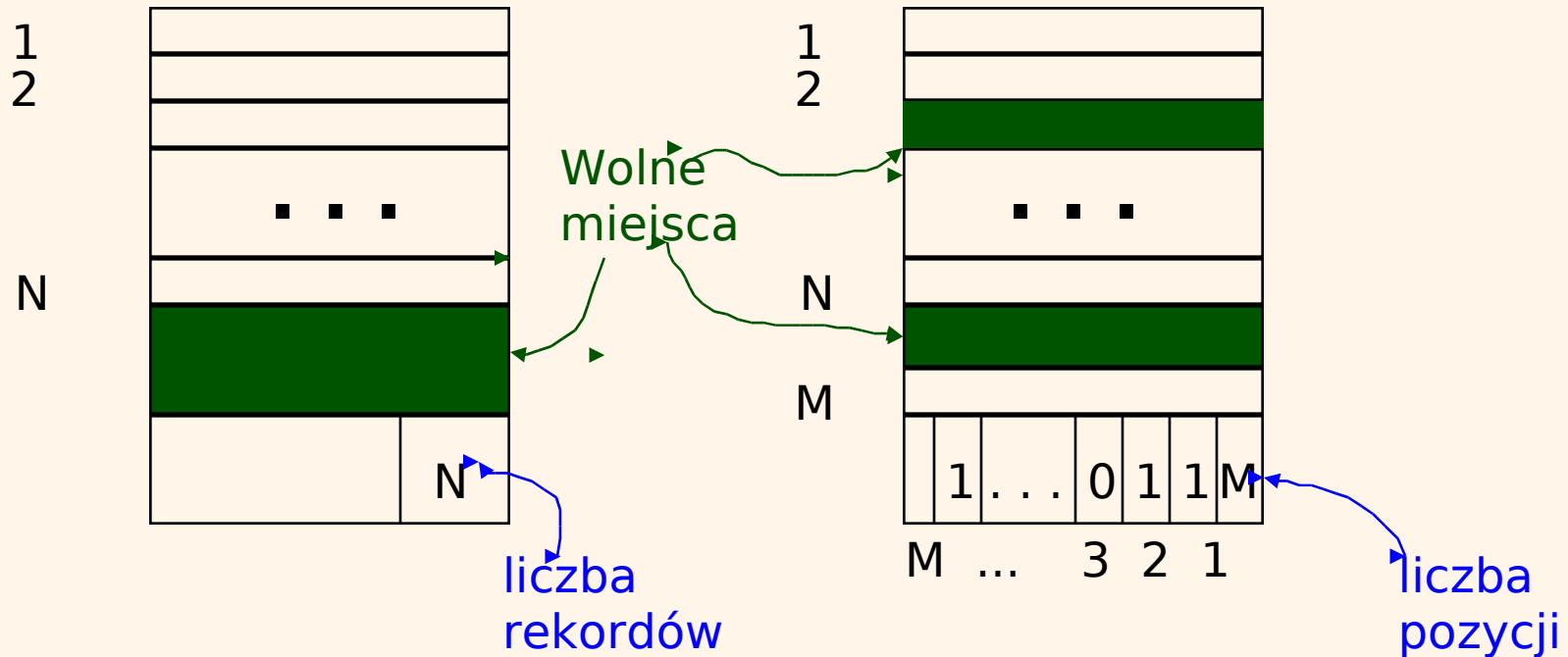
▣ Dwa alternatywne formaty (# pól jest stała):



W drugim przypadku:

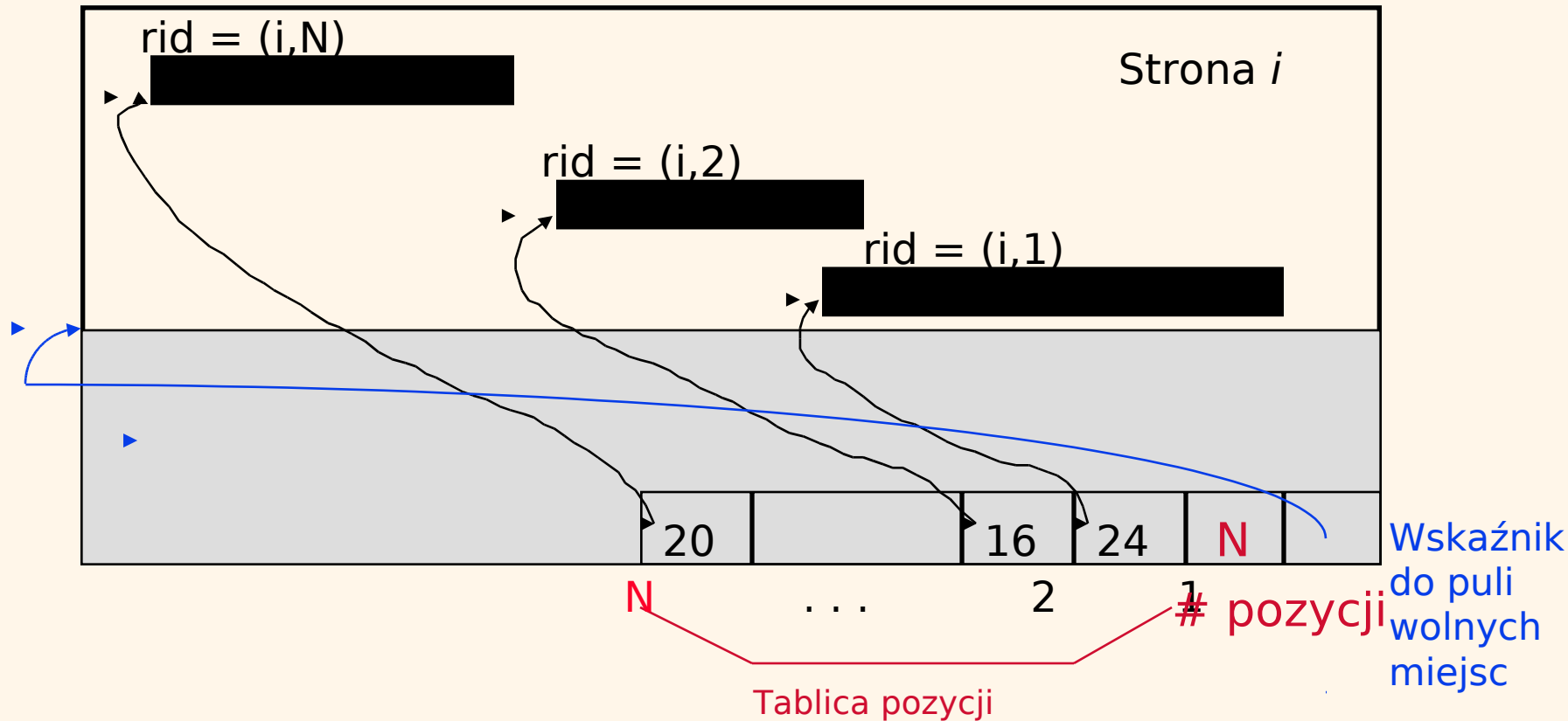
▣ bezpośredni dostęp do wartości i -tego pola;

Formaty stron: rekordy stałej długości



- *Rid (Id rekordu)* = *<idstrony, # pozycji>*. W pierwszym przypadku, przesuwanie rekordów powoduje zmianę id rekordu, co komplikuje odwołania do rekordu przez id rekordu (rid).

Formaty stron: rekordy zmiennej długości



- Można przesuwając rekordy po stronie bez zmiany rid – można także zastosować dla rekordów stałej długości.

Pliki rekordów

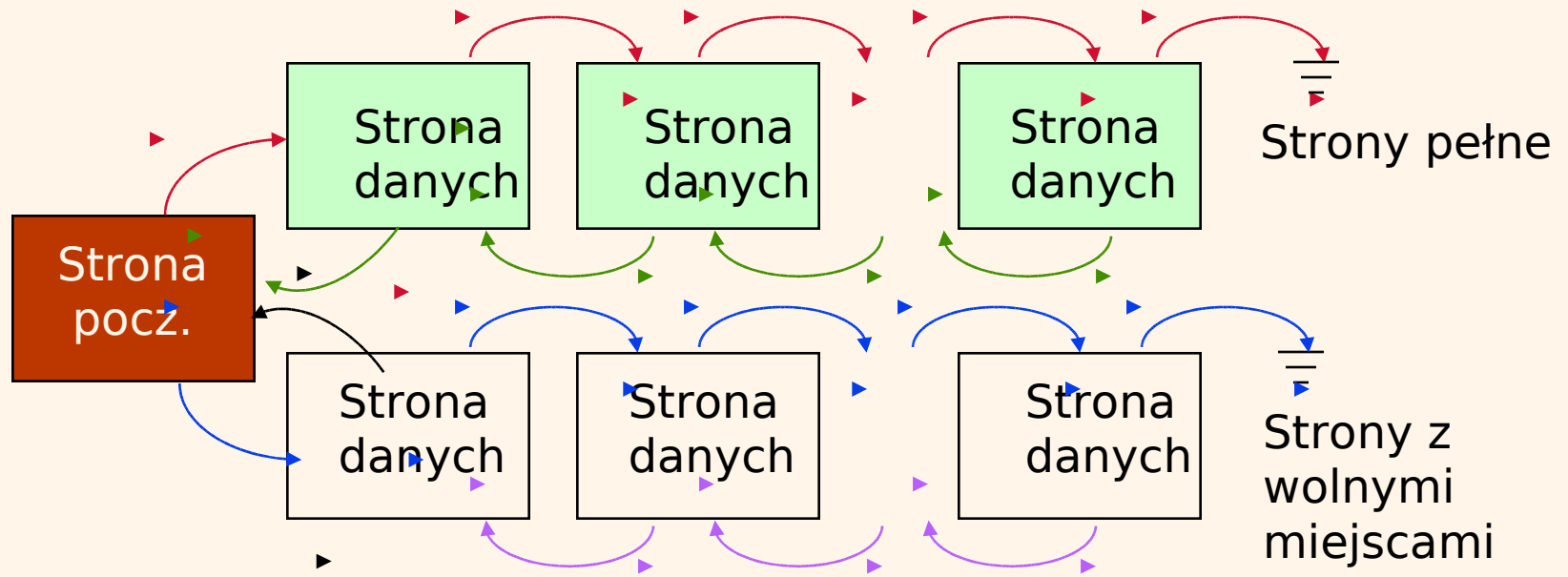
- ▣ PLIK: kolekcja stron, każda zawierająca zbiór rekordów:
 - wstawianie/usuwanie/modyfikowanie rekordów,
 - odczytanie konkretnego rekordu (o podanym rid),
 - wyszukanie wszystkich rekordów (spełniających pewne warunki).

Plik nieuporządkowany (heap)

- *Rekordy są przechowywane na stronach w dowolnym porządku.*
- *Nowy rekord jest wstawiany do pierwszej strony, w której jest wolne miejsce.*
- *Przy wyszukiwaniu trzeba przejść po wszystkich stronach do chwili napotkania szukanego rekordu.*

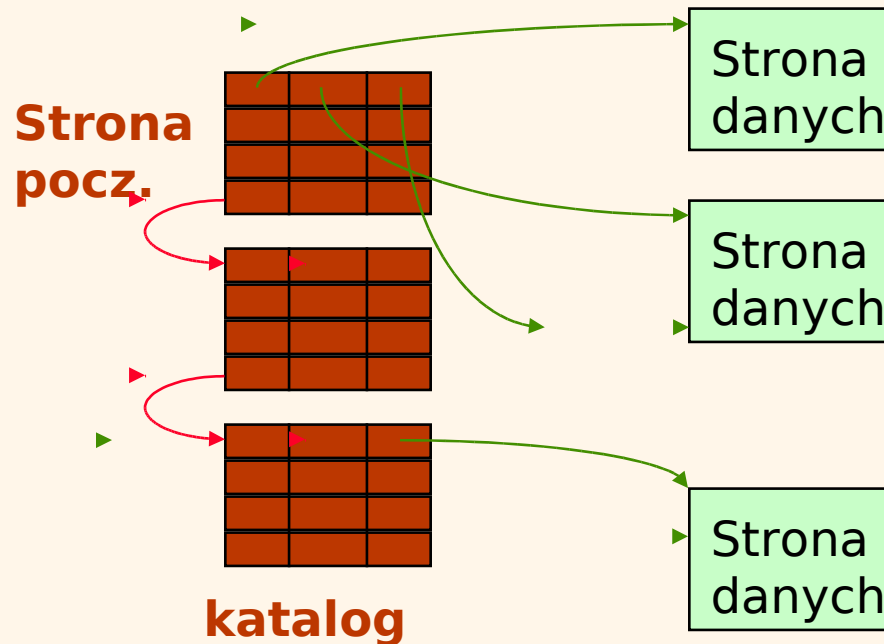
Plik nieuporządkowany

implementacja – dwie listy



Plik nieuporządkowany

implementacja – katalog stron



Inne organizacje pliku rekordów

- Pliki posortowane: wygodne gdy trzeba sprowadzić rekordy w pewnym porządku lub tylko pewien ich zakres.
- Pliki haszowane: Użyteczne przy selekcji przez równość wartości.
 - Plik jest kolekcją “**segmentów**” (*ang. bucket*).
Segment = *strona główna* plus zero lub więcej *stron nadmiarowych*.
 - *Funkcja haszująca* **h**: $h(r)$ = “segment” do którego wpada rekord *r*. **h** bierze pod uwagę tylko niektóre pola *r*, nazywane *polami wyszukiwania*.

Indeksy

- Plik nieuporządkowany umożliwia wyszukanie rekordu:
 - mając dany *rid*, lub
 - przeglądając sekwencyjnie wszystkie rekordy w pliku.
- Często wyszukiwanie na podstawie *wartości jednego lub więcej pól*, np.
 - Wyznacz wszystkich studentów specjalizacji “BD”.
 - Wyznacz wszystkich studentów mających < 20 lat.
- Indeksy są strukturami danych, które pomagają szybko znajdować odpowiedzi na takiego rodzaju zapytania.

Indeksy

- Klucz wyszukiwania dla indeksu
 - wybrane pola rekordu względem których ma odbywać się wyszukiwanie.
- Indeks składa się z **pozycji danych k^*** określanych względem wartości klucza wyszukiwania k .

Indeksy

- **Plik danych** – *rekordy danych.*
- **Plik indeksu** – *pozycje danych, pozycje indeksu.*
- **Indeks wewnętrzny** - plik indeksu może być zbudowany na samym pliku danych. Wówczas pozycja danych k^* pokrywa się z rekordem danych. Może być tylko jeden indeks wewnętrzny.
- **Indeks zewnętrzny** – pozycje danych k^* są rozłączne z rekordami danych. Postacie:
 1. $\langle k, \text{rid rekordu z tą wartością klucza } k \rangle$ albo
 2. $\langle k, \text{lista rid rekordów z tą wartością klucza } k \rangle$
 - Ad 2. Postać bardziej zwarta; zmienna długość rekordu.

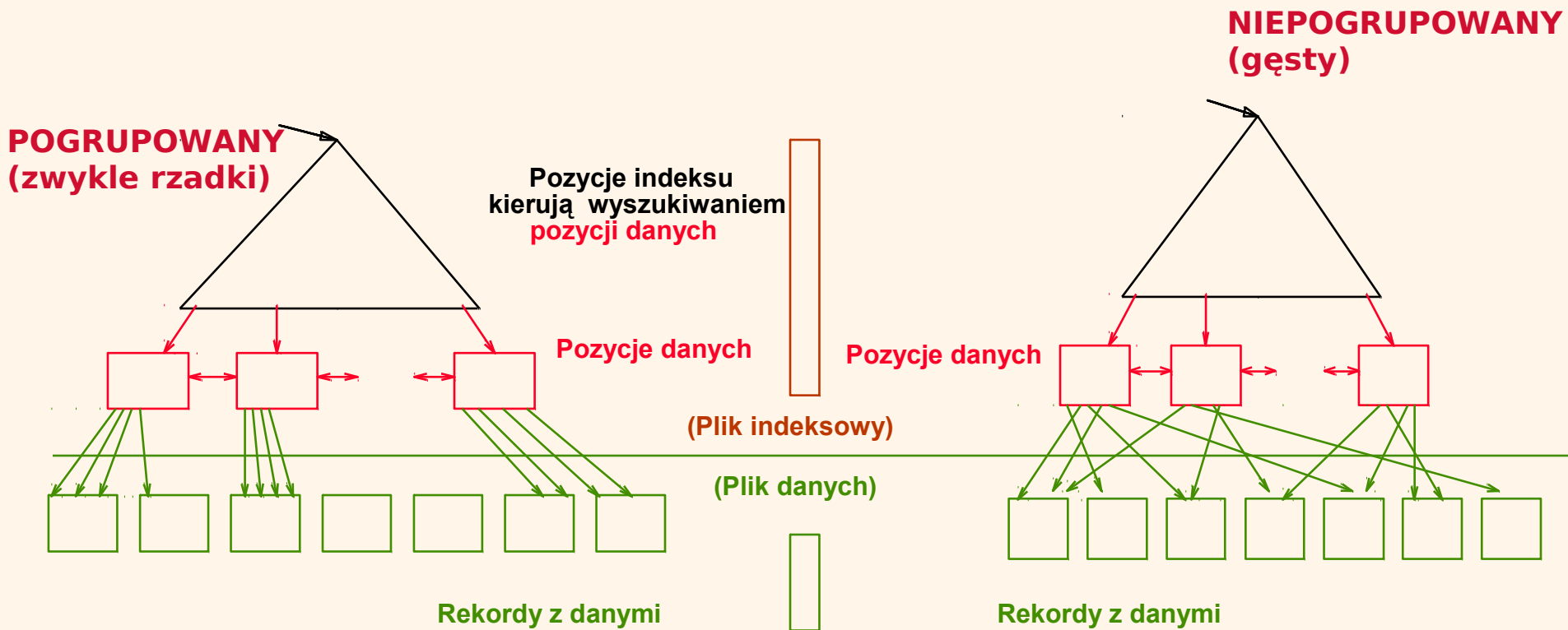
Klasyfikacja indeksów

- Gdy klucz wyszukiwania zawiera klucz główny – ***indeks główny*** wpp. ***indeks niegłówny***.
- ***Indeks jednoznaczny*** - klucz wyszukiwania zawiera klucz kandydujący.

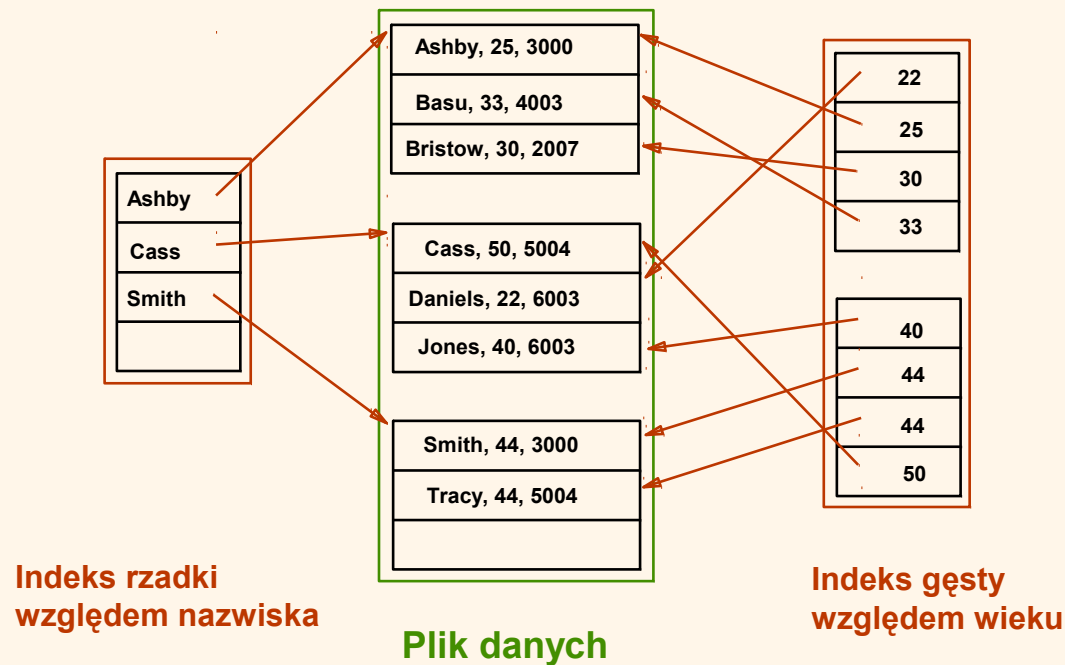
Klasyfikacja indeksów

- Gdy uporządkowanie zapisu rekordów danych jest takie samo jak uporządkowanie zapisu pozycji danych – **indeks pogrupowany (ang. clustered)** wpp. **indeks niegrupowany**.
 - Każdy indeks wewnętrzny jest pogrupowany ale nie vice-versa.
 - Plik może zostać pogrupowany tylko względem jednego klucza wyszukiwania.
- **Indeks gęsty** – rekordy danych i pozycje danych w związku 1-1 wpp. **indeks rzadki** – pozycji danych mniej niż rekordów danych.
 - Każdy indeks niegrupowany jest gęsty ale nie vice-versa.

Indeks pogrupowany i niepogrupowany

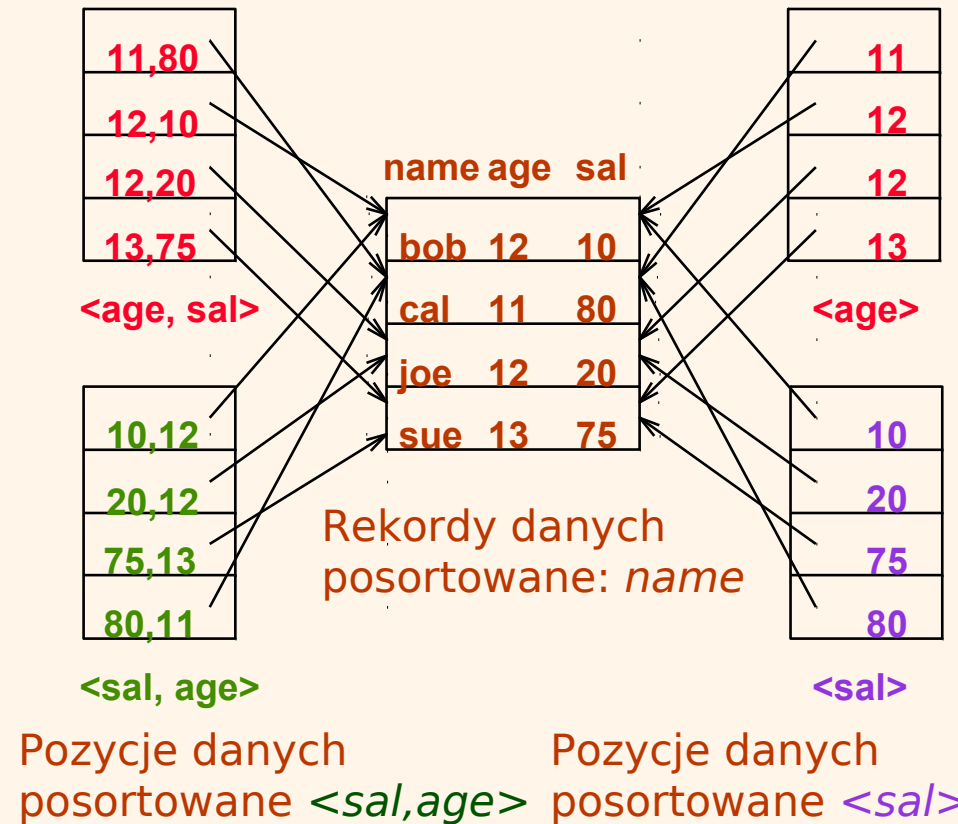


Indeksy gęste (niepogrupowane) oraz rzadkie (pogrupowane)



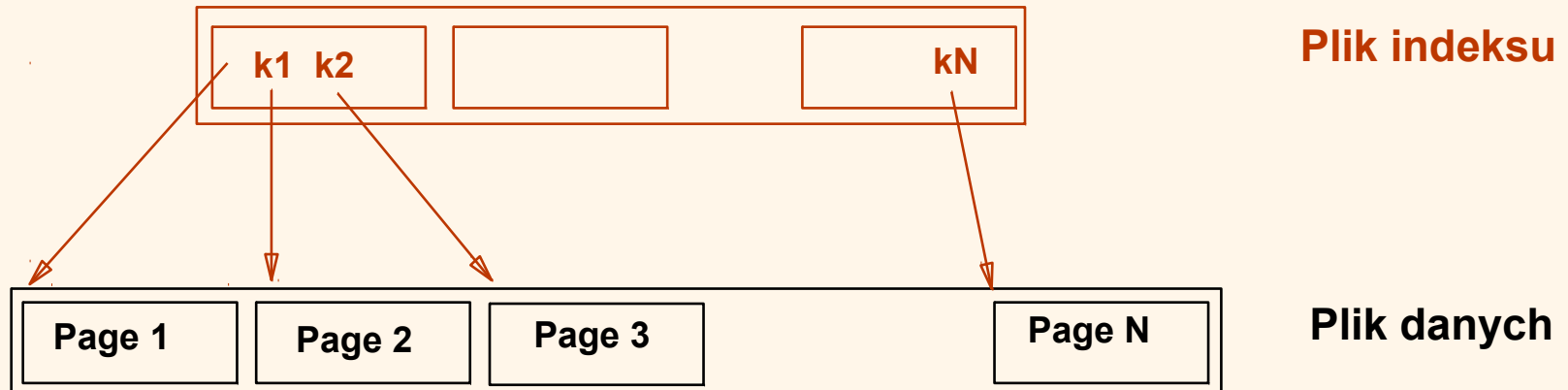
Złożone klucze wyszukiwania

- ▣ *Złożone klucze wyszukiwania:*
kombinacja pól np. <sal,age>.
 - Zapytanie równościowe:
 - ▣ age=20 and sal =75
 - Zapytanie zakresowe:
 - ▣ age < 20; age=20 and sal > 10
- ▣ Porządek leksykograficzny.



Wyszukiwanie zakresowe

- ``Wyznacz studentów ze średnią > 4.0 ``
 - Gdy dane w pliku posortowanym:
 - wyszukiwanie binarne aby znaleźć pierwszego takiego studenta;
 - przejdź plik wypisując pozostałych takich studentów.
 - Plik indeksu $k1 < k2 < \dots < kN$.

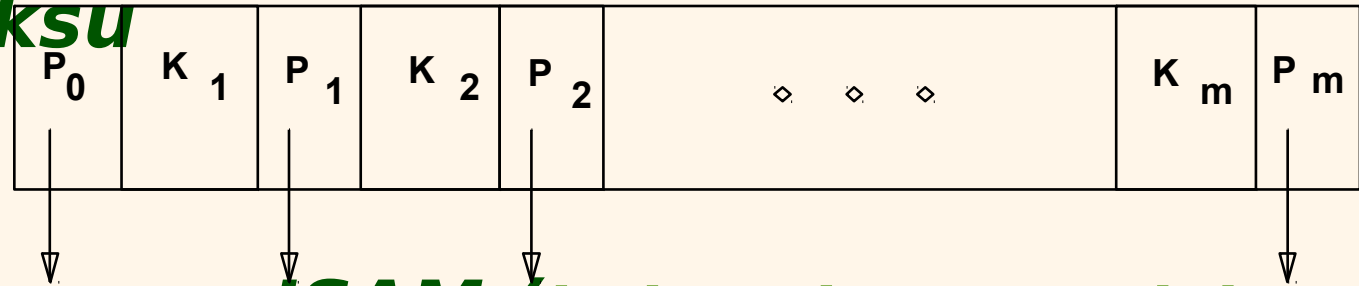


Wyszukiwanie binarne na mniejszym pliku indeksu!

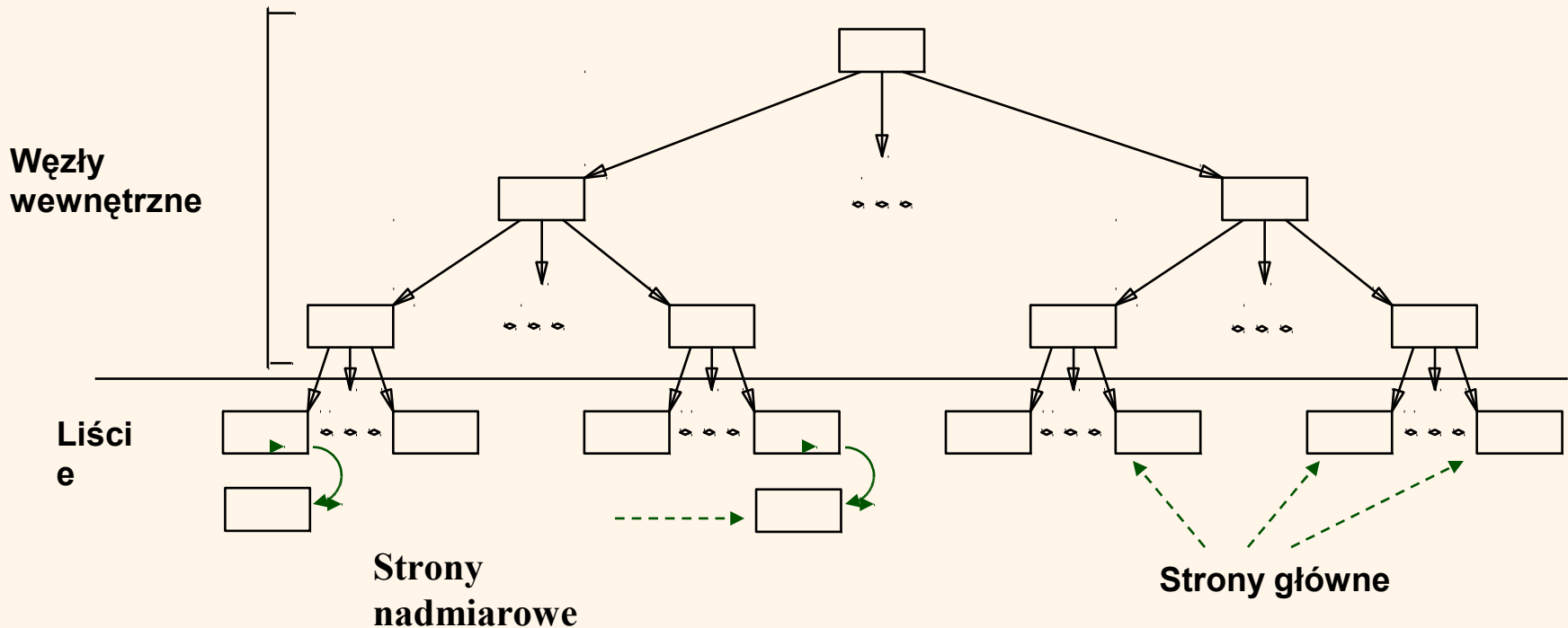
Pozycja indeksu

$K_1 < K_2 \dots < K_m$

Strona indeksu

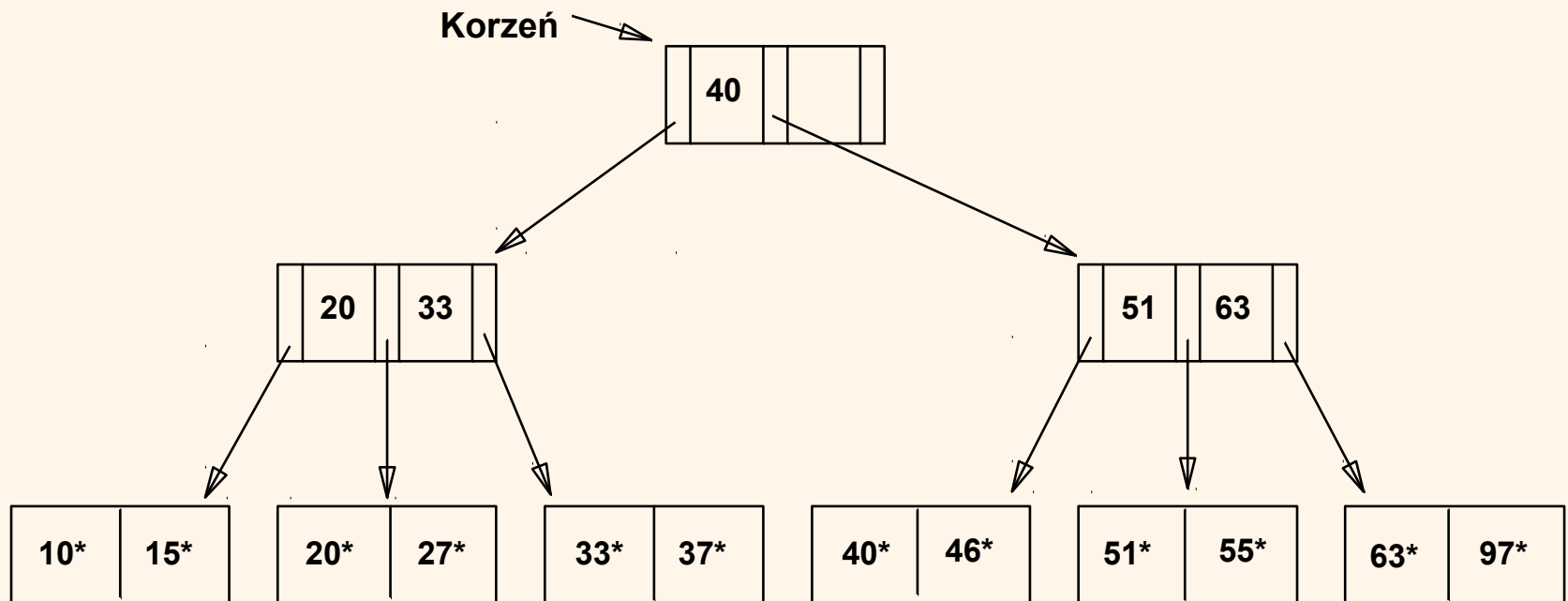


Statyczne drzewo ISAM (indexed sequential access method)



□ W liściach są pozycje danych.

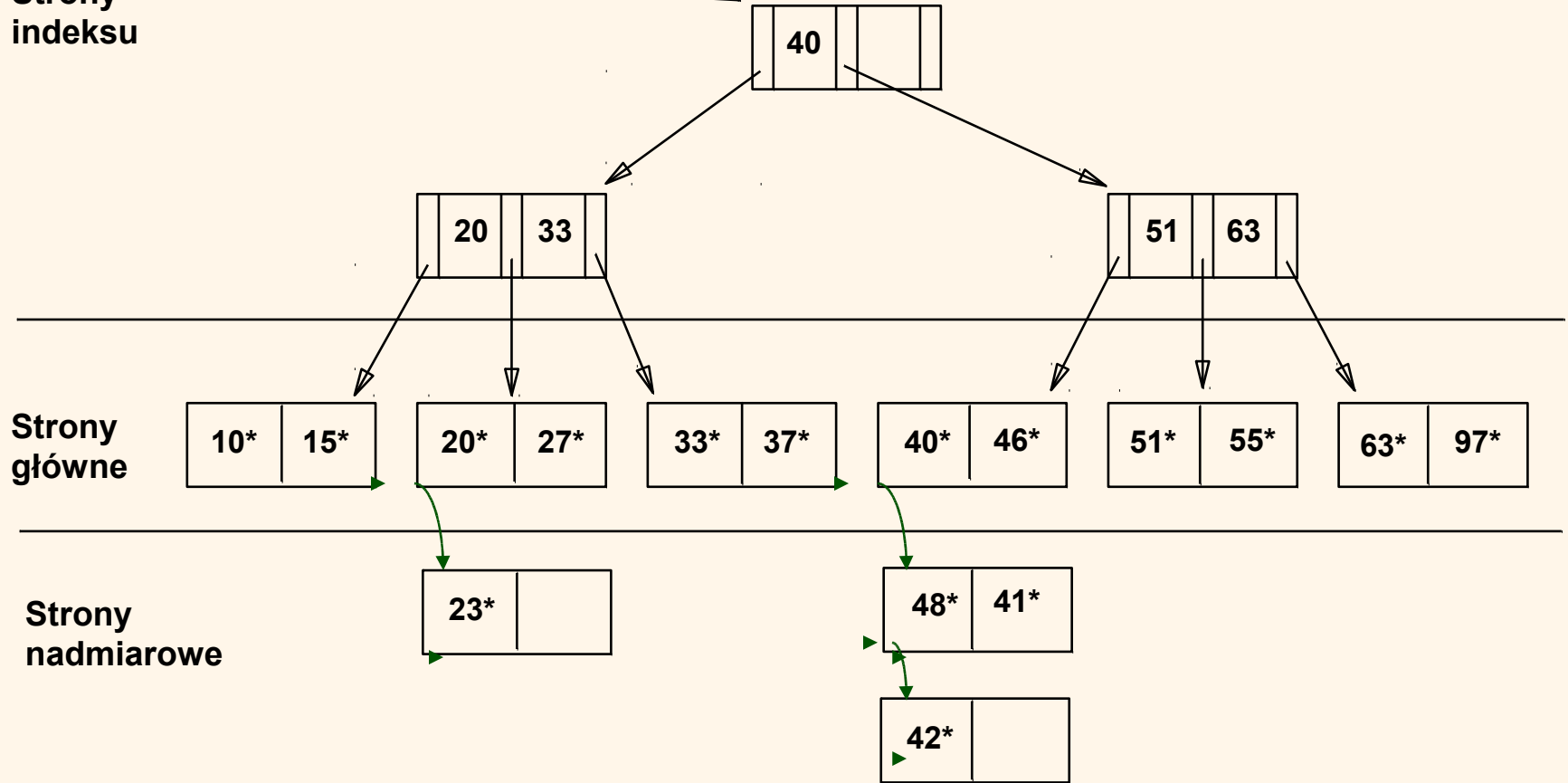
Drzewo ISAM



Wstawiamy 23, 48*, 41*,
42* ...*

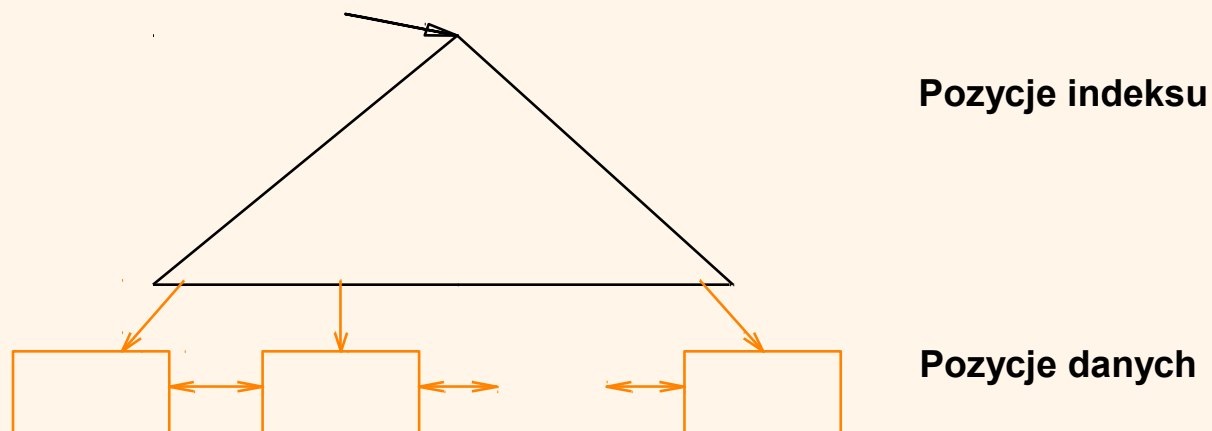
Strony
indeksu

Korzeń



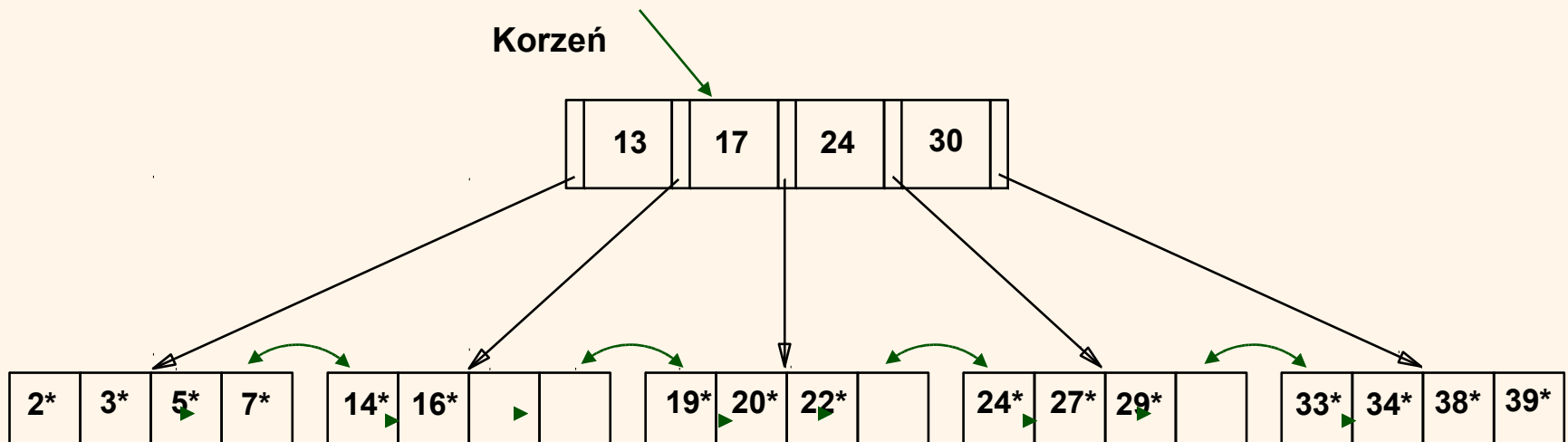
B+ drzewo

- Wstawianie/usuwanie w czasie rzędu $\log N$; wyważone względem wysokości. ($N = \#$ liści)
- Zajętość minimum 50% (z wyjątkiem korzenia). Każdy węzeł zawiera $\mathbf{d} \leq \underline{m} \leq 2\mathbf{d}$ pozycji. Parametr \mathbf{d} - stopień drzewa.
- Zapytania równościowe i zakresowe.



B+ drzewo

- Wyszukiwanie zaczyna się w korzeniu a porównania klucza wyszukiwania prowadzą do liścia tak jak dla ISAM.
- Wyszukiwanie 5*, 15*, $\geq 24^*$...



podstawie wyniku wyszukiwania 15, wiemy że 15* nie ma w drzewie*

Drzewa B+ w praktyce

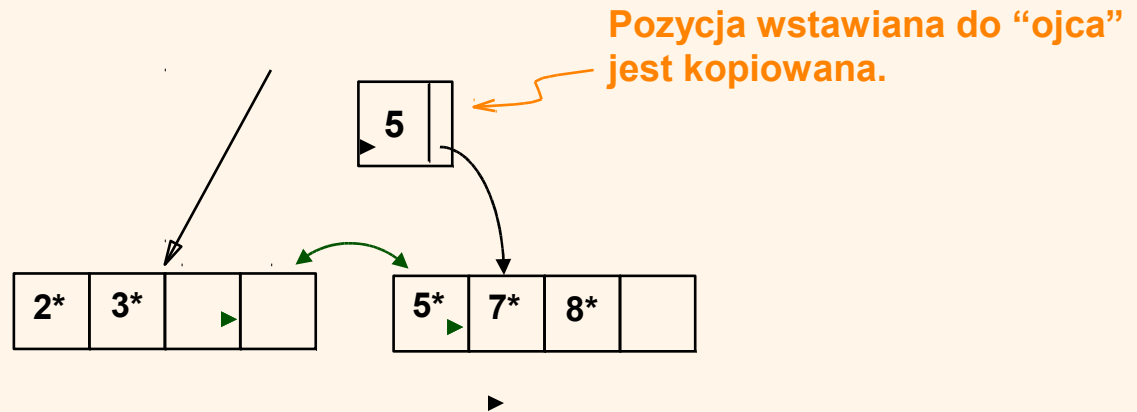
- ▣ Stopień $d=100$. Średnie zapełnienie: 67%.
- ▣ Typowa pojemność:
 - Wysokość 4: $133^4 = 312,900,700$ rekordów
 - Wysokość 3: $133^3 = 2,352,637$ rekordów
- ▣ Zwykle górne poziomy drzewa w cache:
 - Poziom 1 = 1 strona = 8 KB
 - Poziom 2 = 133 strony = 1 MB
 - Poziom 3 = 17,689 strony = 133 MB

Wstawienie pozycji danych do B+ drzewa

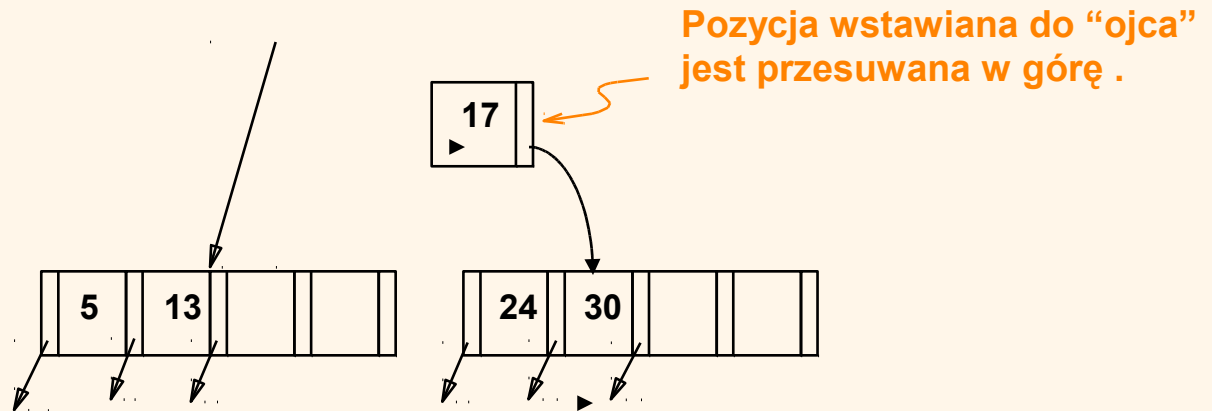
- Wyznacz odpowiedni liść L .
- Wstaw pozycję danych do L .
 - Jeśli mieści się – to *koniec!*
 - Wpp, podziel L (na L i nowy węzeł $L2$)
 - Rozdziel równo pozycje, skopiuj na wyższy poziom środkowy klucz.
 - Do “ojca” L wstaw pozycję indeksu z tym kluczem i wskaźnikiem wskazującym na $L2$.
- W razie potrzeby powtórz krok podziału rekurencyjnie.
- Kroki podziału mogą dojść do korzenia i w rezultacie drzewo może zwiększyć swoją wysokość o jeden.

Wstawianie 8*

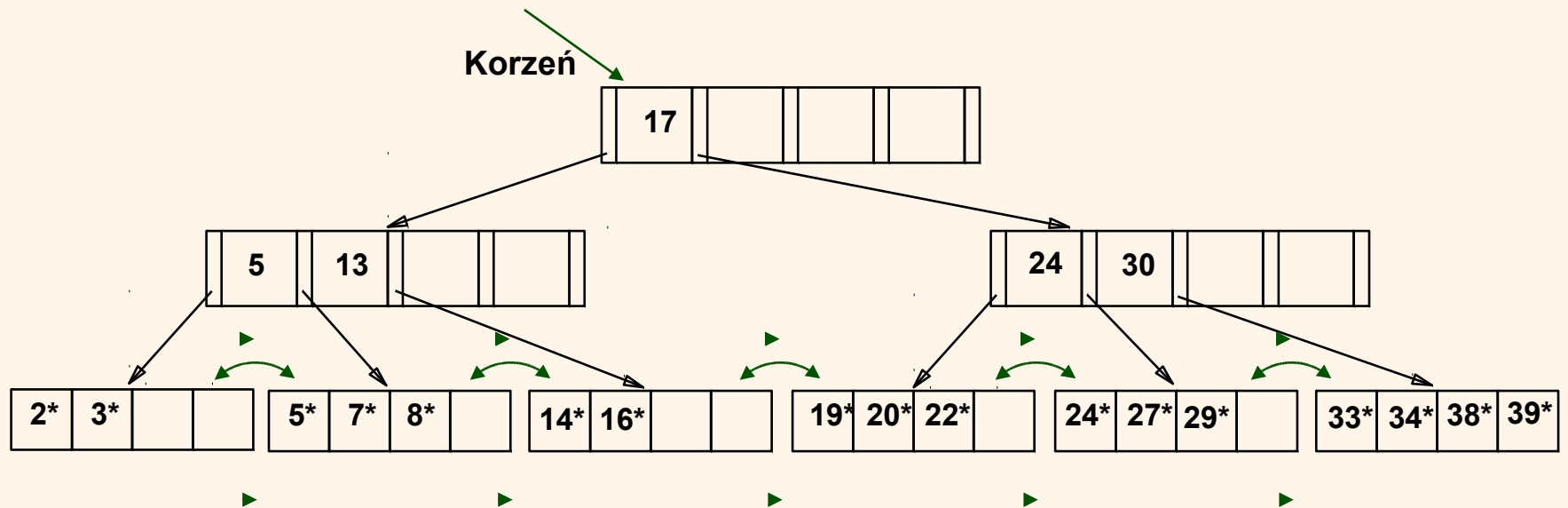
Podział liścia



Podział węzła wewnętrznego



Drzewo po wstawieniu 8*



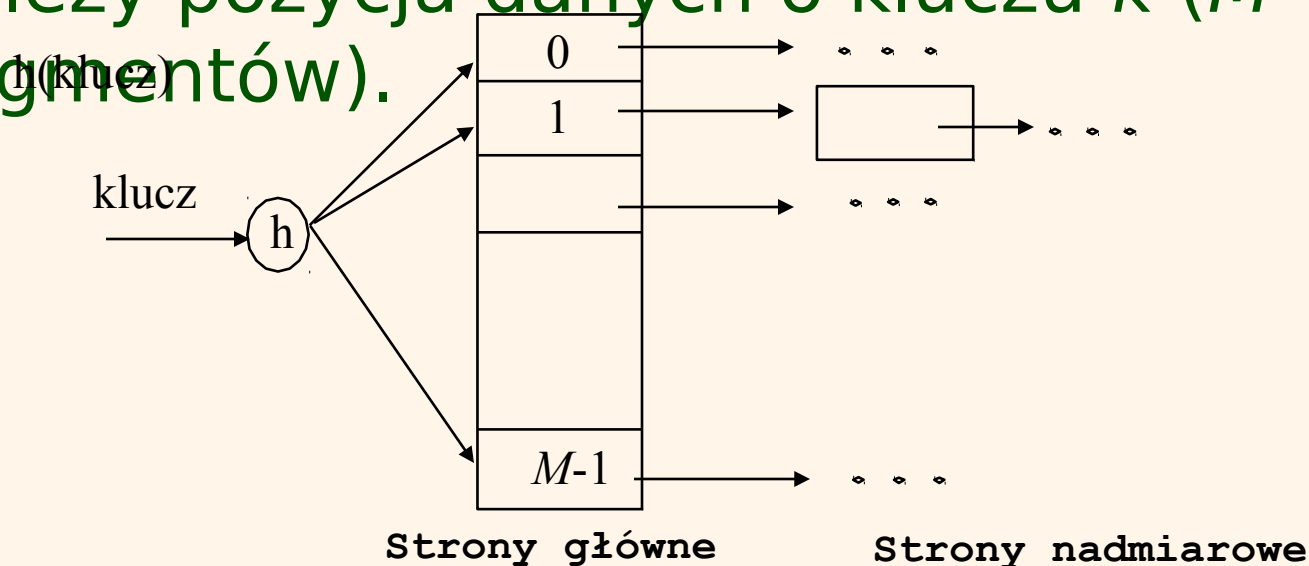
- Poprzedni korzeń uległ podziałowi. Wysokość drzewa zwiększyła się o 1.
- Możliwa redystrybucja pozycji danych w poziomie nie jest w praktyce realizowana.

Podsumowanie drzew

- Zastosowanie indeksów o strukturze drzewa:
 - Wyszukiwanie zakresowe.
 - Wyszukiwanie równościowe.
 - Sortowanie (np. ORDER BY).
- ISAM – struktura statyczna.
 - Modyfikowane są tylko liście.
 - Wymagane są strony nadmiarowe – mogące istotnie pogorszyć działanie algorytmów.
- B+ drzewo – struktura dynamiczna.
 - W praktyce wysokość zwykle ≤ 4 .

Haszowanie (statyczne)

- Ustalona alokacja stron głównych; alokowane dodatkowe strony nadmiarowe w razie potrzeby.
- $h(k) = k \bmod M$ = “segment” do którego należy pozycja danych o kluczu k ($M = \#$ segmentów).



Podsumowanie haszowania

- Mogą powstać długie łańcuchy nadmiarowych stron co prowadzi do pogorszenia działania.
 - *istnieją dynamiczne wersje haszowania – polegające na zwiększaniu liczby M segmentów przez ich podział.*
- Indeksy haszowane dobre przy wyszukiwaniu równościowym. Nie wspomagają wyszukiwania zakresowego ani sortowania.

WYKONYWANIE ZAPYTAŃ. PLANOWANIE INDEKSOW

Przygotował Lech Banachowski na podstawie:

- 1. Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, McGrawHill, 2000 (książka i slide'y).*
- 2. Lech Banachowski, Krzysztof Stencel, Bazy danych – projektowanie aplikacji na serwerze, EXIT, 2001.*

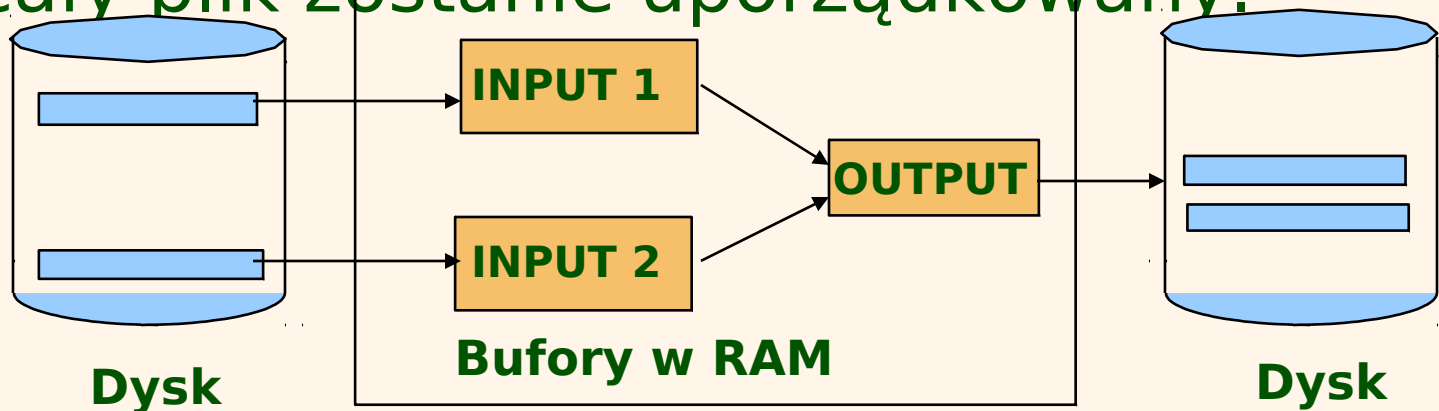
Zastosowania sortowania w bazach danych

- ORDER BY - dane są wymagane w pewnym porządku.
- Budowa indeksu - początkowego B+ drzewa dla wczytywanego zbioru rekordów.
- Złączanie tabel metodą *Sort-merge*.
- Realizacja DISTINCT, GROUP BY, UNION, EXCEPT - alternatywą haszowanie.

Problem: *posortować 1GB danych przy pomocy 10MB RAM.*

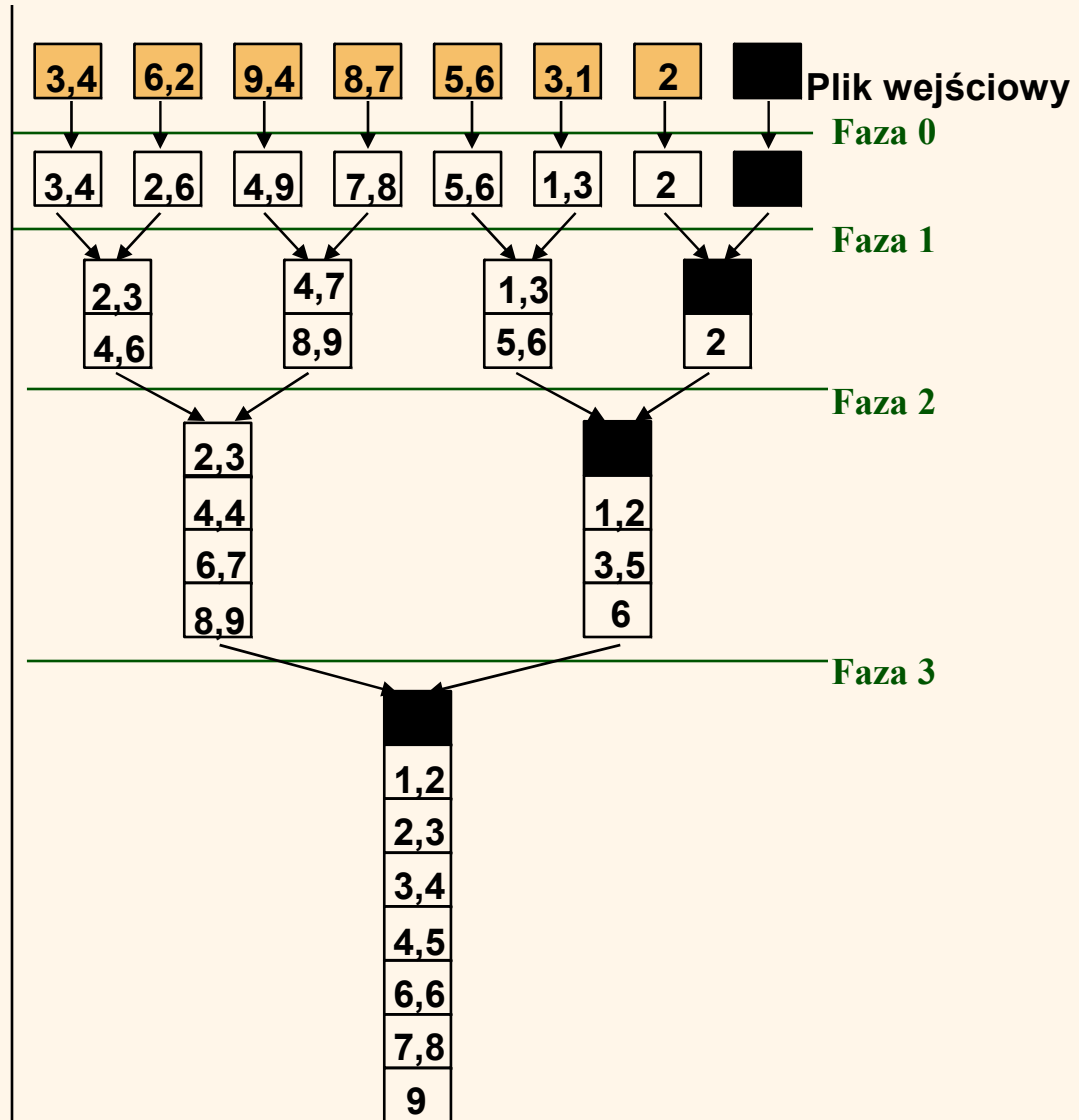
Sortowanie zewnętrzne (wielofazowe przez scalanie)

- Faza 0 – sortowanie rekordów w ramach stron: Wczytaj stronę, posortuj ją, zapisz na dysku.
- Faza 1,2,3 ..., itd: scalaj uporządkowane podpliki w większe uporządkowane podpliki aż cały plik zostanie uporządkowany.



Sortowanie wielotazowe przez scalanie

- W każdej fazie odczytujemy i zapisujemy każdą stronę w pliku.
- $N = \# \text{ stron} \Rightarrow \# \text{ faz} = \lceil \log_2 N \rceil + 1$
- Całkowity koszt = $N \log N$
- Idea: **Dziel i rządź**: sortuj podpliki i je scalaj.
- Zamiast dwóch buforów można użyć więcej.
- Praktycznie liczba faz ≤ 3 .

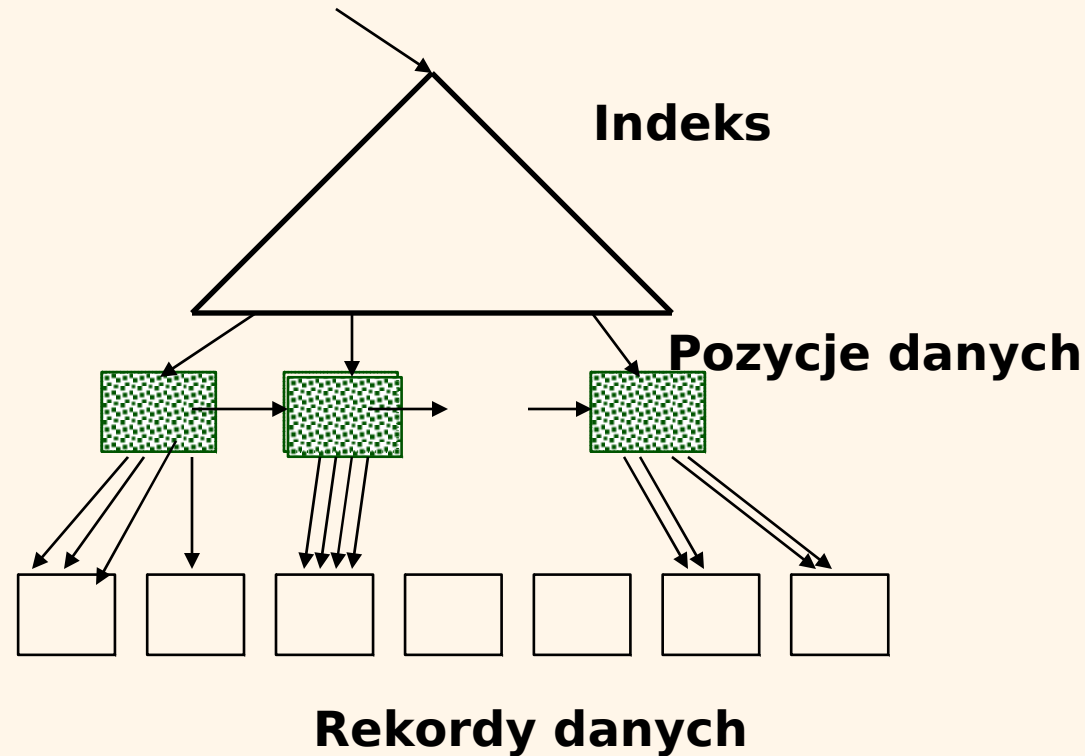


Sortowanie przy pomocy B+ drzewa

- Scenariusz: Tabela ma indeks na B+ drzewie względem kolumn sortowania.
- **Idea**: Przejść po liściach indeksu.
- ***Czy jest to dobra metoda?***
- Przypadki:
 - Indeks *pogrupowany* ***Bardzo dobra!***
 - Indeks *niepogrupowany* ***Może być bardzo zła!***

Indeks pogrupowany

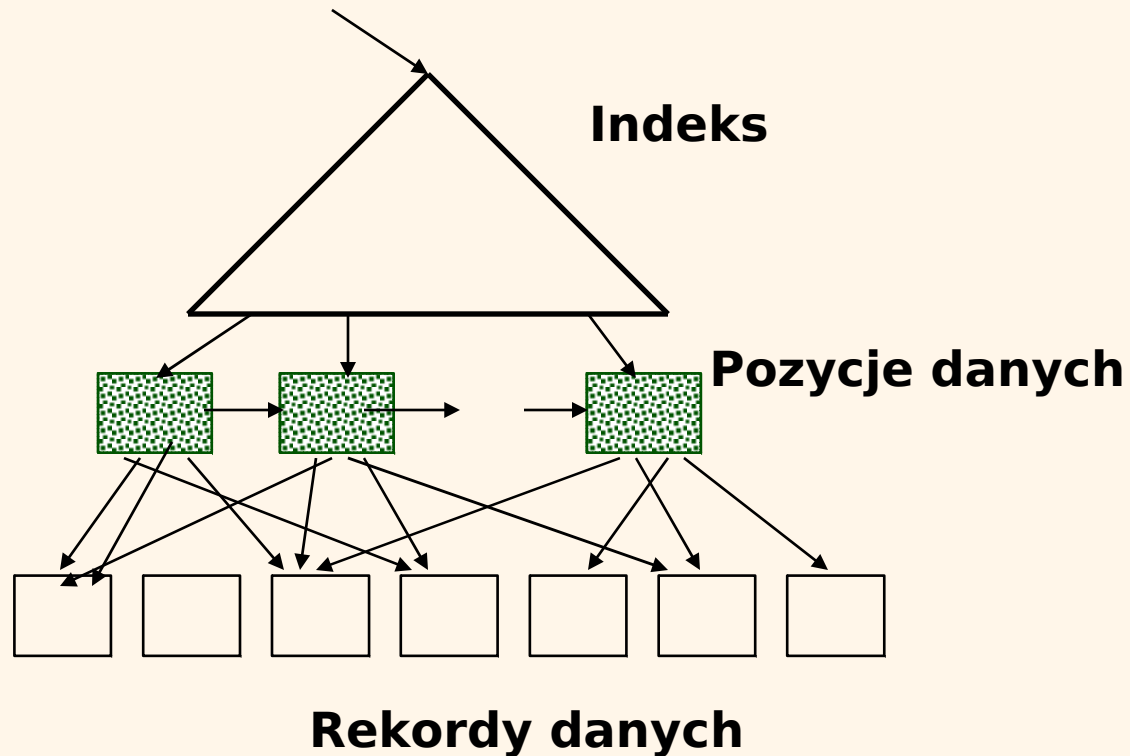
- Od korzenia przejdź do skrajnie lewego liścia a następnie sekwencyjnie w prawo po liściach.



- Zawsze lepsze od sortowania zewnętrznego!***

Indeks niepogrupowany

- **Ogólnie, jedna operacja We/Wy na rekord danych!**



Operatory relacyjne

- Selekcja Selekcja podzbioru wierszy (klauszula **WHERE**).
- Projekcja Pominięcie z wyniku niepotrzebnych kolumn (klauszula **SELECT**).
- Złączenie Złączenie relacji (tabel).
- Suma (UNION) Suma relacji (tabel).
- Agregacja (**SUM**, **MIN**, itd.) i **GROUP BY**.

* *Relacja = tabela*

Przykładowe tabele

Sailors (sid: integer, *sname*: string, *rating*: integer, *age*: real)
Reserves (sid: integer, bid: integer, day: dates, *rname*: string)

□ *Reserves R*:

- Każdy wiersz (rekord) - 40 bajtów, 100 wierszy na stronie, 1000 stron.

□ *Sailors S*:

- Każdy wiersz (rekord) - 50 bajtów, 80 wierszy na stronie, 500 stron.

Proste selekcje

```
SELECT *  
FROM Reserves R  
WHERE R.rname < 'C%'
```

- **Bez indeksu, nieposortowane:** Koszt jest $M = \text{\#stron w R.}$
- **Z indeksem na atrybucie selekcji:** Użyj indeksu, wyznacz pozycje danych, przejdź do rekordów.
 - Najlepiej gdy indeks haszowany dla selekcji równościowych oraz indeks na B+ drzewie dla selekcji zakresowych.

Użycie indeksu do selekcji

- Koszt zależy od liczby zwracanych wierszy (selektywności) i od tego czy indeks jest pogrupowany.
 - Wyznaczenie pozycji danych w indeksie a następnie sprowadzenie rekordów (może być kosztowne bez pogrupowania).
- *Ulepszenie dla niepogrupowanych indeksów:*
 1. Wyznacz odpowiednie pozycje danych.
 2. Posortuj je względem *rid*.
 3. Sprowadzaj rekordy w takim porządku. Każda potrzebna strona zostanie sprowadzona tylko raz.

Realizacja koniunkcji

- Obliczamy zbiory identyfikatorów rekordów spełniających dany warunek dla każdego indeksu.
- Dokonujemy przecięcia zbiorów identyfikatorów.
- Sprowadzamy rekordy ewentualnie stosując pozostałe warunki z klauzuli WHERE.
- Na przykład: *day < 8/9/94 AND bid = 5 AND sid = 3*.
Jeśli mamy indeks B+ drzewo na *day* oraz indeks na *sid*, wyznaczamy identyfikatory rekordów spełniających warunek *day < 8/9/94* używając pierwszego indeksu, identyfikatory rekordów spełniających warunek *sid = 3* używając drugiego indeksu, dokonujemy przecięcia, sprowadzamy rekordy i stosujemy warunek *bid = 5*.

Złożone warunki WHERE

*(day<8/9/94 AND rname='Paul') OR
bid=5 OR sid=3*

❖ Sprowadzamy warunek **WHERE** do
koniunkcyjnej postaci normalnej
conjunctive normal form (CNF):

*(day<8/9/94 OR bid=5 OR sid=3) AND
(rname='Paul' OR bid=5 OR sid=3)*

Selekcja ze złożonym warunkiem WHERE

1. Zastosować przejście całego pliku.
2. Jeśli jeden z czynników koniunkcji jest prostym warunkiem, określającym dostęp przez indeks (np. *sid = 8*), używamy tego indeksu i dla każdej otrzymanej krotki sprawdzamy pozostałe czynniki koniunkcji.
3. Metoda sumowania wyników składników koniunkcji np. gdy (*day < 8/9/02 OR rname = 'Joe'*): wyznaczamy zbiory *rid* dla *day < 8/9/02* i dla *rname = 'Joe'* – stosując indeksy; sortujemy względem *rid*, łączamy i wybieramy rekordy danych.

Projekcja

```
SELECT  DISTINCT  
        R.sid,  
        R.bid  
FROM    Reserves  
R
```

- Bez DISTINCT – przepisanie.
- Z DISTINCT – wymagane jest wyeliminowanie powtórzeń:
 - posortowanie;
 - haszowanie i eliminacja powtórzeń w ramach segmentów haszowania;
 - gdy atrybuty klauzuli SELECT tworzą indeks – wystarczy przejść tylko indeks.

Operatory zbiorowe

- **Przecięcie i iloczyn kartezjański** relacji są specjalnymi przypadkami złączenia.
- **Union (Distinct)** i **Except** są podobne do siebie.
 - Posortuj obie relacje (na kombinacji wszystkich atrybutów).
 - Dokonaj odpowiedniego scalenia wyników.
 - *Alternatywa*: Sortuj od razu razem obie relacje.
 - Zamiast sortowania można użyć haszowania.

Operacje agregacji (AVG, MIN itd.)

□ Bez grupowania:

- Na ogół trzeba rozważyć każdy wiersz.
- Gdy jest indeks, którego klucz wyszukiwania obejmuje wszystkie atrybuty występujące w klauzulach SELECT i WHERE, wystarczy przejrzeć indeks (strategia *tylko-indeks*).

□ Z grupowaniem GROUP BY:

- Posortuj względem wartości atrybutów GROUP BY, przejdź po rekordach w każdej grupie licząc wartości funkcji sumarycznych – w tym celu można użyć pogrupowany indeks na B+ drzewie. (Ulepszenie: liczenie wartości w trakcie sortowania.)
- Gdy jest indeks, którego klucz wyszukiwania obejmuje wszystkie atrybuty występujące w klauzulach SELECT, WHERE i GROUP BY, wystarczy przejrzeć indeks (strategia *tylko-indeks*).
- Zamiast sortowania można użyć haszowania.

Złączenia równościowe z jedną kolumną złączenia

```
SELECT *  
FROM   Reserves R, Sailors S  
WHERE  R.sid = S.sid
```

- Bezpośrednie podejście: generuj wszystkie kombinacje wierszy i stosuj selekcję.
- $M = \# \text{stron w } R$, $p_R = \# \text{wierszy na stronie dla } R$,
 $N = \# \text{stron w } S$, $p_S = \# \text{wierszy na stronie dla } S$.
 - W przykładzie: R – Reserves, S – Sailors.

Algorytm Nested Loops Join

```
foreach row r in R do  
  foreach row s in S do  
    if  $r_i == s_j$  then add  $\langle r, s \rangle$  to result
```

- Dla każdego wiersza *zewnętrznej* relacji R, przeglądamy wszystkie wiersze *wewnętrznej* relacji S.
 - Koszt: $M + p_R * M * N = 1000 + 100 * 1000 * 500$ We/Wy.
- Oczywiste ulepszenie: Dla każdej strony w R, sprowadź każdą stronę w S
 - Koszt: $M + M * N = 1000 + 1000 * 500$.
 - Gdy mniejsza relacja (S) jest zewnętrzna, koszt = $500 + 500 * 1000$.

Algorytm Index Nested Loops

Join

foreach row r in R do

foreach row s in S where $r_i == s_j$ do

add $\langle r, s \rangle$ to result

/ dla r w R użyj indeksu na S do wyznaczenia s w S : $r == s$ */*

- Gdy jest indeks (najlepiej haszowany) na kolumnie złączenia relacji wewnętrznej (S), zastosuj indeks.
 - Koszt: $M + (M * p_R) * \text{koszt wyznaczenia pasujących wierszy w } S$.
- Dla każdego wiersza w R : koszt wyszukania pozycji w indeksie dla S jest ok. 1.2 dla indeksu haszowanego; 2-4 dla B+ drzewa. Mając daną pozycję danych, koszt wyznaczenia pasujących wierszy w S zależy od tego czy indeks jest pogrupowany.
 - Indeks pogrupowany: +1 We/Wy (zwykle);
 - Indeks niegrupowany: +1 We/Wy dla każdego pasującego wiersza w S .
- Razem w przykładzie koszt: $1000 + 1000 * 100(1.2 + 1)$ We/Wy.

Algorytm Sort-Merge Join

- Posortuj R i S na kolumnach złączenia, następnie scal odpowiadające sobie wiersze w R i S. Przy scalaniu na ogół każda z posortowanych relacji R i S jest przeglądana raz (liniowo).

Przykład Sort-Merge Join

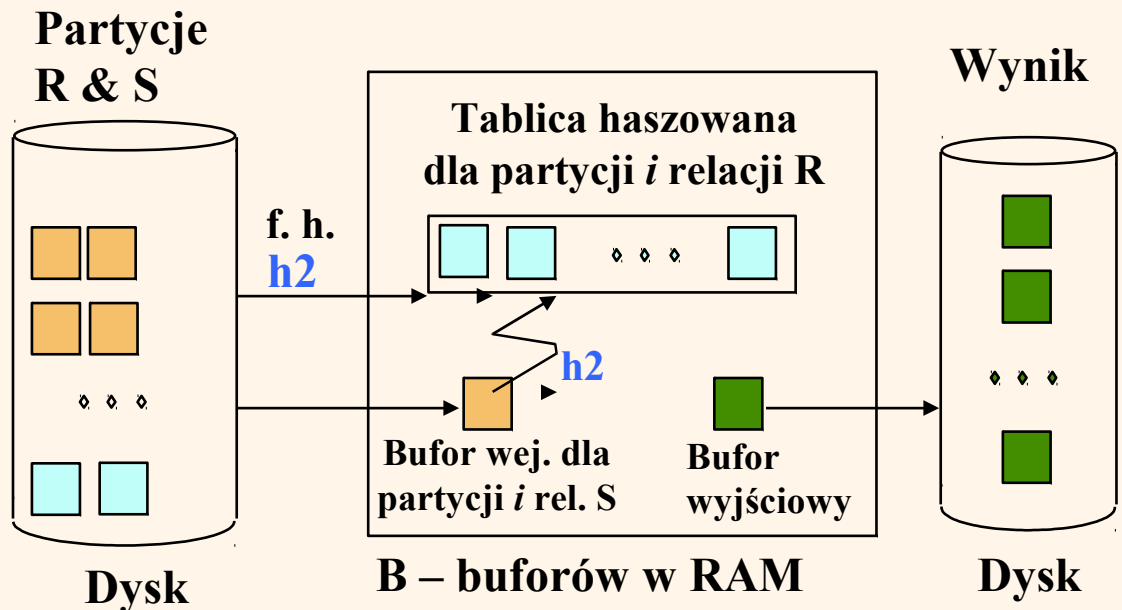
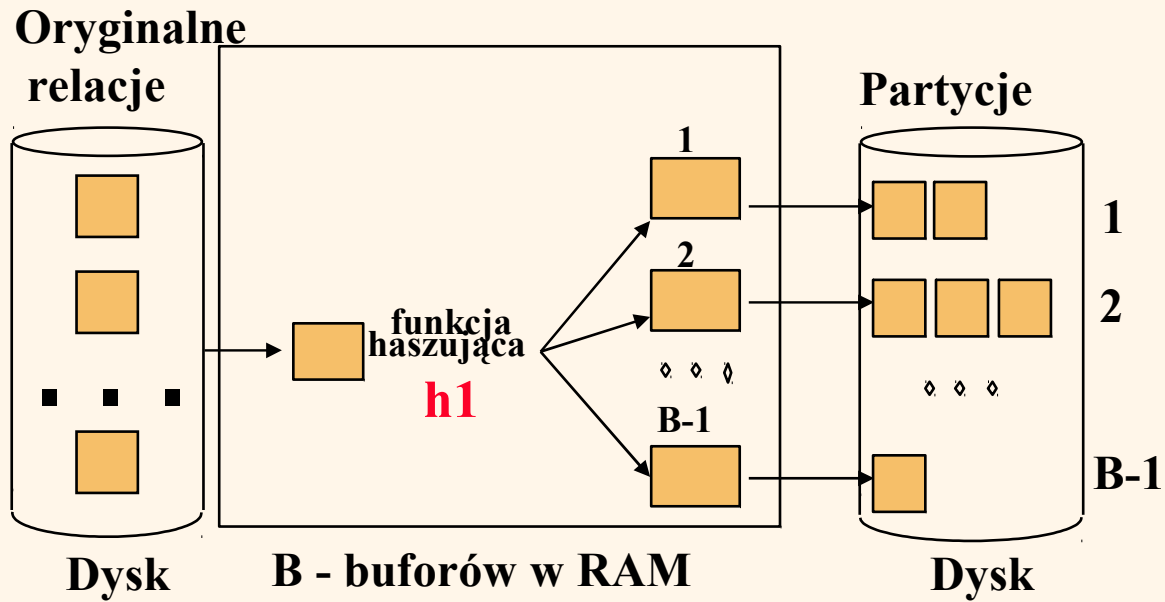
<u>sid</u>	sname	rating	age	<u>sid</u>	<u>bid</u>	<u>day</u>	rname
22	dustin	7	45.0	28	103	12/4/96	guppy
28	yuppy	9	35.0	28	103	11/3/96	yuppy
31	lubber	8	55.5	31	101	10/10/96	dustin
44	guppy	5	35.0	31	102	10/12/96	lubber
58	rusty	10	35.0	31	101	10/11/96	lubber
				58	103	11/12/96	dustin

- Koszt: $M \log M + N \log N + (M+N)$ - w praktyce rzędu liniowego względem $(M+N)$.
- W przykładzie: $2000 + 1000 + 1000 + 500 = 4500$.

Algorytm Hash-Join

Podziel obie relacje R i S na partycje względem wartości funkcji haszującej **h1** na kolumnach złączenia: wiersze R w partycji *i* wystarczy złączyć z wierszami S w partycji *i*.

Wczytaj partycję *i* relacji R dokonując haszowania przy pomocy f.h. **h2** (\neq **h1**!). Wczytaj elementy partycji *i* w S, stosuj **h2** i uzgadniaj z R.



Koszt algorytmu Hash-Join

- Podział na partycje - $2(M+N)$. Uzgadnianie - $M+N$ We/Wy.
- Porównanie Sort-Merge Join i Hash Join:
 - Obie metody mają ten sam koszt $3(M+N)$ We/Wy.
 - Hash Join lepszy przy większej różnicy rozmiarów; łatwy do zrównoleglenia .
 - Sort-Merge mniej wrażliwy na losowość danych; wynik posortowany.

Podsumowanie - realizacja operatorów

- Zaleta relacyjnych SZBD – *zapytania* złożone z kilku *bazowych operatorów*; implementacje tych operatorów można dokładnie dostroić.
- Wiele alternatywnych metod implementacyjnych.
- Dla konkretnego zapytania dla każdego występującego w nim operatora trzeba rozważyć dostępne opcje i wybrać najlepszą korzystając z dostępnych statystyk. Jest to zadanie *optymalizacji zapytania*.

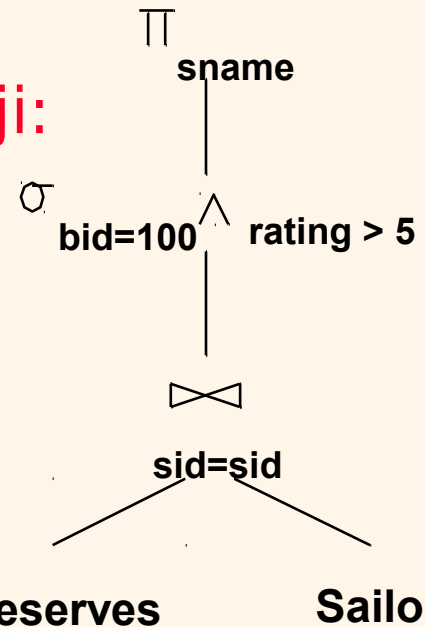
Optymalizacja zapytań

- Plan: Algorytm wykonania zapytania – np. w postaci drzewa.
 - Dla danego zapytania: jakie plany są rozpatrywane?
 - Jak oszacować koszt planu?
- **Idealnie**: Chcemy znaleźć najlepszy plan.
Praktycznie: Staramy się unikać złych planów!

Przykład

```
SELECT S.sname
FROM Reserves R, Sailors S
WHERE R.sid=S.sid AND
      R.bid=100 AND S.rating>5
```

Drzewo instrukcji:



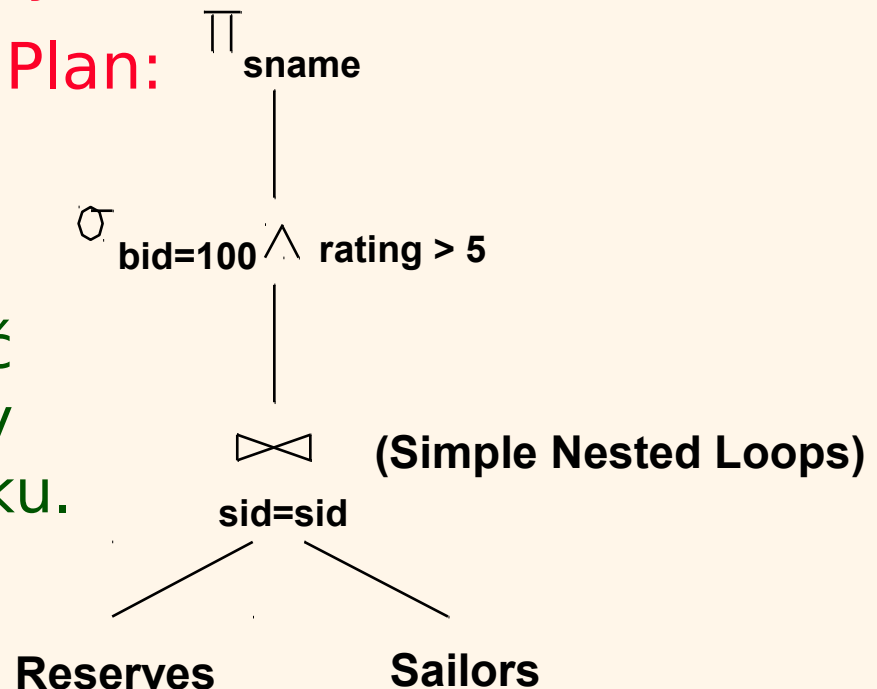
❑ Koszt: $1000 + 500 * 1000$ We/Wy

❑ Nie wykorzystuje:

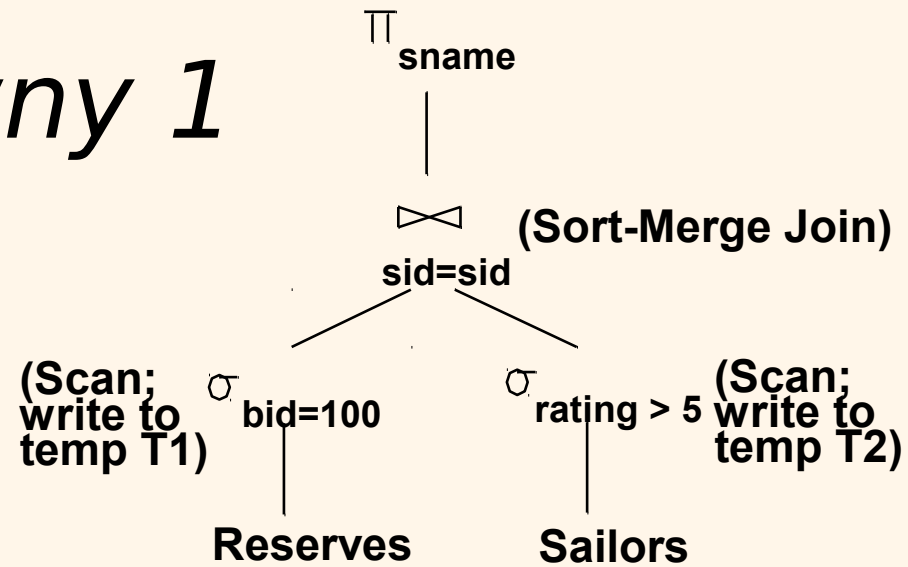
- wcześniejszego wykonywania selekcji,
- indeksów.

❑ Cel optymalizacji: Wyznaczyć inne bardziej efektywne plany obliczenia tego samego wyniku.

Plan:



Plan alternatywny 1 (bez indeksów)



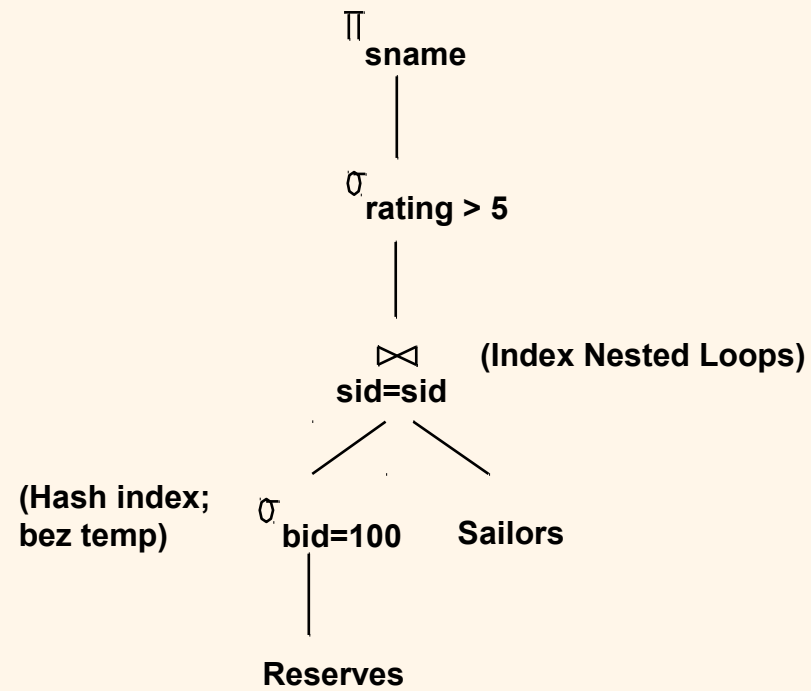
□ **Główna różnica:** selekcje wcześniej.

□ **Koszt planu:**

- Scan Reserves (1000) + write to temp T1 (ok. 10 stron rezerwacji przy 100 łódkach).
- Scan Sailors (500) + write to temp T2 (ok. 250 stron (połowa) przy 10 wartościach atrybutu rating).
- Sort T1 ($2 \times 2 \times 10$), sort T2 ($2 \times 3 \times 250$), merge (10+250).
- W sumie: 3560 We/Wy.

Plan alternatywny 2 (z indeksami)

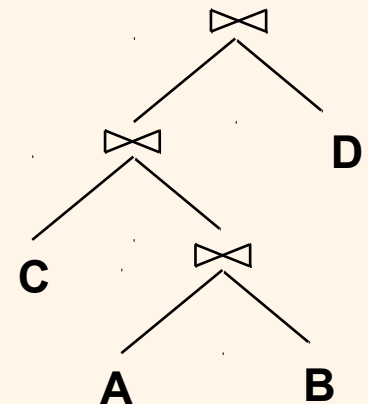
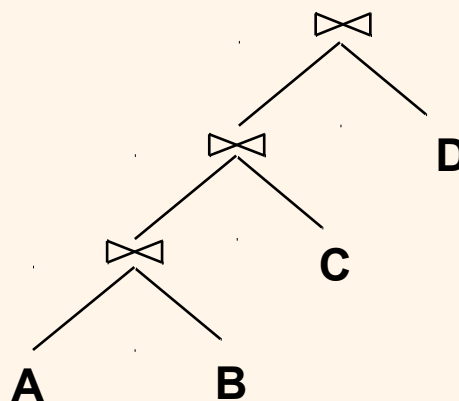
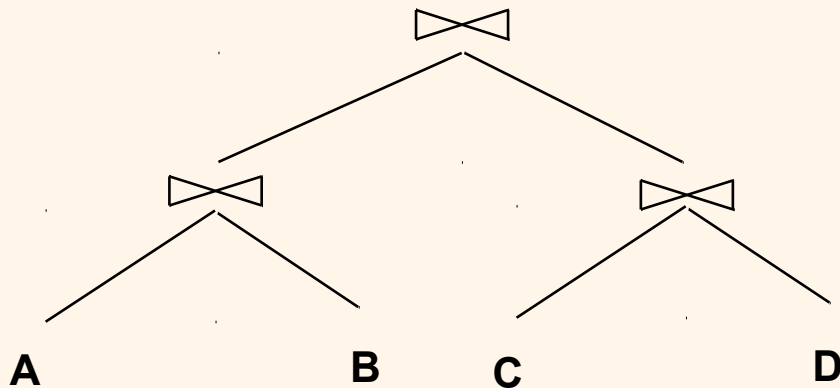
- Z indeksem pogrupowanym na Reserves(*bid*), dostajemy $100,000/100 = 1000$ wierszy na $1000/100 = 10$ stronach.
- INL bez zapisywania **wyniku** selekcji jako tymczasowej relacji.



- Kolumna złączenia *sid* jest kluczem dla tabeli *Sailors*. Wystarczy indeks niepogrupowany.
- Decyzja aby nie dokonywać selekcji *rating*>5 przed złączeniem, ponieważ jest dostępny indeks *Sailors*(*sid*).
- **Koszt:** Selekcja wierszy *Reserves* (10 We/Wy); dla każdego z 1000 wierszy *Reserves* trzeba uzyskać odpowiadające wiersze tabeli *Sailors* ($1000 \cdot 1.2$ gdy indeks haszowany jest wewnętrzny – połączony z tabelą, w p.p. $1000 \cdot 2.2$).

generowanie przez optymalizator planów wykonania zapytania

- Najpierw analiza dostępu do poszczególnych relacji z możliwością zastawiania selekcji, indeksu itd.
- Na ogół ograniczenie do: drzew skierowanych w lewo.
 - Drzewa skierowane w lewo dają plany umożliwiające "potokowe" wykonanie zapytania *"w miejscu"* tj. bez tymczasowych plików.
 - Podstawa: przemienność i łączność operatora złączenia.



Przykład

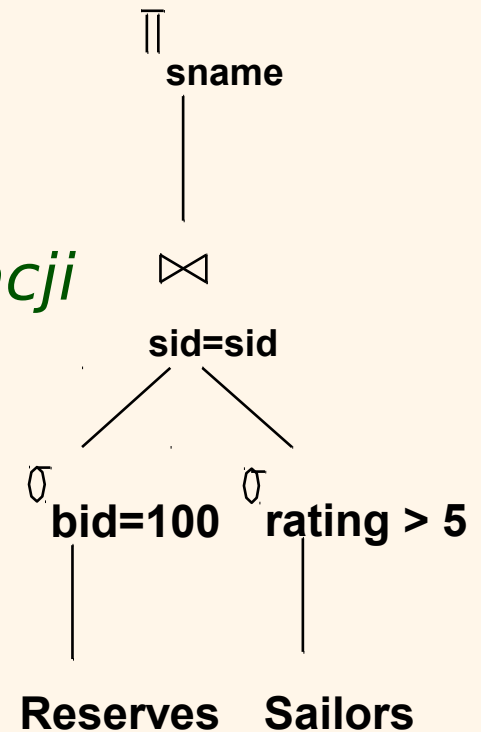
- **Faza 1:** Analiza planów dostępu do relacji *Sailors* i *Reserves*:

Sailors:

B+ drzewo na *rating*
Hash na *sid*
Scan

Reserves:

B+ drzewo na *bid*
Scan



- **Faza 2:** Rozpatrujemy każdy plan z Fazy 1 jako określający relację zewnętrzną złączenia i badamy jak dokonać złączenia z drugą relacją (NLJ, INLJ, SMJ, HJ) liczymy orientacyjny koszt korzystając ze statystyk zebranych przez system jak liczba wierszy, liczba stron dla plików z danymi i plików indeksów.

□ np., dla *Reserves*: Mając dane *sid* z *Reserves* można użyć indeksu haszowanego na *Sailors(sid)* aby dokonać złączenia obu relacji

Podzapytania

- Podzapytania są optymalizowane niezależnie.
- Główne zapytanie jest optymalizowane z brany pod uwagę kosztem „wywoływanych” podzapytań.
- Ewentualnie sprowadzane do złączeń i optymalizowane łącznie.

Ogólne strategie optymalizacyjne

- Selekcje wykonuj jak najwcześniej.
- Staraj się łączyć selekcje z iloczynem kartezjańskim, w celu zidentyfikowania rodzaju złączenia relacji.
- Wykonuj jednocześnie ciągi operacji jednoargumentowych takich jak selekcje i rzuty.
- Wybierz plan wykonania działający “w miejscu” bez pomocniczych relacji (drzewa skierowane w lewo).
- Wyszukuj wspólne podwyrażenia i wykonuj je tylko raz.
- Przetwórz wstępnie plik we właściwy sposób (indeksy, sortowanie, haszowanie).
- Gromadź statystyki ilościowe dotyczące tabel, kolumn i indeksów – w tym *histogramy* – dystrybucja wartości w kolumnie.
- Przed przystąpieniem do realizacji zapytania dokonaj analizy możliwych opcji z oszacowaniem ich kosztu.

Pole działania aplikacji bazodanowej (workload)

- Najważniejsze zapytania i jak często będą używane.
- Najważniejsze aktualizacje i jak często będą używane.
- Pożądana szybkość działania tych zapytań i aktualizacji.

Wybór indeksów – analiza zapytań

- Dla każdego zapytania w projektowanej aplikacji:
 - Jakich relacji i atrybutów dotyczą?
 - Które atrybuty występują w warunkach ograniczających a które w warunkach złączenia?
Jak bardzo selektywne są te warunki?
- Podobnie dla instrukcji INSERT/DELETE/UPDATE.

Decyzje

- ▣ Jakie założyć indeksy?
- ▣ Jakiego rodzaju?
 - Pogrupowany? Hasz/B+ drzewo/Bitmap?
 - Dynamiczny/statyczny? Czy powinniśmy zmienić schemat tabel?
 - Inaczej pogrupować atrybuty w ramach relacji?
 - Normalizacja, denormalizacja (pionowy podział)?
 - Poziomy podział (np. tabelę *Osoby* na osobne tabele *Pracownicy* i *Studenti*?)
 - Perspektywa zmaterializowana z wynikami obliczeń (np. statystyka operacji na kontach bankowych)?
- ▣ Czy połączyć zapis kilku tabel w klaster? Złączenie – szybsze; operacje na pojedynczych tabelach wolniejsze niż bez klastra.

Wybór indeksów

- Zaczynając od najważniejszych zapytań przeanalizuj plany ich wykonania – czy nie powinniśmy dodać nowy indeks do już wybranych?
- Rozpatrz jaki wpływ będzie miał ten indeks na operacje INSERT/DELETE/UPDATE oraz na ilość potrzebnego miejsca na dysku?

Wybór indeksów

- Atrybuty w klauzuli WHERE są kandydatami na klucze wyszukiwania w indeksie.
 - Równość sugeruje indeks haszowany.
 - Przedział wartości sugeruje B+ drzewo.
 - Indeks pogrupowany jest użyteczny przy zapytaniach zakresowych ale także przy mało-selektywnych zapytaniach równościowych.
- Jeden indeks powinien być użyteczny dla wielu zapytań.
- Ponieważ tylko jeden indeks może być pogrupowany dla jednej relacji, jeśli zachodzi potrzeba jego użycia, wybierz go biorąc pod uwagę najważniejsze zapytania.

Wybór indeksów

- Gdy klauzula WHERE zawiera kilka warunków, należy rozpatrzyć możliwość założenia indeksu o wieloatrybutowym kluczu wyszukiwania.
 - Przy warunkach zakresowych istotna jest kolejność atrybutów w kluczu wyszukiwania.
 - Indeksy takie mogą czasem umożliwić zastosowanie strategii “*tylko-indeks*” – wtedy pogrupowanie nie ma znaczenia.
- Przy rozpatrywaniu warunku złączenia:
 - Indeks haszowany na relacji wewnętrznej jest dobry dla metody Index Nested Loops.
 - Powinien być pogrupowany jeśli kolumna złączenia nie jest kluczem dla relacji wewnętrznej a wiersze relacji wewnętrznej mają się znaleźć w wyniku.
 - *Pogrupowany indeks na B+ drzewie* względem kolumn złączenia jest dobry dla metody Sort-Merge.

Przykład 1

```
SELECT E.ename, D.mgr  
FROM Emp E, Dept D  
WHERE D.dname='Toy' AND E.dno=D.dno
```

- Indeks haszowany na *D.dname* wspomaga selekcję *D.dname='Toy'* (D – relacja zewnętrzna).
- Indeks haszowany na *E.dno* wspomaga złączenie (E – relacja wewnętrzna). Powinien być pogrupowany, ponieważ spodziewamy się wybrania wielu wierszy z E.

Przykład 2

```
SELECT E.ename, D.mgr  
FROM   Emp E, Dept D  
WHERE  E.sal BETWEEN 10000 AND 20000  
       AND E.hobby='Stamps' AND E.dno=D.dno
```

- Emp E – relacja zewnętrzna złączenia. Dept D – relacja wewnętrzna złączenia.
 - Stąd indeks haszowany na *D.dno*.
- Jaki indeks na relacji Emp?
 - Albo B+ drzewo na *E.sal* albo indeks haszowany na *E.hobby* – zależy od ich selektywności.

Przykład 3

```
SELECT E.dno, COUNT (*)  
FROM Emp E  
WHERE E.age>20  
GROUP BY E.dno
```

- Zapytanie GROUP BY.
 - Indeks pogrupowany na B+ drzewie dla *E.dno*.
- Zapytanie równościowe i duplikaty.
 - Indeks pogrupowany na *E.hobby*.

```
SELECT E.dno  
FROM Emp E  
WHERE E.hobby='Stamps'
```

Podsumowanie

- Wybór indeksów ma istotny wpływ na szybkość wykonywania zapytań.
 - Aktualizacja pól wyszukiwania w indeksach zwalnia INSERT/DELETE/UPDATE.
 - Wybieraj indeksy, które wspomagają wykonywanie wielu zapytań.
 - Buduj indeksy umożliwiające strategie *tylko-indeks*.
 - Tylko jeden indeks może być pogrupowany dla jednej relacji.
 - Kolejność pól w kluczach wielo-atrybutowych może być istotna.
- Od czasu do czasu trzeba przebudowywać statyczne indeksy.
- Od czasu do czasu trzeba odświeżać statystyki.
- Sprawdzaj plan wybrany przez optymalizator - ewentualnie zmień indeks, zapis zapytania. Ewentualnie użyj wskazówek do optymalizatora (*Oracle hint*).
- Unikaj podzapytań, DISTINCT, wyrażeń (może być trudno użyć indeks), tymczasowych tabel.

Λεωνίδας: ὦ ξεῖν', ἀγγέλλειν
Λακεδαιμονίοις ὅτι τῇδε
κείμεθα τοῖς κείνων ῥήμασι
πειθόμενοι

Leonidas: *Przechodniu,
powiedz Sparcie, tu leżym
jej syny. Prawom jej do
ostatniej posłuszni godziny.*

DBA pod Termopilami

Hopliti: *Co się tak, pik, patrzysz, nas tu nie ma.*

Leonidas: *Chyba już pójdę, nie? Co tu będę tak sam
siedział...*



Przykłady

<E.dno>

```
SELECT D.mgr
FROM Dept D, Emp E
WHERE D.dno=E.dno
```

<E.dno,E.eid>

```
SELECT D.mgr, E.eid
FROM Dept D, Emp E
WHERE D.dno=E.dno
```

<E.dno>

```
SELECT E.dno, COUNT(*)
FROM Emp E
GROUP BY E.dno
```

<E.dno,E.sal>

```
SELECT E.dno, MIN(E.sal)
FROM Emp E
GROUP BY E.dno
```

B+ drzewo!

<E. age,E.sal>

lub


<E.sal, E.age>


B+ drzewo!

```
SELECT AVG(E.sal)
FROM Emp E
WHERE E.age=25 AND
E.sal BETWEEN 3000 AND 5000
```

- Pewne zapytania można wykonać bezpośrednio z indeksu (bez przechodzenia do relacji).

Pułapki eliminacji podzapytań

<pre>SELECT DISTINCT * FROM Sailors S WHERE S.sname IN (SELECT Y.sname FROM YoungSailors Y)</pre>		<pre>SELECT DISTINCT S.* FROM Sailors S, YoungSailors Y WHERE S.sname = Y.sname</pre>
---	--	--

<pre>SELECT * FROM Sailors S WHERE S.sname IN (SELECT DISTINCT Y.sname FROM YoungSailors Y)</pre>		<pre>SELECT S.* FROM Sailors S, YoungSailors Y WHERE S.sname = Y.sname</pre>
---	--	---

Pułapki eliminacji podzapytań



```
SELECT dname FROM Department D
WHERE D.num_emps >
      (SELECT COUNT(*) FROM Employee E
       WHERE D.building = E.building)
```

```
CREATE VIEW Temp (empcount, building) AS
SELECT COUNT(*), E.building
FROM Employee E
GROUP BY E.building
```

```
SELECT dname
FROM Department D, Temp
WHERE D.building = Temp.building
AND D.num_emps > Temp.empcount;
```

A gdy tabela Employee jest pusta?

Zarządzanie transakcjami

Przygotował Lech Banachowski na podstawie:

- 1. Raghu Ramakrishnan, Johannes Gehrke, Database Management Systems, McGrawHill, 2000 (książka i slide'y).*
- 2. Lech Banachowski, Krzysztof Stencel, Bazy danych – projektowanie aplikacji na serwerze, EXIT, 2001.*

Transakcje

- Współbieżne wykonywanie programów użytkowników jest istotne dla szybkości działania aplikacji bazodanowych.
 - Dostęp do danych na dysku jest częsty i względnie wolny, więc procesor może współbieżnie wykonywać kilka programów.
- Transakcja jest abstrakcyjną reprezentacją programu użytkownika: ciągiem odczytów i zapisów.
 - Współbieżność uzyskuje się przez przeplecenie odczytów i zapisów różnych transakcji. Efekt powinien być taki jakby transakcje były wykonywane niezależnie od siebie.

Aksjomaty wykonywania transakcji ACID

1. *Atomowość (niepodzielność)* - albo wszystkie akcje wchodzące w skład transakcji są wykonywane albo żadna.
2. *Spójność* - po wykonaniu transakcji stan bazy danych powinien być spójny (pod warunkiem, że każda instrukcja transakcji zachowuje spójność).
3. *Izolacja* - wynik działania transakcji powinien być taki sam, jakby od chwili rozpoczęcia transakcji nie działała na wspólnych danych żadna inna transakcja. Każdy użytkownik powinien mieć iluzję, że sam korzysta z bazy danych.
4. *Trwałość* - dane zatwierdzone przez transakcję powinny być dostępne nawet w sytuacji awarii oprogramowania lub sprzętu.

Mechanizmy współdzielenia zasobów bazy danych

- ▣ *blokady* – ograniczające działanie innych transakcji na zablokowanym obiekcie, oraz
- ▣ mechanizm *wielowersyjności* - umożliwiający odczytywanie danych zmienianych równocześnie przez inne transakcje w takiej postaci w jakiej istniały w chwili rozpoczynania się danej transakcji (przy czym dana transakcja widzi zmiany dokonane od tego czasu przez samą siebie).

Atomowość transakcji

- Transakcja może dokonać swojego zatwierdzenia (*commit*) po zakończeniu wszystkich swoich akcji lub dokonać wycofania (*abort*) (ewentualnie zostać wycofana przez SZBD) po wykonaniu wszystkich lub części swoich akcji.
- *Atomowość (niepodzielność)* – transakcja albo wykonuje wszystkie swoje akcje tak jakby w jednym kroku albo nie wykonuje żadnych swoich akcji.
- SZBD zapisuje wszystkie wykonywane akcje w dzienniku (*logu*) tak aby w razie potrzeby móc skasować wszystkie akcje wycofywanych transakcji.

Przykład: poprawna realizacja transakcji

T1:	BEGIN	A=A+100,	B=B-100	END
T2:	BEGIN	A=1.06*A,	B=1.06*B	END

- Intuicyjnie transakcja T1 dokonuje transferu \$100 z konta B na konto A. Transakcja T2 dopisuje do obu kont 6% odsetki.
- Poprawna realizacja obu transakcji powinna być równoważna albo szeregowemu wykonaniu T1 potem T2 albo szeregowemu wykonaniu T2 potem T1.
- Efekt wykonania nie jest (bo nie musi być) taki sam w obu przypadkach
- Efekt poprawnego wykonania kilku transakcji

Przykład - poprawna realizacja transakcji (c.d.)

- Możliwy poprawny przeplot akcji obu transakcji (plan):

T1:	$A = A + 100,$	$B = B - 100$
T2:	$A = 1.06 * A,$	$B = 1.06 * B$

- A to niepoprawny plan:

T1:	$A = A + 100,$	$B = B - 100$
T2:	$A = 1.06 * A, B = 1.06 * B$	

- Z punktu widzenia SZBD drugi plan jest postaci:

T1:	$R(A), W(A),$	$R(B), W(B)$
T2:	$R(A), W(A), R(B), W(B)$	

Plan wykonania transakcji

- Plan szeregowy: Najpierw akcje jednej transakcji, następnie akcje drugiej transakcji.
- Równoważne plany: Efekt realizacji obu planów taki sam dla każdego stanu bazy danych.
- Plan szeregowalny: Plan, który jest równoważny pewnemu planowi szeregowemu (realizacja transakcji przez SZBD ma własność **izolacji**) .
- Jeśli każda transakcja zachowuje spójność bazy danych, każdy plan szeregowalny także zachowuje spójność bazy danych (realizacja transakcji przez SZBD ma własność **spójności**) .

Anomalie przy przeplataniu akcji

- Odczyt niezatwierdzonych danych (konflikt WR):

T1:	R(A), W(A),	R(B), W(B), Abort
T2:	R(A), W(A), C	

- Niepowtarzalny odczyt (konflikt RW):

T1:	R(A),	R(A), W(A), C
T2:	R(A), W(A), C	

Anomalie przy przeplataniu akcji (c.d)

- ▣ Nadpisanie niezatwierdzonych danych (konflikt WW):

T1:	W(A),	W(B), C
T2:	W(A), W(B), C	

Podstawowe rodzaje blokad

- Współdzielona (ang. shared lock) - daje transakcji współdzielony dostęp do zasobu. Np. kilka transakcji może jednocześnie pracować na tej samej tabeli. Jeśli transakcja zakłada współdzieloną blokadę, inne transakcje też mogą założyć współdzieloną blokadę, ale nie mogą założyć wyłączonej blokady.
- Wyłączna (ang. exclusive lock) - daje transakcji wyłączny dostęp do obiektu. Tylko jedna transakcja może mieć założoną wyłączną blokadę na obiekcie i w tym czasie nie może być założonej żadnej innej blokady nawet współdzielonej.

Zarządzanie współbieżnością oparte na blokadach zakładanych na obiekty

□ Protokół ścisłego blokowania dwufazowego (Strict 2PL):

- Każda transakcja musi uzyskać **blokadę S (współdzieloną)** na obiekcie zanim odczyta ten obiekt oraz **blokadę X (wyłącznie)** na obiekcie przed zapisaniem go.
- Jeśli transakcja trzyma blokadę X na obiekcie, żadna inna transakcja nie ma prawa założyć żadnej blokady (ani S ani X) na tym obiekcie.
- Jeśli transakcja trzyma blokadę S na obiekcie, żadna inna transakcja nie ma prawa założyć blokady X na tym obiekcie.
- Gdy transakcja nie może założyć blokady na obiekcie, może ustawić się w kolejce oczekujących transakcji stowarzyszonej z tym obiektem.
- **Blokady trzymane przez transakcję są zwalniane gdy transakcja kończy się.**

□ Protokół Strict 2PL gwarantuje realizację wyłącznie planów szeregowalnych.

Protokół blokowania dwufazowego (2PL)

□ *Zamiast*

- ***Blokady trzymane przez transakcję są zwalniane gdy transakcja kończy się.***

□ *przyjmujemy:*

- ***Transakcja nie może założyć żadnej nowej blokady po zwolnieniu jakiegokolwiek blokady.***

□ *Protokół 2PL także prowadzi do planów szeregowalnych.*

Zakleszczenia (deadlocks)

- Cykl transakcji oczekujących wzajemnie na zwolnienie blokady.
- Dwa sposoby radzenia sobie z zakleszczeniami:
 - zapobieganie,
 - wykrywanie.

Zapobieganie zakleszczeniom

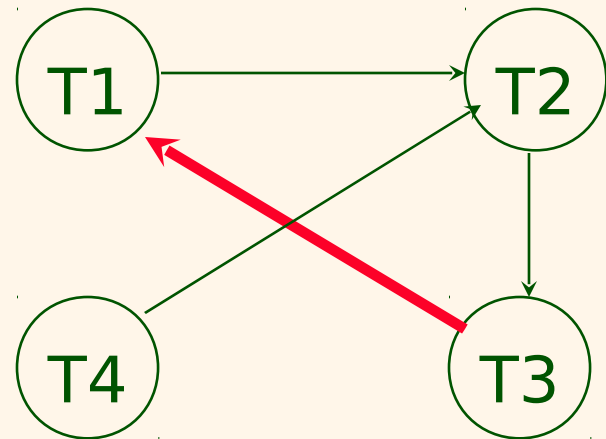
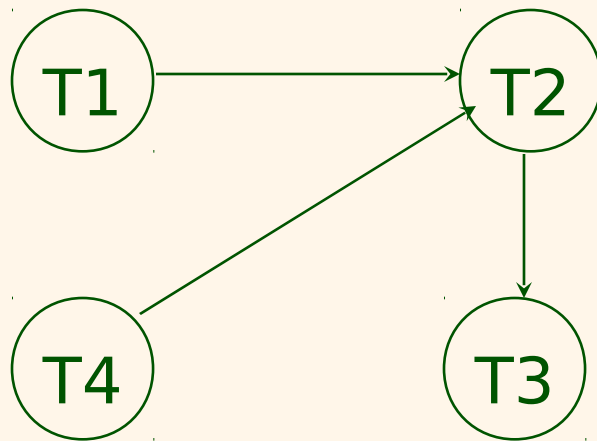
- Przypisz priorytety na podstawie znaczników czasowych. Przypuśćmy, że transakcja T_i chce założyć blokadę, którą utrzymuje transakcja T_j . Dwie strategie:
 - “Wait-Die”: Jeśli T_i ma wyższy priorytet, T_i czeka na T_j ; wpp. T_i zostaje wycofana.
 - “Wound-wait”: Jeśli T_i ma wyższy priorytet, T_j zostaje wycofana; wpp. T_i czeka.
- Przy restartowaniu transakcji, ma ona swój początkowy znacznik czasowy zapewniający jej wyższy priorytet od transakcji, które rozpoczęły się później od niej. Gwarantuje to wykonanie tej transakcji prędzej lub później (*“nie zagłodzenie jej”*) .

Wykrywanie zakleszczeń

- Utwórz graf oczekiwania na blokadę:
 - Węzłami są transakcje.
 - Istnieje krawędź od T_i do T_j jeśli T_i oczekuje na zwolnienie blokady przez T_j .
- Co jakiś czas sprawdzaj czy jest cykl.

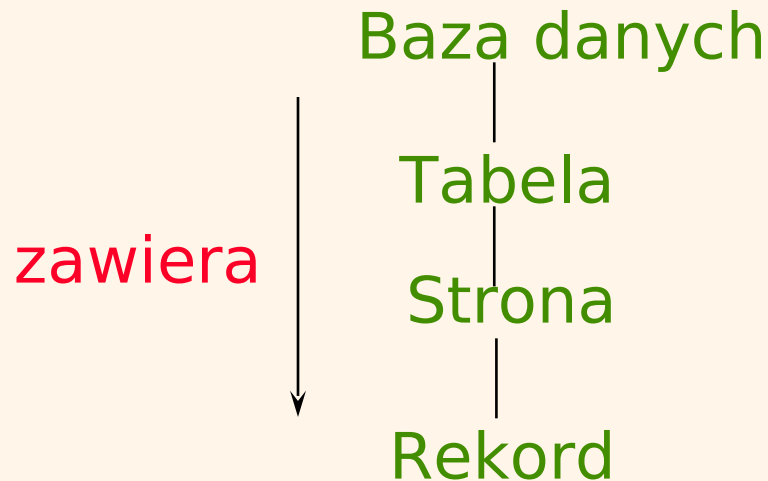
Przykład

T1: S(A), R(A), S(B)
T2: X(B), W(B) X(C)
T3: S(C), R(C) X(A)
T4: X(B)



Blokady wielo-poziomowe

- Obiekty bazodanowe są zagnieżdżone. Blokada na pod-obiekcie implikuje pewną blokadę na nad-obiekcie (i na odwrót).



Nowe typy blokad

□ Blokady intencyjne

- Przed zablokowaniem obiektu transakcja musi założyć blokadę intencyjną na wszystkich “przodkach” danego obiektu.
- Przy odblokowywaniu idziemy z dołu w górę.
- Dodatkowo typ **SIX**: S & IX.

	--	IS	IX	S	X
--	✓	✓	✓	✓	✓
IS	✓	✓	✓	✓	
IX	✓	✓	✓		
S	✓	✓		✓	
X	✓				

Przykłady

- T1 przebiega R i aktualizuje kilka rekordów:
 - T1 uzyskuje blokadę SIX na R, następnie kolejno uzyskuje blokadę S na rekordach w R i czasami podwyższa blokadę rekordu na X.
- T2 używa indeksu do odczytania części tabeli R:
 - T2 uzyskuje blokadę IS na R, a następnie kolejno uzyskuje blokadę S na rekordach w R.
- T3 odczytuje całą tabelę R:
 - T3 uzyskuje blokadę S na R.

		IS	IX	S	X
	✓	✓	✓	✓	✓
IS	✓	✓	✓	✓	
IX	✓	✓	✓		
S	✓	✓		✓	
X	✓				

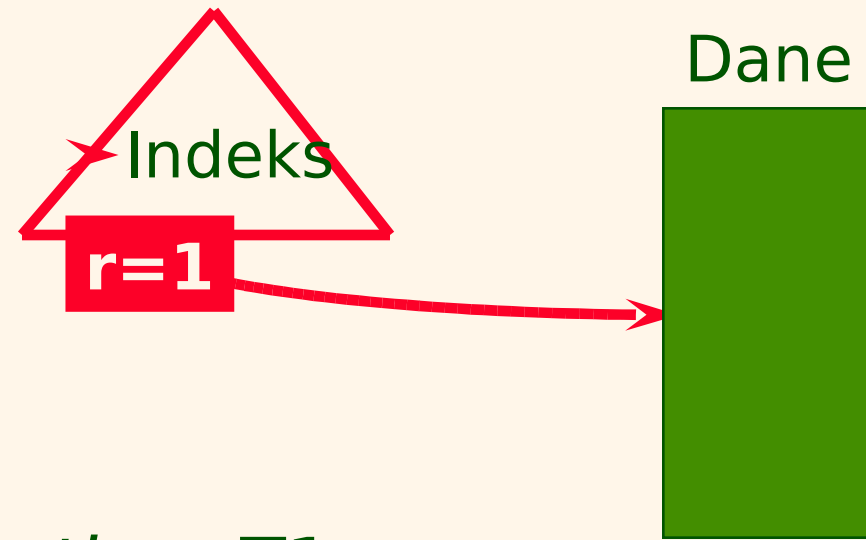
Problem fantomów

- Protokół Strict 2PL (w dotychczasowej postaci) jest poprawny pod warunkiem, że baza danych jest ustaloną, nie zmieniającą się kolekcją obiektów:
 - T1 blokuje wszystkie strony zawierające rekordy żeglarzy z *rating* = 1 i wyznacza najstarszego żeglarza (*age* = 71).
 - Następnie T2 wstawia nowego żeglarza: *rating* = 1, *age* = 96.
 - T2 usuwa najstarszego żeglarza z *rating* = 2 (powiedzmy *age* = 80) i zatwierdza.
 - T1 blokuje wszystkie strony zawierające rekordy żeglarzy z *rating* = 2 i wyznacza najstarszego (powiedzmy *age* = 63).
- Wykonywania tych transakcji nie da się uszeregować!

Problem

- T1 zakłada zablokowanie wszystkich rekordów żeglarzy z *rating* = 1!
- Potrzebne są blokady na zbiory rekordów określone przez predykaty np. *rating*=1. Można to uzyskać przez zablokowanie węzła indeksu z *rating*=1 jeśli taki indeks istnieje.

Rozwiązanie



- Jeśli jest indeks na polu *rating*, T1 blokuje stronę indeksu zawierającą pozycje danych z *rating* = 1.
- Jeśli nie ma indeksu na polu *rating*, T1 musi zablokować cały plik/tabele.

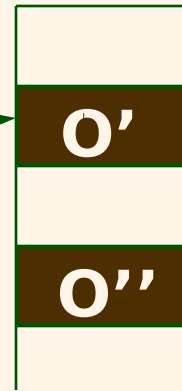
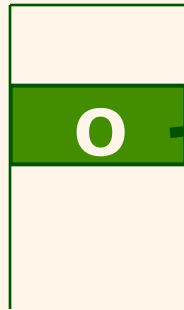
Optymistyczne blokowanie

- **Faza 1**: Transakcja wczytuje potrzebne dane do swoich lokalnych buforów i na nich dokonuje zmian bez zakładania żadnych blokad.
- **Faza 2**: Transakcja sprawdza czy dokonane przez nią odczyty i zapisy nie pozostają w konflikcie z odczytami i zapisami zatwierdzonych już transakcji. Jeśli nie, następuje przepisanie zmian z lokalnych buforów do globalnych i zatwierdzenie transakcji. Jeśli tak, następuje restartowanie jeszcze raz tej samej transakcji.
- Tylko w czasie realizacji **Fazy 2** jest konieczność założenia blokad X na zmieniane obiekty.

Wieloversyjność

- **Idea:** Procesy zapisujące tworzą nową kopię obiektu podczas gdy procesy odczytujące korzystają ciągle ze starej wersji:

Główny segment
(Aktualne
wersje
obiektów)



Pula wersji

(Starsze wersje obiektów
używane przez procesy
odczytujące.)

- Procesy odczytujące mogą działać bez zakładania blokad.

Zjawiska związane ze współbieżnymi transakcjami

- Odczyt niezatwierdzonych danych (ang. *dirty read*) - transakcja odczytuje dane, które zmieniła druga transakcja ale ich nie zatwierdziła.
- Niepowtarzalny odczyt - w ramach tej samej transakcji, widzieć zmiany wprowadzane przez zatwierdzone transakcje.
- Fantom - wiersz, którego nie było w tabeli na początku wykonywania transakcji, a który został wprowadzony przez zatwierdzoną transakcję w trakcie wykonywania transakcji.

Transakcje w SQL-92

Standard ANSI/ISO definiuje *poziomy izolacji*: czy transakcje widzą zmiany dokonywane przez inne współbieżnie działające transakcje.

Poziom izolacji	Niezatw-ierdzo ny odczyt	Niepowtarzalny odczyt	Fantomy
Read Uncommitted	TAK	TAK	TAK
Read Committed	NIE	TAK	TAK
Repeatable Reads	NIE	NIE	TAK
Serializable	NIE	NIE	NIE

Ustawianie poziomu izolacji w SQL

```
□ SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;  
    .....  
    COMMIT;
```

```
□ SET TRANSACTION ISOLATION LEVEL READ COMMITTED;  
    .....  
    COMMIT;
```

Blokady w Oracle

- System Oracle automatycznie zakłada blokadę na wiersz, który ma być zmieniany. Blokada zostaje zdjęta w chwili wykonywania COMMIT lub ROLLBACK.
- Gdy transakcja dochodzi do zablokowanego wiersza, może albo czekać na zwolnienie blokady albo dokonać ROLLBACK (jeśli transakcja może się obyć bez zmieniania tego wiersza, może też zrobić coś innego).
- Jeśli użytkownik chce mieć pewność, że wszystkie obiekty będą dostępne w trakcie wykonywania jego transakcji musi sam dokonać blokady wszystkich potrzebnych mu obiektów (tabel lub konkretnych wierszy).

Przykład założenia wyłączonej blokady na wybrane wiersze

```
□  SELECT * FROM Klienci
    WHERE Kraj = 'Polska'
    FOR UPDATE
    -- założenie blokady wyłączonej na klientów z
    Polski i                                -- współdzielonej na
    tabelę Klienci
    NOWAIT;
-- gdy nie można założyć blokady, nie czekaj
```

Przykład założenia wyłączonej blokady na tabelę

```
LOCK TABLE Klienci  
    IN EXCLUSIVE MODE  
    -- zablokowanie całej tabeli w trybie  
    wyłącznym  
NOWAIT;
```


Blokady w Oracle (c.d)

- Przy wykonywaniu instrukcji DDL (**CREATE/ALTER/DROP**) też są zakładane blokady:
 - dla obiektu bezpośrednio związanego z operacją - blokada wyłączna;
 - dla obiektu pośrednio związanego z operacją - np. przy **CREATE PROCEDURE** - tabele w niej występujące - blokada współdzielona;
- Oracle w trybie READ COMMITTED nie zakłada blokad współdzielonych przy wykonywaniu zapytań używa za to *wielowersyjności* – podobnie jak przy wykonywaniu transakcji tylko-odczyt:

SET TRANSACTION READ ONLY;

Dziennik wycofań

- ▣ W celu umożliwienia wycofania transakcji SZBD zapisuje wszystkie wykonywane przez nią zmiany w specjalnym *dzienniku wycofań* (ang. *undo log*) nazywanym w Oracle *segmentami wycofań*. Gdy trzeba wycofać transakcję system odczytuje w tył zapisy o zmianach wprowadzonych przez transakcję i przywraca poprzednie wartości danych.

Dziennik wycofań i wielowersyjność

- Efekt wykonania zapytania powinien być spójny i odpowiadać chwili rozpoczęcia jego wykonywania. Już w chwili zakończenia wykonywania instrukcji stan bazy danych może być inny (jeśli nie stosujemy blokad współdzielonych przy wykonywaniu zapytania).
- SCN - *systemowy numer zmiany*. Każda zatwierdzona transakcja zwiększa ten licznik o jeden. SCN może być uważany za identyfikator zatwierdzanej transakcji. Na każdej stronie jest zapisany SCN ostatniej transakcji, która ją zmieniła.

Algorytm wykonywania zapytania

- ▣ Niech q_SCN będzie aktualnym SCN w chwili rozpoczęcia wykonywania zapytania. W trakcie wykonywania zapytania są odczytywane strony danych. Dla każdej takiej strony z nagłówka jest odczytywany zapisany w nim s_SCN (numer transakcji, która ją ostatnio zmieniła).
 - Jeśli $s_SCN \leq q_SCN$, wtedy można zawartość strony użyć w obliczeniach.
 - Jeśli $s_SCN > q_SCN$, wtedy w oparciu o *segmenty wycofań* należy obliczyć zawartość strony w chwili q_SCN i użyć tę zawartość do wykonania zapytania.
- ▣ W podobny sposób są wykonywane transakcje raportujące typu READ ONLY.

Dziennik powtórzeń i odtwarzanie

- Dziennik rejestrujący wszystkie zmiany zachodzące w bazie danych - nazywany *dziennikiem powtórzeń* (ang. *redo log*). Jest on z założenia trzymany na innym nośniku danych niż pliki z danymi w bazie danych. Na ogół dokonuje się stale jego archiwizacji przepisując go na taśmę.
- Zmiana danych na stronie i informacja o zatwierdzeniu są najpierw zapisywane do *dziennika powtórzeń*, dopiero potem uwzględniane w pliku danych na dysku (zasada WAL – *write-ahead logging*). Po zapisie do dziennika powtórzeń nawet awaria serwera lub dysku z danymi nie spowoduje utraty danych bo można je odtworzyć (własność **trwałości**) .

Dziennik powtórzeń i odtwarzanie

- Gdy nastąpi awaria dysku, pozycje dziennika powtórzeń (z części on-line na dysku lub części archiwizacyjnej na taśmie) zastosowane do kopii zabezpieczającej pozwalają odtworzyć stan bazy danych w chwili awarii. Jest to proces nazywany *odtworzeniem do przodu*.
- W przypadku awarii serwera bazy danych analiza samego dziennika powtórzeń pozwala na odtworzenie stanu bazy danych w chwili awarii.

Dziennik powtórzeń i odtwarzanie

- Ponieważ w dzienniku zapisane są również pozycje segmentów wycofań, jest możliwość wycofania nie zatwierdzonych transakcji, których działanie zostało przerwane w chwili awarii
 - na przykład przy transferze pieniędzy - z jednego konta pieniądze zostają zdjęte, w tej chwili następuje awaria, na drugie konto pieniądze już nie zostają przelane.
- Jest to proces nazywany *odtworzeniem do tyłu*. W rezultacie tych dwóch odtwarzań jest możliwość doprowadzenia stanu bazy danych do spójnego stanu.

Rezerwowa baza danych (*standby*)

- Dodatkowa instalacja bazy danych na osobnym komputerze utrzymywana stale w specjalnym trybie *standby* - z ciągle dokonywanym odtwarzaniem w oparciu o kopie zarchiwizowanego dziennika powtórzeń generowane przez główną, operacyjną bazę danych.
- W przypadku awarii dysku lub katastrofy w rodzaju trzęsienia ziemi, pożaru czy kradzieży, rezerwowa baza danych przechodzi z trybu *standby* w tryb *read write* i przejmuje obowiązki głównej bazy danych.
- Rezerwowa baza danych zwykle znajduje się w fizycznie oddalonym węźle, do którego stale są przesyłane kolejne części zarchiwizowanego dziennika powtórzeń.

Rezerwowa baza danych c.d.

- Rezerwowej bazy danych można używać do raportowania – przechodząc w tryb READ ONLY – napływające pozycje zarchiwizowanego dziennika powtórzeń utrzymywane są w kolejce, dopóki nie wrócimy do trybu STANDBY. Po przejściu bazy danych w tryb READ WRITE nie jest już możliwy jej powrót do trybu STANDBY.
- Zamiast rezerwowej bazy danych alternatywę stanowi użycie replikacji bazy danych.