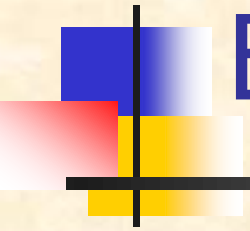


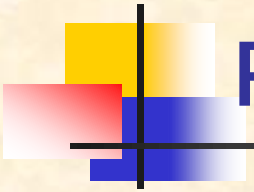
Bazy Danych

dr inż. Mariusz Kopeć

pok. 103, D-4

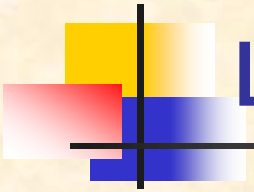
tel. 888 5182

makopec@agh.edu.pl



Program wykładu

1. Pojęcia podstawowe
2. Systemy Zarządzania Bazami Danych
3. Technologie baz danych
4. Modele logiczne danych
5. Relacyjny model danych
6. Model związków encji
7. Asocjacje
8. Hierarchie encji
9. Transformacje do modelu relacyjnego



Literatura

- R. Elmasri, S. B. Navathe, Wprowadzenie do systemów baz danych, Helion, 2005
- M. J. Hernandez, Bazy danych dla zwykłych śmiertelników, Mikom, 2004
- P. Beynon-Davies, Systemy baz danych, WNT, 2003
- C. J. Date, Wprowadzenie do systemów baz danych, WNT, Warszawa, 2000
- http://dev.mysql.com/doc/refman/5.6/en/

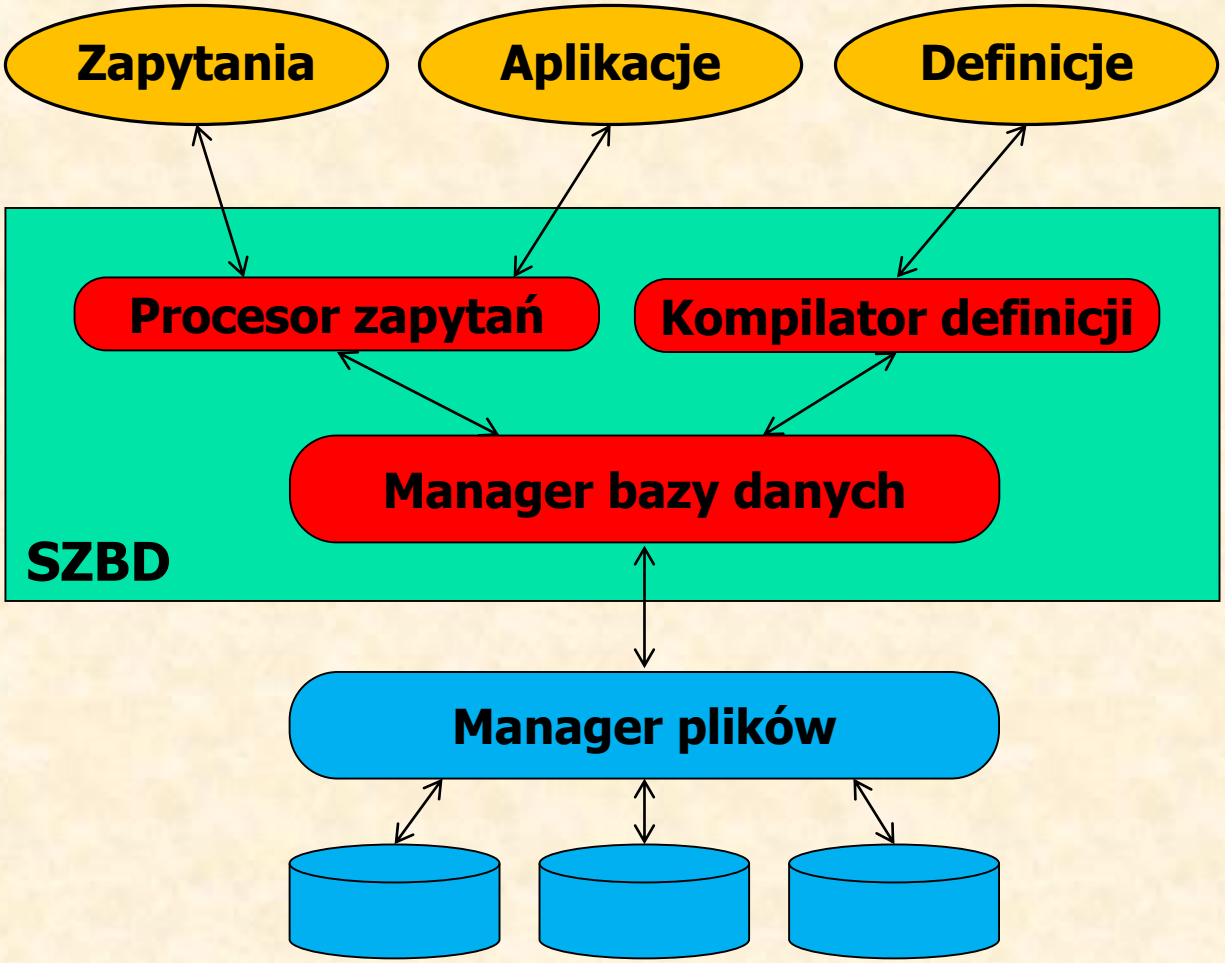
Pojęcia podstawowe

- **Dane** są symbolami lub zbiorami symboli odzwierciedlających pewne fakty. Same w sobie nie mają znaczenia bowiem konieczna jest jeszcze ich odpowiednia interpretacja:
  1983 samo w sobie nic nie znaczy, ale interpretowane jako rok wydania książki zawiera pewną informację.
- **Baza danych** jest kolekcją danych będącą modelem pewnego aspektu rzeczywistości określanej jako **obszar analizy**.
- Rzeczy istotne z punktu widzenia obszaru analizy nazywamy **klasami** lub **encjami** – przykład: *Klient*, *Zamówienie*
- Encje posiadają cechy nazywane **właściwościami** lub **atrybutami** – przykład: *Klient* posiada *Nazwisko*
- Pomiędzy encjami mogą występować określone związki nazywane **relacjami** – przykład: *Klient* złożył *Zamówienie*

Systemy Zarządzania Bazą Danych

- **System Zarządzania Bazą Danych (DBMS)** – zbiór gotowych narzędzi zapewniających odpowiedni dostęp , manipulację, modyfikację i aktualizację danych gromadzonych w systemie komputerowym (bazie danych)
- **Elementy składowe SZBD:**
  - manager bazy danych – zarządzanie obiektami bazy danych
  - procesor zapytań – przetwarzanie poleceń kierowanych do BD
  - kompilator definicji schematu – przetwarzanie definicji obiektów znajdujących się w bazie na postać zrozumiałą dla managera bazy
- **SZBD komunikuje się z:**
  - managerem plików (fizyczną bazą danych)
  - zapytaniami użytkownika
  - aplikacjami użytkownika
  - narzędziami definicji schematu bazy

Systemy Zarządzania Bazą Danych

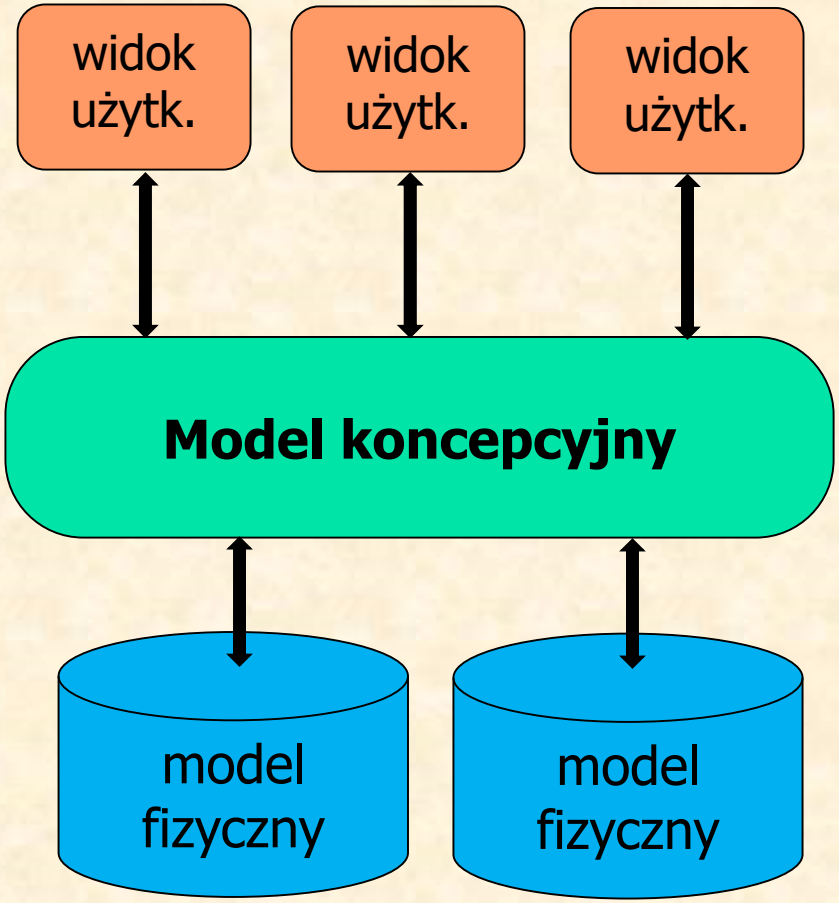


Architerktura ANSI-SPARC SZBD

Architektura trójwarstwowa zaproponowana w 1975 roku przez *American National Standards Institute* oraz *Standards Planning And Requirements Committee* jako abstrakcyjny standard projektowania SZBD. Zakłada istnienie następujących 3 poziomów:

- **Poziom zewnętrzny (użytkownika)** – opisuje jak użytkownicy widzą dane. Realizacji na poziomie zewnętrznym, czyli sposobów w jaki użytkownicy widza dane może być wiele
- **Poziom koncepcyjny (pojęciowy)** – opisuje logiczny widok wszystkich danych w bazie bez szczegółów dotyczących praktycznej realizacji
- **Poziom wewnętrzny (fizyczny)** – opisuje fizyczny sposób przechowywania danych oraz dostępu do nich

Architektura ANSI-SPARC SZBD



Poziom zewnętrzny

Poziom pojęciowy

Poziom wewnętrzny

Przesłanki architektury 3-poziomowej

- **niezależność aplikacji i danych**
  - różne widoki użytkowników
  - dopasowanie do szczegółowych potrzeb użytkowników
  - zmiany jednego widoku nie wpływają na inne
- **możliwość zmiany koncepcji przechowywania danych**
  - na tym poziomie działa administrator bazy danych
  - wprowadzane przez administratora zmiany struktury bazy danych nie wpływają na widoki użytkowników
  - ***logiczna niezależność danych***
- **ukrycie fizycznej implementacji**
  - użytkownicy nie znają i nie muszą znać szczegółów przechowywania danych
  - zmiany w warstwie fizycznej nie wpływają na działanie bazy danych widzianej ze strony użytkowników - na przykład zmiana dysku czy stosowanego systemu plików jest dla użytkowników niewidoczna
  - ***fizyczna niezależność danych***

Wymagania odnośnie SZBD

System Zarządzania Bazą Danych umożliwia:

- opisywanie encji w terminach składowanych danych
- tworzenie oraz trwałe i wydajne składowanie dużych i bardzo dużych kolekcji danych (nawet 10^{15} bajtów)
- wydajne przeszukiwanie i aktualizowanie danych
- wydajne modyfikowanie danych
- zapewnienie współbieżnego dostępu do danych
- odtwarzanie aktualnego i spójnego stanu bazy danych po awariach
- zapewnienie spójności przechowywanych danych
- kontrolę dostępu do danych
- obsługę różnych interfejsów współpracujących z bazą danych
- działanie w długim cyklu życia

Najpopularniejsze SZBD (2019)

System	Model BD	Licencja	Firma	Start	Rank
Oracle	relacyjny	komercyjna	Oracle	1980	1341
MySQL	relacyjny	open source	Oracle	1995	1260
MS SQL Server	relacyjny	komercyjna	Microsoft	1989	1098
PostgreSQL	relacyjny	open source	PostgreSQL	1989	514
MongoDB	nierelacyjny	open source	MongoDB Inc	2009	438
DB2	relacyjny	komercyjna	IBM	1983	163
Elasticsearch	nierelacyjny	open source	Elastic	2010	149
Redis	nierelacyjny	open source	Sanfilippo S.	2009	148
MS Access	relacyjny	komercyjna	Microsoft	1992	125
SQLite	relacyjny	open source	Dwayne Rich.	2000	122
Cassandra	nierelacyjny	open source	Apache Soft.	2008	121

Technologie baz danych

- **Modele danych**
  - modele pojęciowe – związki encji (ERM), zunifikowany język modelowania (UML), język definicji obiektów (ODL)
  - modele logiczne – relacyjne, obiektowe, post-relacyjne
- **Fizyczne struktury danych i metody dostępu**
  - pliki uporządkowane, indeksy, bitmapy
  - operacje połączenia, sortowania, grupowania, połowienie binarne, haszowanie
  - regułowe i kosztowe metody optymalizacji zapytań
- **Przetwarzanie transakcyjne**
  - dostęp do danych przez transakcje
  - metody synchronizacji transakcji (znaczniki czasowe)
  - metody odtwarzania spójności (punkty kontrolne, logi)
  - metody archiwizacji

Języki baz danych

- **Język definiowania danych DDL**
  - umożliwia tworzenie, modyfikowanie i usuwanie struktur danych oraz całych baz danych
- **Język manipulowania danymi DML**
  - umożliwia realizację żądań użytkownika obejmujących wyszukiwanie, wstawianie, modyfikowanie i usuwanie danych
- **Język kontroli danych DCL**
  - zapewnia autoryzację dostępu do danych
- **Język integralności danych DIL**
  - służy do określenia więzów integralności danych

czasami wyróżnia się dodatkowo

- **Język definicji widoku VDL**
  - umożliwia definicję widoku zewnętrznego (rzadko implementowany)

Użytkownicy baz danych

- **Użytkownicy zwykli z prawem odczytu** – mają prawo do wyszukiwania i odczytu informacji
- **Użytkownicy zwykli z prawem odczytu i modyfikacji danych** – mają prawo do wyszukiwania i odczytu informacji oraz dodawania, zmieniania i usuwania danych
- **Administratorzy baz danych** – zarządzają dostępem do baz i nadają uprawnienia użytkownikom; zarządzają logiczną i fizyczną konfiguracją systemu
- **Projektanci baz danych** – projektują schematy baz danych, definiują perspektywy użytkowników, określają strukturę aplikacji i interfejsy, utrzymują system
- **Analitycy baz danych** – wyznaczają charakterystyki danych, wykrywają zależności i cechy jakościowe
- **Inżynierowie wiedzy** – wydobywają wiedzę z danych (data mining)

Modele logiczne danych

- **Modele logiczne danych** – zasady posługiwania się danymi obejmujące:
  - definicje danych – zbiory reguł określających strukturę danych
  - operowanie danymi – zbiory reguł dotyczących dostępu do danych oraz ich modyfikacji
  - integralność danych – zbiory reguł określających które stany danych są poprawne, czyli jakie operacje prowadzące do zmiany danych są dozwolone
- **Modelowanie logiczne**
  - jest realizowane przez projektantów na podstawie specyfikacji wymagań i modelu pojęciowego
  - jest ściśle powiązane z określonym modelem bazy danych, a często z jej konkretną implementacją
  - określa struktury modelu danych, ale nie ingeruje w struktury fizyczne

Modele logiczne danych

- **Chronologiczne zestawienie modeli danych**
  - hierarchiczny – 1965
    - dane są przechowywane w postaci rekordów pomiędzy którymi występuje zależność nadrzędności-podrzędności
    - każdy rekord (z wyjątkiem pierwszego) posiada dokładnie jeden rekord nadrzędny
    - jeżeli rekordowi można by przypisać więcej rekordów nadrzędnych, musi zostać do nich skopiowany
    - usunięcie rekordu powoduje usunięcie wszystkich jego rekordów podrzędnych
    - *implementacja: system plików*
  - sieciowy (grafowy) – 1968
    - zmodyfikowana wersja modelu hierarchicznego
    - rekordy zawierają pola przechowujące dane
    - rekordy mogą mieć wiele rekordów nadrzędnych i podrzędnych
    - implementacja: Integrated Database Management System - 1983
  - **relacyjny** – 1970
    - dane przechowywane są w tabelach w rekordach (wierszach)
    - każda tabela ma określoną stałą kolumn i dowolna liczbę wierszy
    - każda tabela ma zdefiniowany **klucz danych** jednoznacznie identyfikujący wiersz

Modele logiczne danych cd

- **Chronologiczne zestawienie modeli danych cd**
  - semantyczny – 1978
    - dane są zorganizowane w oparciu o obiekty oraz relacje zachodzące pomiędzy tymi obiektami a rzeczywistym światem
    - obiekty są interpretowane z uwzględnieniem relacji, które pomiędzy nimi zachodzą
    - semantyczne modele danych budowane w celu maksymalnie dokładnego reprezentowania świata rzeczywistego w oparciu o istniejące dane
  - obiektowy – 1980
    - dane zapisywane są w formie obiektów
    - dane wykorzystują założenia modelu obiektowego takie jak: klasy, dziedziczenie, metody, przeciążanie metod, referencje do obiektów
    - wykorzystanie paradygmatu obiektowego zapewnia automatycznie realizację wymagań stawianych SZBD
  - dedukcyjny – 1982
    - dane zgromadzone tak jak w modelu relacyjnym lub obiektowym uzupełnione są tzw. bazą reguł wnioskowania, zawierającą reguły wnioskowania i więzy integralności
    - dane znajdujące się w bazie dostępne są użytkownikowi bezpośrednio
    - dane nieznajdujące się w bazie są tworzone na bieżąco w procesie wnioskowania
  - postrelacyjny – 1990
    - model relacyjny rozszerzony o elementy obiektowości

Relacyjny model danych

- Model oparty na teorii zbiorów i teorii predykatów, opublikowany w 1970 przez E.F. Codd'a
- Zakłada istnienie tylko jednej struktury danych nazywanej **relacją** lub **tabelą**
- Tabele reprezentują różne realne „rzeczy" ze świata rzeczywistego, będące rzeczywistymi obiektami materialnymi (np. *Osoba*) lub niematerialnymi (np. *Rozmowa, Przedmiot, Wiedza*)
- Każda tabela ma unikalną nazwę i reprezentuje tylko jedną „rzecz"
- Każda „rzecz" posiada pewien zestaw atrybutów, taki sam dla wszystkich „rzeczy" w tabeli
- Zbiór nazw atrybutów nazywa się **schematem relacji**
- Każdy wiersz tabeli reprezentujący wystąpienie obiektu rzeczywistego jest **krotką** *(typ struktury danych)*
- Tabele znajdujące się w bazie mogą być powiązane między sobą poprzez **związki**

Relacyjny model danych

- **Relacja jest tabelą spełniającą następujące warunki**
  - Każda relacja w bazie ma jednoznaczną nazwę
  - Każda kolumna w ramach relacji ma jednoznaczną nazwę
  - Wszystkie wartości w kolumnie są tego samego rodzaju (mają tę samą dziedzinę)
  - Porządek kolumn w relacji nie jest istotny
  - Wiersze w tabeli nie mogą się powtarzać (są różne)
  - Kolejność wierszy nie jest istotna
  - Każde pole relacji musi mieć wartości elementarne (atomowe)

Nr_ind	Imię	Nazwisko	Kierunek	Rok
1010	Jan	Kowalski	energ	2
1013	Anna	Nowak	energ	2
1001	Adam	Zawadzki	techn	3
1023	Anna	Nowak	energ	2

Relacyjny model danych

Postulaty Codd'a

0 – Każdy system uznany za relacyjny musi mieć możliwość zarządzania danymi tylko za pomocą mechanizmów relacyjnych

1 – Postulat informacji: dane w bazie muszą być reprezentowane wyłącznie na poziomie logicznym jako wartości w tabelach

2 – Postulat dostępu: każda tabela musi mieć zdefiniowany klucz główny, będący identyfikatorem wiersza w tabeli. Dostęp do każdej wartości atomowej musi być możliwy przez nazwę tablicy, nazwę kolumny i nazwę klucza głównego

3 – Postulat wartości NULL: dostępna jest specjalna wartość NULL reprezentująca wartości puste lub nieadekwatne, inna od wszystkich i podlegająca przetwarzaniu

4 – Postulat metadanych: baza musi wykorzystywać katalog systemowy, w którym w tablicach są przechowywane metadane (dane o danych)

5 – Postulat uniwersalnego języka: musi istnieć jeden język umożliwiający manipulowanie danymi w bazie, dostępny w trybie interaktywnym i z aplikacji

6 – Postulat modyfikowalności perspektyw: muszą być możliwe modyfikacje danych wynikowych udostępnianych przez perspektywy

Relacyjny model danych

Postulaty Codd'a cd

7 – Postulat modyfikowalności danych: system musi umożliwiać operacje modyfikacji danych takie jak INSERT, UPDATE, DELETE

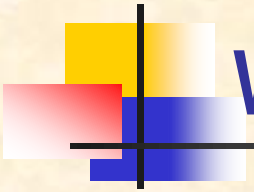
8 – Postulat fizycznej niezależności danych: zmiany fizycznej reprezentacji danych oraz organizacji dostępu do nich nie wpływają na aplikacje

9 – Postulat logicznej niezależności danych: zmiany wartości w tabelach nie wpływają na aplikacje

10 – Postulat niezależności więzów spójności: więzy spójności są definiowane w bazie i nie zależą od aplikacji

11 – Postulat niezależności dystrybucyjnej: działanie aplikacji nie zależy od modyfikacji i dystrybucji bazy

12 – Postulat bezpieczeństwa względem operacji niskiego poziomu: operacje niskiego poziomu nie mogą naruszać modelu relacyjnego i więzów spójności



Wartość NULL

- Reprezentuje brak wartości danego atrybutu ze względu na:
  - brak wiedzy o jego wartości (np. numer PESEL jakiejś osoby)
  - brak wiedzy o jego stosowalności (np. dana osoba może mieć numer prawa jazdy lub nie)
  - jego niestosowalność (np. objętość figury płaskiej)
- Jest różny od zera, spacji lub stringu pustego
- Podlega wyświetlaniu
- Nie podlega sortowaniu
- Nie stosuje się wobec niego operatorów porównania
- Implikuje stosowanie logiki trójwartościowej

Logika trójwartościowa

AND	true	false	null
true	true	false	null
false	false	false	false
null	null	false	null

OR	true	false	null
true	true	true	true
false	true	false	null
null	true	null	null

NOT	true	false	null
	false	true	null

Relacyjny model danych - klucze

- **Superklucz tabeli** – kolumna lub zestaw kolumn tabeli jednoznacznie identyfikujący rekord w tabeli, czyli rekord musi mieć unikalną wartość superklucza. Zwykle jest nadmiarowy
- **Klucz tabeli** – minimalny superklucz. Może ich być wiele i wtedy nazywane są **kluczami kandydującymi**
- **Klucz podstawowy (PK)** – kluczy wybrany spośród kluczy kandydujących. Najczęściej dodatkowo zapewnia się jego unikalność
- **Klucz obcy (FK)** – kolumna lub zestaw kolumn wskazujących na klucz podstawowy innej tabeli
- **Reguły integralności** – gwarantują, że wszystkie wprowadzane dane będą spełniać nałożone warunki i obejmują:
  - Klucze
  - Zawężenie dziedziny
  - Unikatowość wartości
  - Dopuszczenie lub blokowanie wartości pustych

Relacyjny model danych

- **Klucz podstawowy i klucz obcy**

Nr_ind (PK)	Imię	Nazwisko	Id_kier (FK)	Rok
1010	Jan	Kowalski	1	2
1013	Anna	Nowak	1	2
1001	Adam	Zawadzki	2	3
1023	Anna	Nowak	1	2

Id_kier (PK)	Kierunek
1	energ
2	techn

Algebra relacyjna

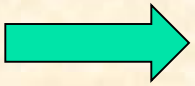
- Służy do wyszukiwania danych w relacji
- Jest proceduralnym językiem zapytań
- Jest zbiorem 8 operatorów:
  - rzutowanie (projekcja) π
  - selekcja (wybór) σ
  - produkt (iloczyn kartezjański) $\times$
  - złączenie $\bowtie$
  - suma $\cup$
  - przecięcie $\cap$
  - różnica $\setminus$
  - iloraz $\div$

Algebra relacyjna

- **Projekcja** jest operatorem, który bierze jedną relację i zwraca inną relację, w której dla wszystkich krotek (wierszy) wypisuje tylko żądane atrybuty (kolumny)

$$\pi_{Nazwisko,Rok}(Student)$$

Student		
Imię	*Nazwisko*	*Rok*
Adam	Kowalski	2
Ewa	Nowak	3
Jan	Kucharski	3



π(Student)	
Nazwisko	*Rok*
Kowalski	2
Nowak	3
Kucharski	3

Algebra relacyjna

- **Selekcja** jest operatorem, który bierze jedną relację i zwraca inną relację, w której umieszcza tylko krotki spełniające zadane warunki

$$\sigma_{\text{Rok}=3}(\text{Student})$$

Student		
Imię	*Nazwisko*	*Rok*
Adam	Kowalski	2
Ewa	Nowak	3
Jan	Kucharski	3

σ(Student)		
Imię	*Nazwisko*	*Rok*
Ewa	Nowak	3
Jan	Kucharski	3

Algebra relacyjna

- **Iloczyn kartezjański** zwraca relację, której schemat jest sumą schematów obu relacji, a krotki są wszystkimi kombinacjami krotek wejściowych

$$(\text{Student}) \times (\text{Przedmiot})$$

Student		
Imię	*Nazwisko*	*Rok*
Adam	Kowalski	2
Ewa	Nowak	3
Jan	Kucharski	3

×

Przedmiot	
ID	*Nazwa*
10	Bazy danych
11	Statystyka

Algebra relacyjna

(Student)×(Przedmiot)				
Imię	*Nazwisko*	*Rok*	*ID*	*Nazwa*
Adam	Kowalski	2	10	Bazy danych
Ewa	Nowak	3	10	Bazy danych
Jan	Kucharski	3	10	Bazy danych
Adam	Kowalski	2	11	Statystyka
Ewa	Nowak	3	11	Statystyka
Jan	Kucharski	3	11	Statystyka

Algebra relacyjna

- **Iloczyn kartezjański** może być wykonany również na relacjach posiadających takie same atrybuty. W tej sytuacji atrybuty wynikowe wskazują relację, z której pochodzą

$$(S) \times (W)$$

S		
Imię	*Nazwisko*	*Rok*
Adam	Kowalski	2
Ewa	Nowak	3
Jan	Kucharski	3

×

W	
Imię	*Nazwisko*
Jerzy	Adamski
Anna	Mroczek

Algebra relacyjna

(S)×(W)				
S.Imię	*S.Nazwisko*	*S.Rok*	*W.Imię*	*W.Nazwisko*
Adam	Kowalski	2	Jerzy	Adamski
Ewa	Nowak	3	Jerzy	Adamski
Jan	Kucharski	3	Jerzy	Adamski
Adam	Kowalski	2	Anna	Mroczek
Ewa	Nowak	3	Anna	Mroczek
Jan	Kucharski	3	Anna	Mroczek

Algebra relacyjna

- **Złączenie** jest operacją składającą się z dwóch kroków, z których pierwszym jest wykonanie iloczynu kartezjańskiego, a drugim wykonanie odpowiedniej selekcji
  - Złączenie naturalne $\bowtie$
  - Złącznie wewnętrzne (równozłączenie) $\bowtie_{=}$
  - Złącznie wewnętrzne θ warunkowe $\bowtie_{\theta}$
    - semizłączenie θ: <, >, >=, <=
    - antyzłączenie θ: <> (!=)
  - Złącznie zewnętrzne lewostronne $⟕_{L}$
  - Złącznie zewnętrzne prawostronne $⟖_{R}$
  - Złącznie zewnętrzne dwustronne $⟗$
  - Samozłączenie

Algebra relacyjna

- **Złączenie naturalne** – z wyniku iloczynu kartezjańskiego wybierane są tylko te krotki, których wspólne atrybuty mają takie same wartości. W relacji wynikowej takie atrybuty wypisywane są tylko raz. Warunek złączenia nie jest określany, bo dotyczy wszystkich wspólnych kolumn.

$$(\text{Wykładowca}) \bowtie (\text{Przedmiot})$$

Wykładowca		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Ewa	Nowak	3

$\bowtie$

Przedmiot	
ID	*Przedmiot*
2	Bazy danych
3	Statystyka
4	Fizyka

Algebra relacyjna

(Wykładowca)×(Przedmiot)				
Imię	*Nazwisko*	*ID*	*ID*	*Przedmiot*
Adam	Kowalski	2	2	Bazy danych
Ewa	Nowak	3	2	Bazy danych
Adam	Kowalski	2	3	Statystyka
Ewa	Nowak	3	3	Statystyka
Adam	Kowalski	2	4	Fizyka
Ewa	Nowak	3	4	Fizyka

(Wykładowca)⋈(Przedmiot)			
Imię	*Nazwisko*	*ID*	*Przedmiot*
Adam	Kowalski	2	Bazy danych
Ewa	Nowak	3	Statystyka

Algebra relacyjna

- **Złączenie wewnętrzne** θ – z wyniku iloczynu kartezjańskiego wybierane są tylko te krotki, których wskazane atrybuty spełniają warunek θ.
  - **Równozłączenie** – warunkiem jest równość. Podobne do naturalnego, ale powtarzające się kolumny nie są pomijane
  - **Warunkowe** – warunek dowolny (inny niż '=')

$$(\text{Wykładowca}) \bowtie_{\text{W.ID=S.ID}} (\text{Przedmiot})$$

Wykładowca		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Ewa	Nowak	3

$\bowtie$

Przedmiot	
ID	*Przedmiot*
2	Bazy danych
3	Statystyka
4	Fizyka

Algebra relacyjna

(Wykładowca)×(Przedmiot)				
Imię	*Nazwisko*	*ID*	*ID*	*Przedmiot*
Adam	Kowalski	2	2	Bazy danych
Ewa	Nowak	3	2	Bazy danych
Adam	Kowalski	2	3	Statystyka
Ewa	Nowak	3	3	Statystyka
Adam	Kowalski	2	4	Fizyka
Ewa	Nowak	3	4	Fizyka

(Wykładowca)⋈$_{=}$(Przedmiot)				
Imię	*Nazwisko*	*ID*	*ID*	*Przedmiot*
Adam	Kowalski	2	2	Bazy danych
Ewa	Nowak	3	3	Statystyka

Algebra relacyjna

- **Złączenie zewnętrzne** – podobne do naturalnego, ale w relacji wynikowej pozostawiane są wiersze nieposiadające odpowiedników w obu relacjach. Może być lewostronne, prawostronne lub dwustronne

$$(\text{Wykładowca}) \bowtie_L (\text{Przedmiot})$$

Wykładowca		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Ewa	Nowak	3
Jan	Kucharski	5

$\bowtie_L$

Przedmiot	
ID	*Przedmiot*
2	Bazy danych
3	Statystyka
4	Fizyka

Algebra relacyjna

(Wykładowca)⟕$_L$(Przedmiot)			
Imię	*Nazwisko*	*ID*	*Przedmiot*
Adam	Kowalski	2	Bazy danych
Ewa	Nowak	3	Statystyka
Jan	Kucharski	5	NULL

Złączenie zewnętrzne lewostronne

(Wykładowca)⟕$_R$(Przedmiot)			
Imię	*Nazwisko*	*ID*	*Przedmiot*
Adam	Kowalski	2	Bazy danych
Ewa	Nowak	3	Statystyka
NULL	NULL	5	Fizyka

Złączenie zewnętrzne prawostronne

Algebra relacyjna

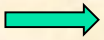
(Wykładowca)⟖(Przedmiot)			
Imię	*Nazwisko*	*ID*	*Przedmiot*
Adam	Kowalski	2	Bazy danych
Ewa	Nowak	3	Statystyka
Jan	Kucharski	5	NULL
NULL	NULL	5	Fizyka

Złączenie zewnętrzne dwustronne

Algebra relacyjna

- **Samozłączenie** – złączenie relacji z samą sobą. Konieczne jest użycie aliasów.

S1	
Nazwisko	*Obecny*
Kowalski	2
Kowalski	3
Kucharski	2



S2	
Nazwisko	*Obecny*
Kowalski	2
Kowalski	3
Kucharski	2

$$(S1) \times (S2)$$

Algebra relacyjna

(S1)×(S2)			
S1.Nazwisko	*S1.Obecny*	*S2.Nazwisko*	*S2.Obecny*
Kowalski	2	Kowalski	2
Kowalski	3	Kowalski	2
Kucharski	2	Kowalski	2
Kowalski	2	Kowalski	3
Kowalski	3	Kowalski	3
Kucharski	2	Kowalski	3
Kowalski	2	Kucharski	2
Kowalski	3	Kucharski	2
Kucharski	2	Kucharski	2

Samozłączenie

Algebra relacyjna

- **Suma** – działa na dwóch zgodnych relacjach (tzn. relacjach o tym samym schemacie) i zwraca wszystkie **różne** wiersze z obu relacji.

(W1)∪(W2)

W1		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Ewa	Nowak	3

∪

W2		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Jan	Kucharski	5

W1∪W2		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Ewa	Nowak	3
Jan	Kucharski	5

Algebra relacyjna

- **Przecięcie** – działa na dwóch zgodnych relacjach (tzn. relacjach o tym samym schemacie) i zwraca wszystkie **wspólne** wiersze z obu relacji.

(W1)∩(W2)

W1		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Ewa	Nowak	3

∩

W2		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Jan	Kucharski	5

W1∩W2		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2

Algebra relacyjna

- **Różnica** – działa na dwóch zgodnych relacjach (tzn. relacjach o tym samym schemacie) i zwraca wszystkie wiersze z pierwszej relacji niewystępujące w drugiej.

(W1)\(W2)

W1		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Ewa	Nowak	3

\

W2		
Imię	*Nazwisko*	*ID*
Adam	Kowalski	2
Jan	Kucharski	5

W1\W2		
Imię	*Nazwisko*	*ID*
Ewa	Nowak	3

Algebra relacyjna

- **Iloraz** – działa na dwóch relacjach z których:
  - jedna jest binarna (2 kolumny) a druga unarna (1 kolumna)
  - w obu występuje taka sama dziedzina (Nazwa kolumny)
  - wyznaczane są wartości z kolumny niewspólnej tabeli binarnej, dla których wartości z kolumny wspólnej odpowiadają wartościom z tabeli unarnej
  - jeżeli wszystkie wyznaczone wartości są takie same, są one wyprowadzane do tabeli wynikowej.

Zajęcia	
Przedmiot	*Data*
Fizyka	2015-10-11
Bazy danych	2015-10-11
Bazy danych	2015-10-12

÷

Przedmioty
Przedmiot
Fizyka
Bazy danych

Wynik
Data
2015-10-11

Model związków encji (ERM)

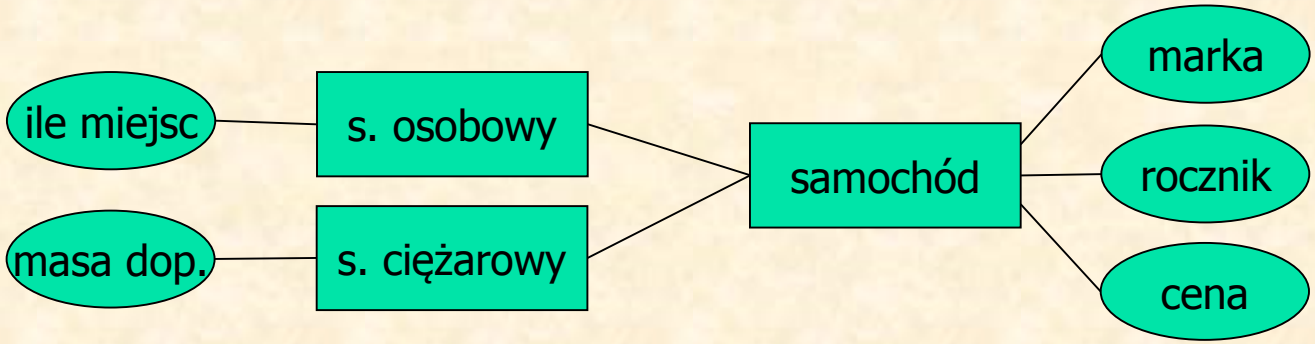
- **Model związków encji** nazywany także **modelem ER** służy do odwzorowania obiektów świata rzeczywistego w pewne abstrakcyjne obiekty (encje), które da się później reprezentować w systemie informatycznym
- Obiekty świata rzeczywistego reprezentowane są przez **encje** a powiązania między obiektami przez **związki** między encjami
- **Diagram związków encji (ERD)** służy do przedstawienia modelu związków encji w postaci graficznej
- ERD może być zapisywany w różnych notacjach:
  - notacja Chena
  - notacja Martina (notacja kruczej stopki)
  - notacja Barkera (Oracle) – najczęściej używana
  - notacja Bachmana
  - notacja UML *(i wiele innych)*

Model związków encji (ERM)

- **Encja** może reprezentować:
  - obiekt materialny, np. studenta, książkę, samochód
  - obiekt niematerialny, np. konto, zlecenie, projekt
  - zdarzenie, np. wysłanie zamówienia, lądowanie samolotu
  - stan rzeczywistości, np. stan rachunku
- Każda encja posiada:
  - unikalną nazwę
  - zbiór atrybutów obejmujący:
    - identyfikator – jednoznacznie identyfikujący wystąpienie encji
    - deskryptory
- Encje mogą wchodzić w związki z innymi encjami
- Każdy obiekt świata rzeczywistego może być reprezentowany tylko przez jedną encję
- Nazwa encji powinna być rzeczownikiem liczby pojedynczej

Model związków encji (ERM)

- **Encja słaba**
  - nie posiada swojego identyfikatora
  - jej wystąpienia mogą istnieć wyłącznie w kontekście wystąpień encji powiązanych z encją słabą
  - przykład: wystąpienie encji słabej **Pozycja_zamówienia** możliwe jest tylko w kontekście wystąpienia encji **Zamówienie**
- **Hierarchia encji** – pewne encje o wspólnym zbiorze atrybutów można uogólnić i stworzyć encję wyższego poziomu (nadencja, encja generalizacji). Encje niższego poziomu nazywane są encjami specjalizacji.



Model związków encji (ERM)

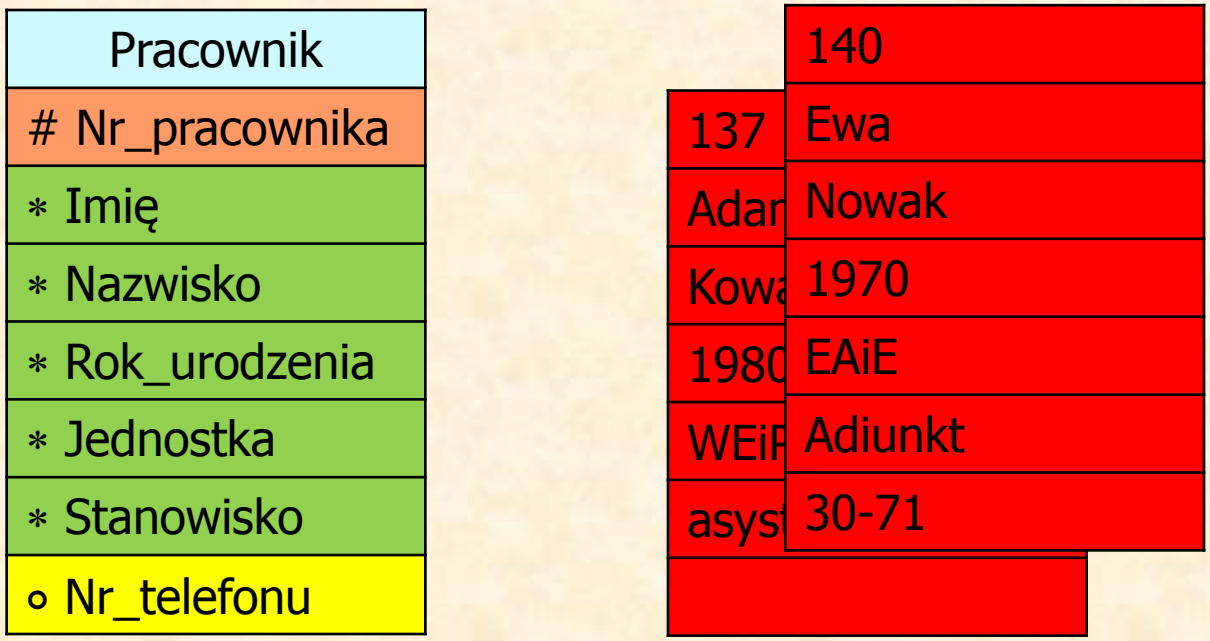
- Atrybuty encji:
  - proste (atomowe) – niepodzielne, np. PESEL
  - złożone – podzielne, np. adres
  - jednowartościowe – jedna wartość w danej chwili, np. wiek
  - wielowartościowe – wiele wartości jednocześnie, np. język obce
  - oryginalne – zapisane w bazie danych, np. PESEL
  - wywiedzione – wyliczone z innych atrybutów, np. wiek
  - obowiązkowe – muszą być podane
  - opcjonalne – mogą być pominięte
- Identyfikatory:
  - proste naturalne, np. PESEL
  - proste sztuczne, np. identyfikator klienta
  - złożone, np. numer rejestracyjny

Cechy związków między encjami

- Stopień związku
  - Unarny, np. Pracownik **jest przełożonym** Pracownika
  - Binarny, np. Osoba **posiada** Paszport
  - Ternarny, np. Student **uczęszcza** na Zajęcia **prowadzone** przez Asystenta
  - N-arny
- Typ związku (kardynalność)
  - jeden do jeden (1:1), np. 1 Osoba posiada 1 Paszport
  - jeden do wiele (1:N), np. 1 Klient składa N Zamówień
  - wiele do wiele (M:N) – *traktowany jako nieimplementowanly*
- Uczestnictwo
  - obowiązkowe, np. Auto jest własnością Osoby
  - opcjonalne, np. Osoba posiada Auto

Diagram związków encji (ERD)

- Encja i wystąpienie encji



Encja

Wystąpienie encji

Diagram związków encji (ERD)

- Oznaczenia atrybutów w notacji Oracle
  - # - identyfikator
  - * - atrybut obowiązkowy
  - o - atrybut opcjonalny
- Zapis atrybutów encji **Pracownik** w notacji Oracle:

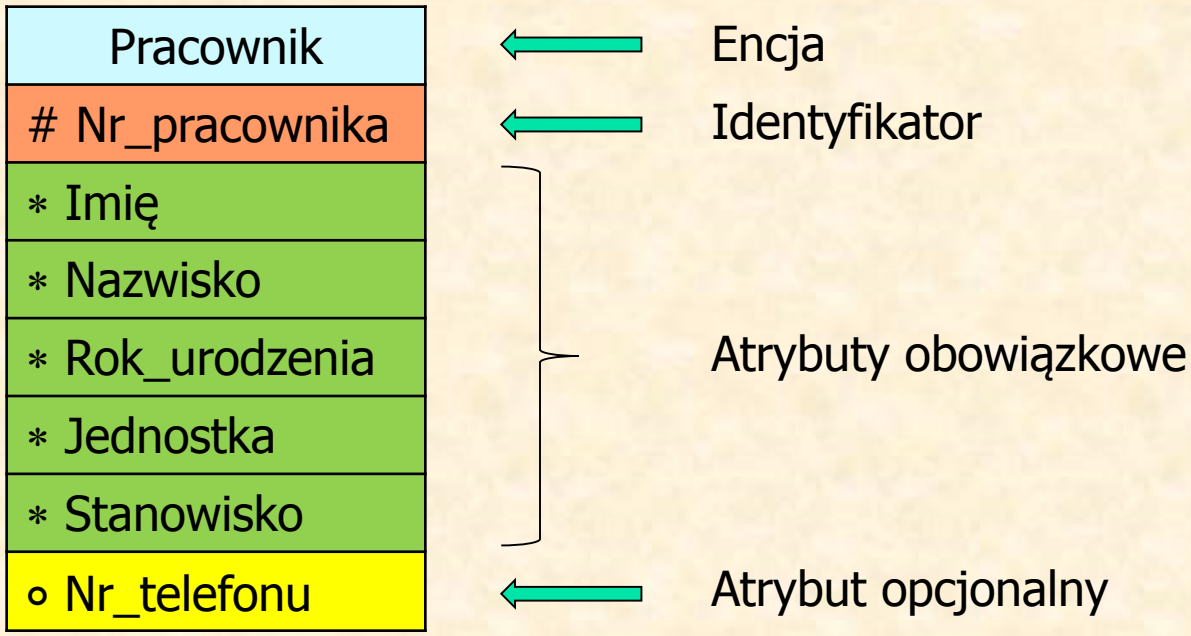
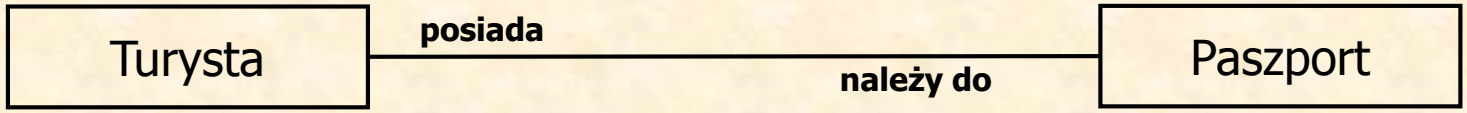


Diagram związków encji (ERD)

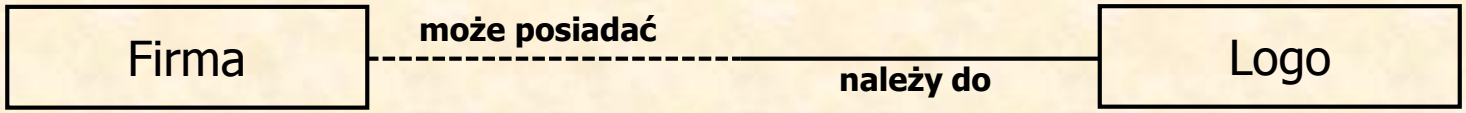
- Oznaczenia cech związków w notacji Oracle
  - Kardynalność
    - (1:1)
    - (1:N)
  - Uczestnictwo
    - obowiązkowe
    - opcjonalne
  - Notacja Martina (kruczej stopki)
    - dokładnie jeden
    - zero lub więcej
    - jeden lub więcej
    - zero lub jeden
    - więcej niż jeden

Diagram związków encji (ERD)

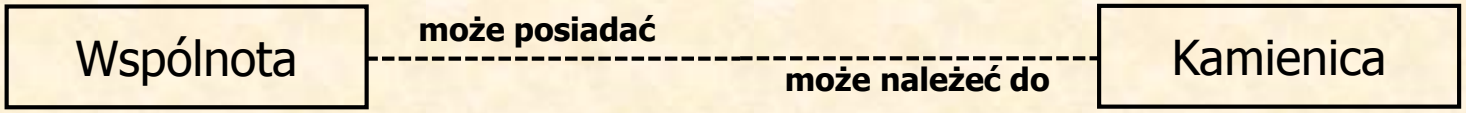
- Oznaczenia cech związków w notacji Oracle



- Turysta posiada dokładnie jeden Paszport
- Paszport należy do dokładnie jednego Turysty



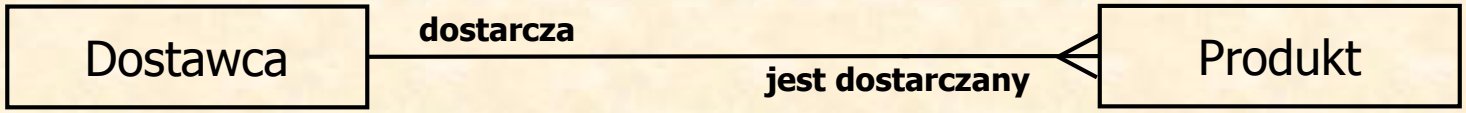
- Firma może posiadać jedno Logo
- Logo należy do dokładnie jednej Firmy



- Wspólnota może posiadać jedną Kamienicę
- Kamienica może należeć do dokładnie jednej Wspólnoty

Diagram związków encji (ERD)

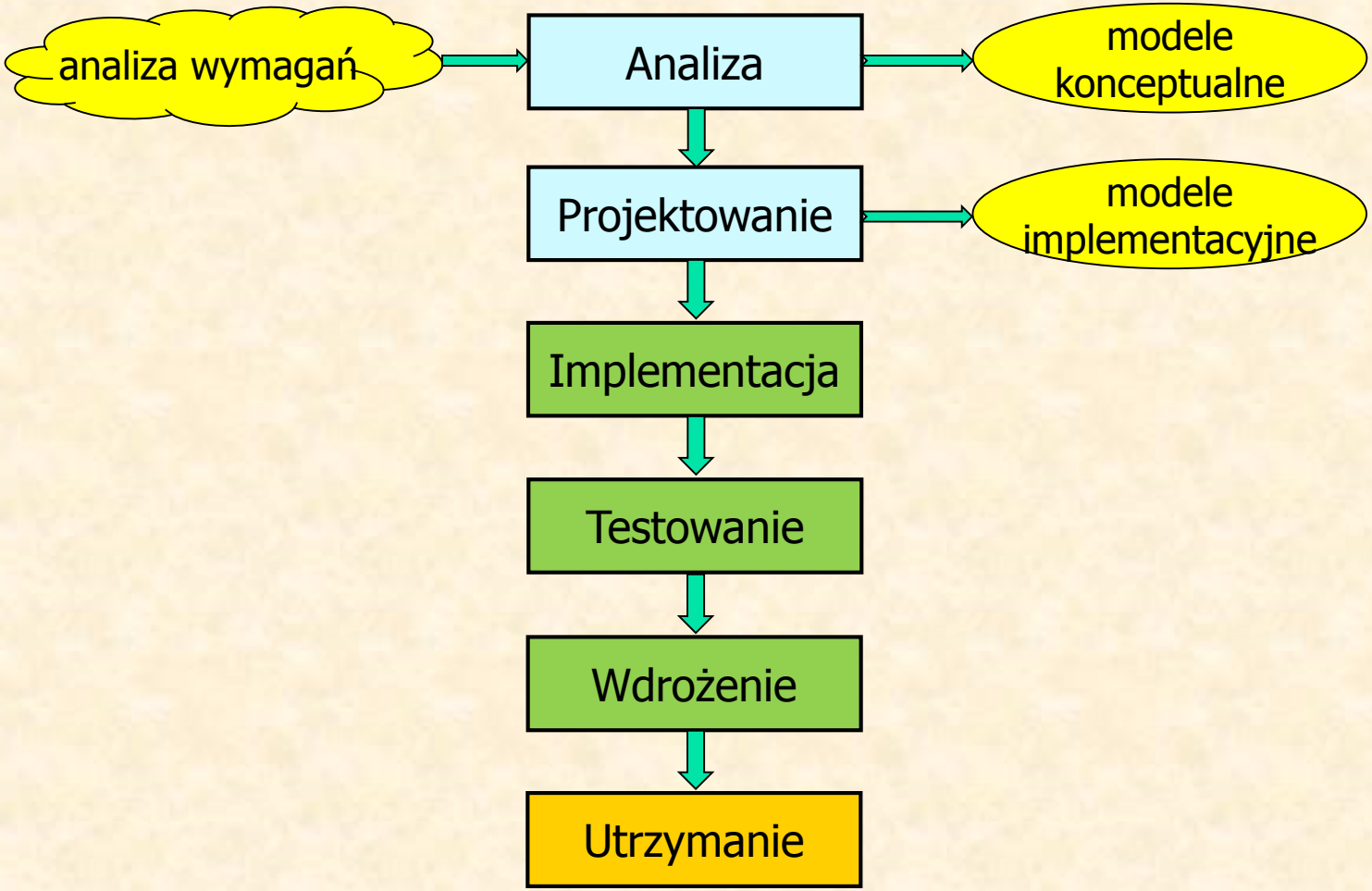
- Ojciec może posiadać wielu Synów
- Syn ma dokładnie jednego Ojca

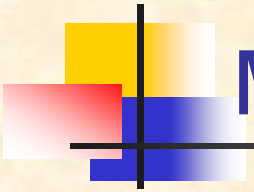


- Dostawca dostarcza Produkty
- Produkty są dostarczane przez jednego Dostawcę

- Produkt jest na Zamówieniach
- Zamówienie zawiera różne Produkty

Cykl projektowy

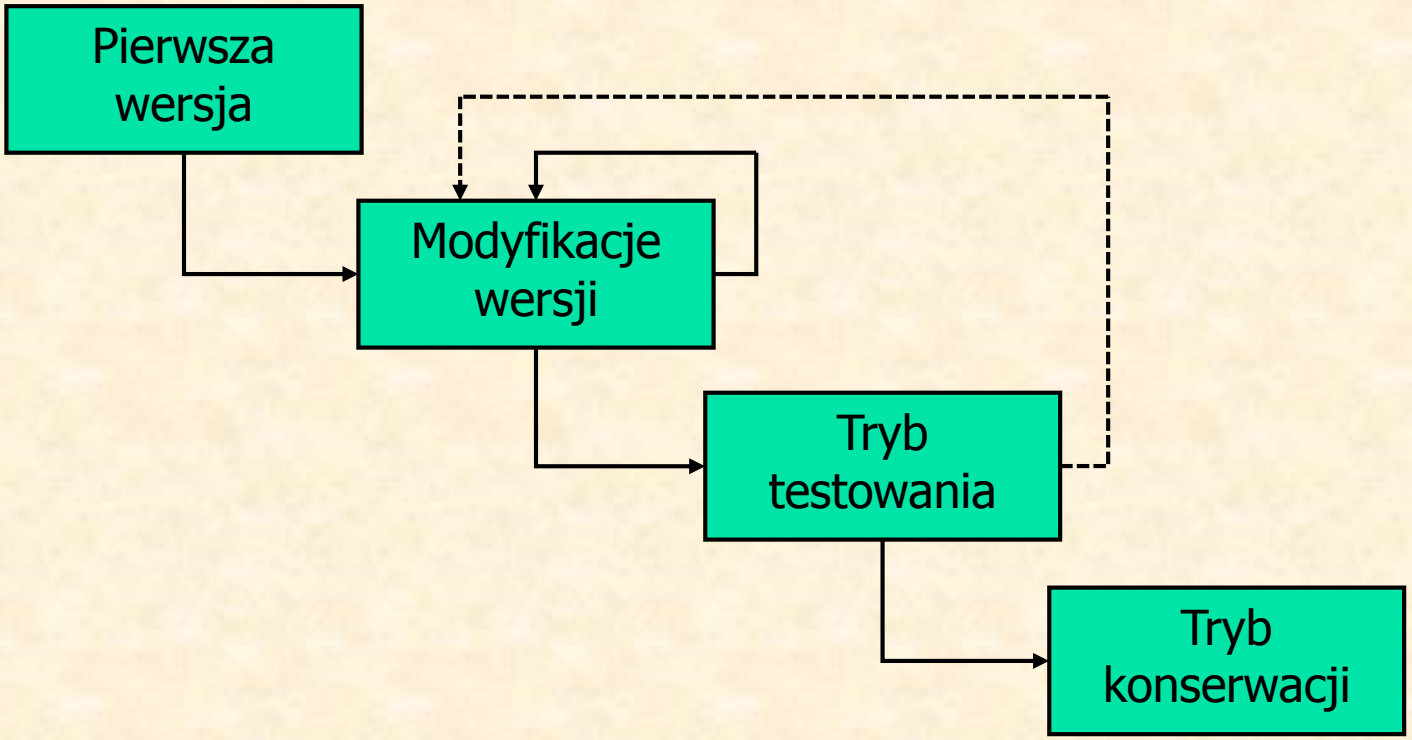




Modele cyklu życia

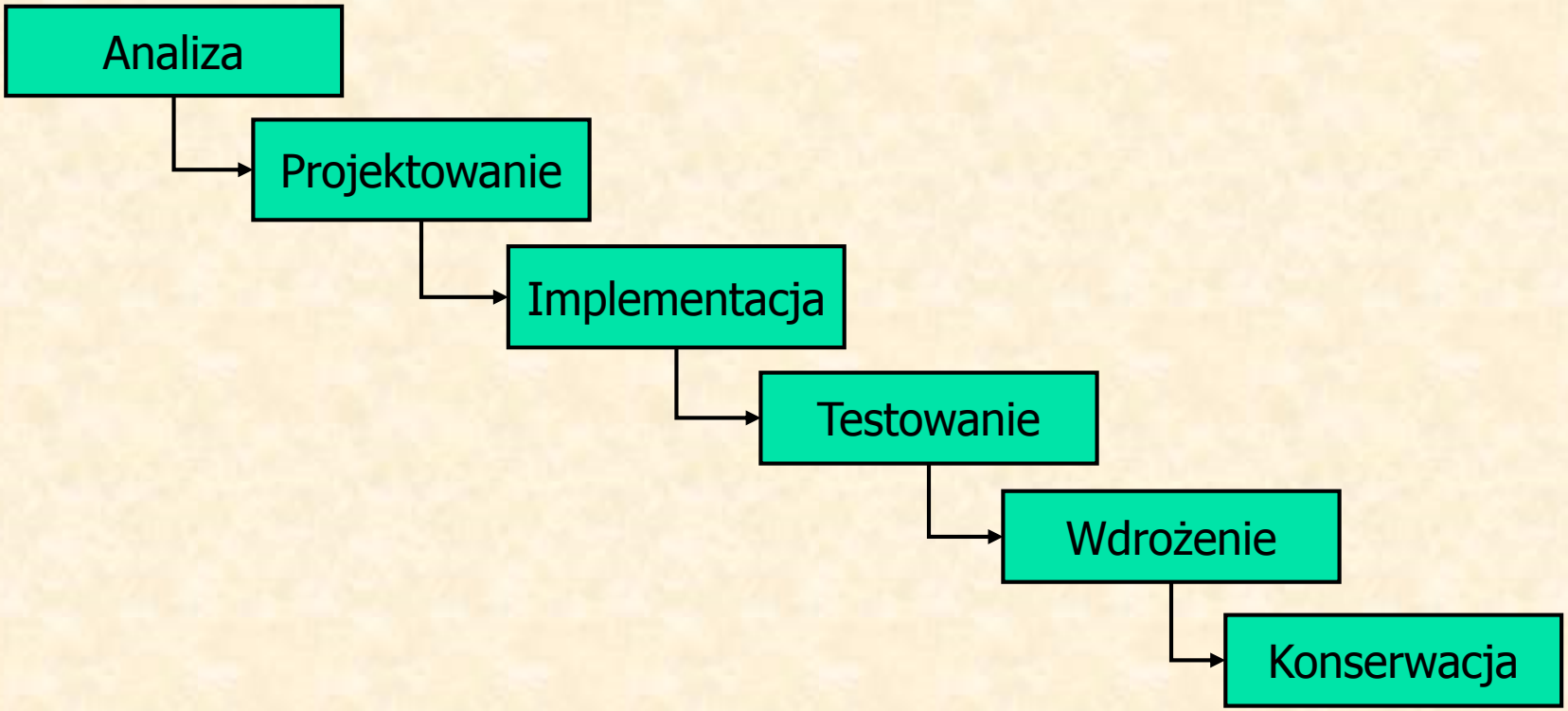
- **Model "buduj i poprawiaj"**
  - Kolejne modyfikacje aż do uzyskania zadowalającego efektu
- **Model kaskadowy**
  - Każdy etap musi być zakończony i zatwierdzony by przejść do następnego
  - Etapy nie zazębiają się
- **Model spiralny**
  - Szereg kolejnych implementacji modelu wodospadowego
  - Każda kolejna iteracja jest rozszerzeniem poprzedniej
- **Model przyrostowy**
  - Analiza i projekt ogólny wykonywane są według modelu kaskadowego
  - Pozostałe etapy wykonywane są według modelu spiralnego
- **Model z prototypem**
  - Model zbliżony do modelu kaskadowego
  - Pierwszym etapem jest stworzenie prototypu

Model "buduj i poprawiaj"

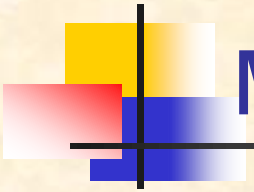


- nadaje się do rozwiązywania niewielkich problemów
- stosowany kiedy nie wiadomo jak zacząć

Model kaskadowy



- dobrze zdefiniowane wymagania
- dobrze określone zastosowania
- rzadko stosowany w postaci czystej



Model kaskadowy

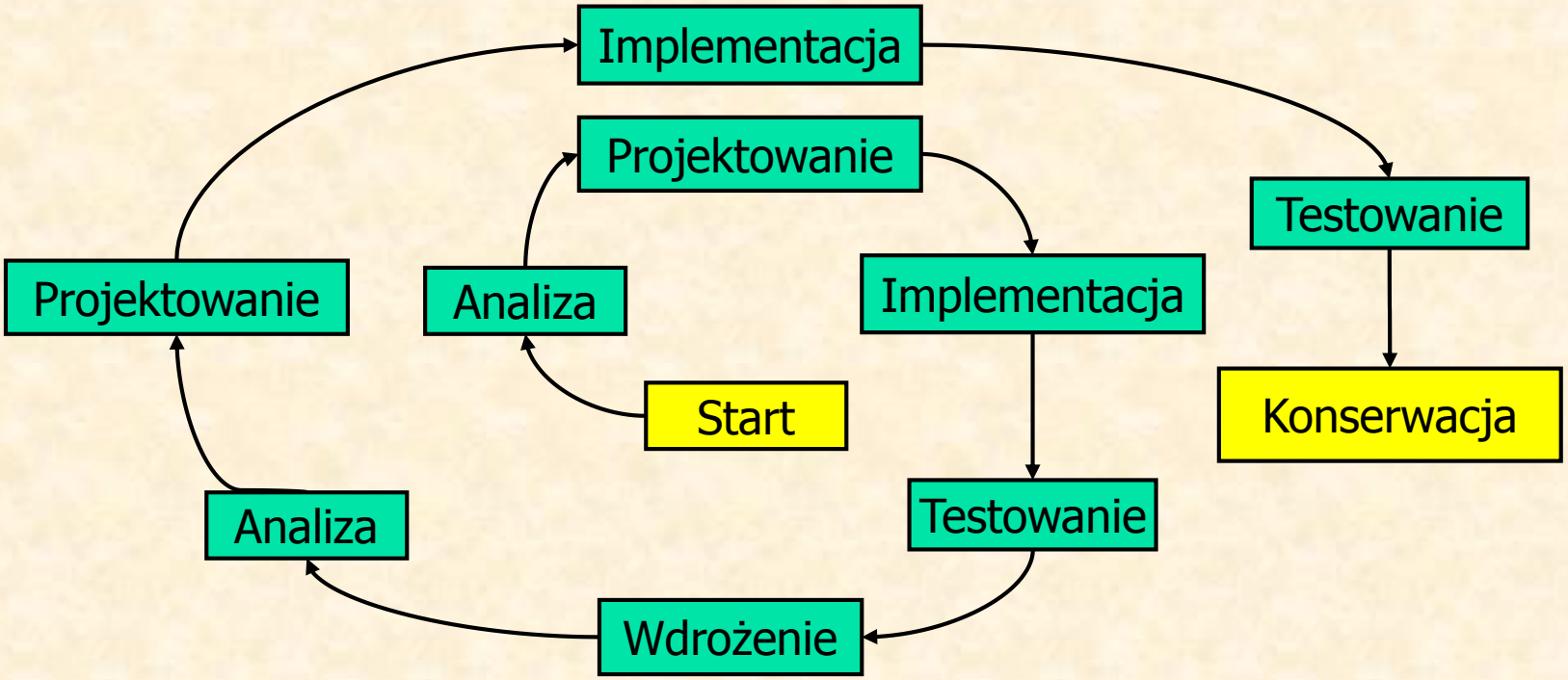
Zalety

- ułatwia organizację przedsięwzięcia
- wymusza dyscyplinę pracy
- wymusza kontrolę wyników każdego etapu pracy
- wymusza dokumentowanie każdej fazy pracy

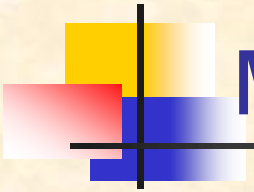
Wady

- trudności w formułowaniu wymagań
- możliwa niezgodność z faktycznymi potrzebami klienta
- narzuca ścisłą kolejność prac
- wymusza oczekiwanie na zakończenie wcześniejszych faz
- wysokie koszty ewentualnych błędów we wcześniejszych fazach

Model spiralny



- eliminuje problemy związane z modelem kaskadowym
- kolejne iteracje rozszerzają wcześniejsze rozwiązania
- duże znaczenie ma właściwa wstępna ocena ryzyka realizacji projektu



Model spiralny

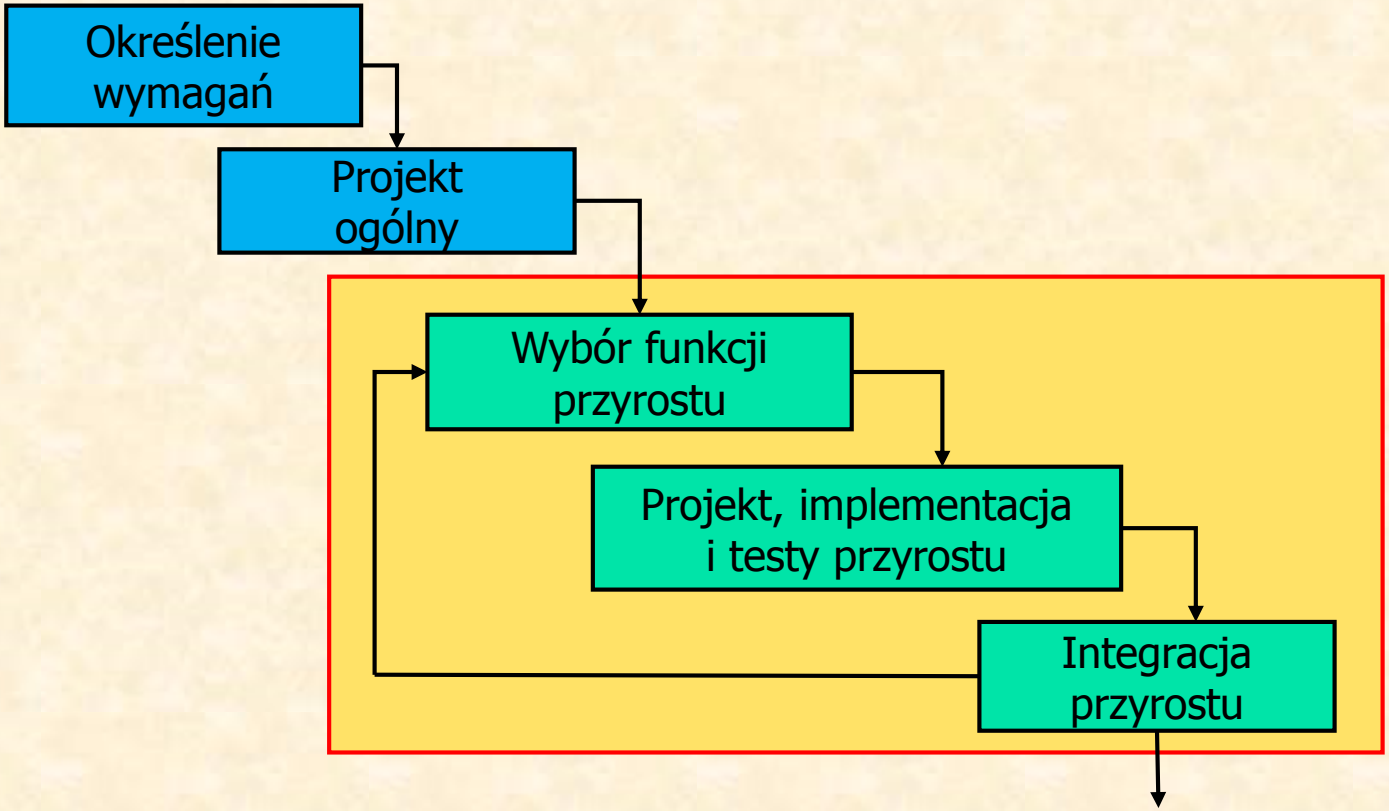
Zalety

- nadaje się do dużych projektów
- eliminuje konieczność powrotu do wcześniejszych etapów
- pozwala szybko reagować na zmieniające się wymagania
- software powstaje już na wczesnym etapie

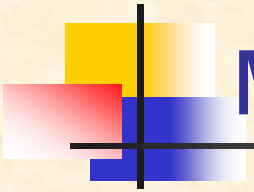
Wady

- nie nadaje się do małych projektów
- kolejne iteracje mogą dezaktualizować wcześniejsze prace
- ostateczna postać systemu nie jest znana do późnych etapów realizacji projektu
- trudności w oszacowaniu czasu koniecznego na ukończenie
- trudności w oszacowaniu kosztów

Model przyrostowy



- stanowi połączenie dwóch poprzednich modeli
- wyodrębnienie podstawowych składowych projektu wg modelu kaskadowego
- kolejne przyrosty realizowane wg modelu spiralnego



Model przyrostowy

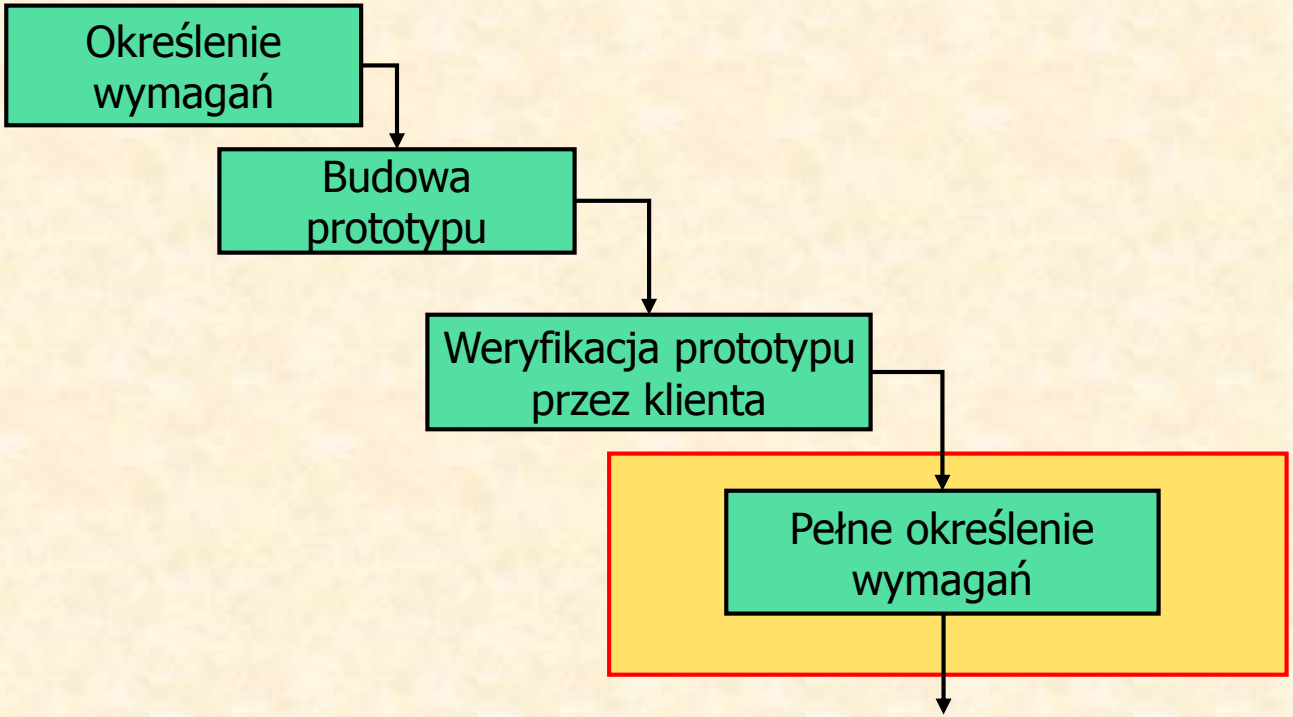
Zalety

- wymusza częsty kontakt z klientem
- nie trzeba od razu formułować pełnych wymagań
- pozwala na wczesne wykrywanie problemów
- opóźnienia pewnych fragmentów nie przenoszą się na całość
- obniża ryzyko projektu
- podstawowe funkcjonalności są dostępne od początku

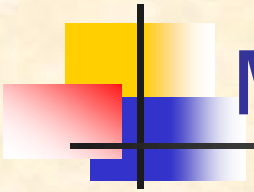
Wady

- trudności z wyodrębnieniem niezależnych funkcjonalności
- kolejne przyrosty nie mogą być duże
- dodatkowy koszt interfejsów zgodnych z docelowym systemem

Model z prototypem



- zmniejsza ryzyko związane z niewłaściwym określeniem wymagań użytkownika
- w ramach prototypu ujmuje się funkcje trudne do określenia i mogące budzić wątpliwości



Model z prototypem

Zalety

- prototyp jest łatwy do zmiany
- zwiększenie zrozumienia potrzeb klienta przez twórców systemu
- przedstawienie klientowi rozwiązania już na wczesnym etapie
- prototyp pozwala na redukcję kosztów wynikających z niezrozumienia z klientem
- prototyp umożliwia wczesne rozpoczęcie szkolenia po stronie klienta
- podstawowe funkcjonalności są dostępne od początku

Wady

- możliwości nieporozumień z klientem wynikających z prezentacji rozwiązania wstępnego
- koszty związane z budową prototypu odrzuconego przez klienta

Projektowanie baz danych

Tworzenie schematu relacyjnej bazy danych

- projektowanie diagramu związków encji
  - wywiad z docelowymi użytkownikami bazy danych
  - zbieranie i analizowanie informacji o wymaganiach
  - przedstawienie wyników w postaci modelu danych
- transformacja diagramu związków encji w schemat relacyjnej bazy danych

Oprogramowanie specjalistyczne – narzędzie CASE (Computer Aided System Engineering)

- tworzenie diagramów związków encji
- transformacja do schematu RBD
- *na przykład Workbench*

Projektowanie baz danych - normalizacja

Normalizacja – proces upraszczania struktury danych tak, aby osiągnęła postać optymalną

- weryfikacja poprawności projektowanych struktur relacji
- dekompozycja na mniejsze schematy o pożądanych cechach
- usunięcie anomalii danych

Anomalie danych

- redundancja (nadmiarowość) – przechowywania tej samej informacji w kilku krotkach
- anomalia wprowadzania – wprowadzenie pewnej informacji wymaga jednoczesnego wprowadzenia innej informacji
- anomalia modyfikacji – modyfikacja informacji dokonywana jest w pewnych krotkach a w innych nie
- anomalia usuwania – usunięcie części informacji powoduje utracenie innych informacji

Projektowanie baz danych - normalizacja

Anomalie danych - przykład

Sklep	Adres	Artykuł	Cena
Żabka	ul. Czarnowiejska 37	masło	3.80
Lewiatan	ul. Mazowiecka 28	masło	4.00
Jubilat	ul. Głowackiego 30	mleko	2.50
Żabka	ul. Czarnowiejska 37	mleko	2.70

- redundancja – adresy i nazwy sklepów występują wielokrotnie
- anomalia wprowadzania – wprowadzenie artykułu wymaga wprowadzenia adresu sklepu
- anomalia modyfikacji – zmiana adresu sklepu musi być wprowadzana we wszystkich krotkach
- anomalia usuwania – usunięcie wszystkich artykułów danego sklepu powoduje usunięcie wszystkich danych sklepu

Projektowanie baz danych - dekompozycja

Dekompozycja – proces eliminowania anomalii przez podział atrybutów relacji wyjściowej R o schemacie $(A_1,A_2,\ldots,A_n)$ między dwie relacje S i T o schematach $(B_1,B_2,\ldots,B_k)$ i $(C_1,C_2,\ldots,C_l)$ według następujących zasad:

- $(A_1,A_2,\ldots,A_n) = (B_1,B_2,\ldots,B_k) \cup (C_1,C_2,\ldots,C_l)$
- krotki relacji S powstają przez pobranie wartości atrybutów z każdej krotki $(A_1,A_2,\ldots,A_n)$ relacji R (rzutowanie). Jeżeli w ten sposób tworzymy powtarzające się krotki, to w relacji S zapisujemy tylko jedną ich kopię
- w analogiczny sposób tworzymy krotki relacji T

Zasady dekompozycji

- zasada zachowania atrybutów
- zasada zachowania informacji
- zasada zachowania zależności funkcyjnych

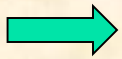
Projektowanie baz danych - dekompozycja

- **Zasada zachowania atrybutów** – w procesie dekompozycji muszą być uwzględnione wszystkie atrybuty – żaden z nich nie może zostać pominięty
- **Zasada zachowania informacji** – krotki w nowych relacjach muszą zawierać wszystkie informacje z kolumn relacji dekomponowanej
- **Zasada zachowania zależności funkcyjnych** – w strukturach nowych tabel muszą być zachowane wszystkie zależności między danymi występujące w tabeli dekomponowanej

- **Dekompozycja odwracalna** – relację pierwotną R można odzyskać przez złączenie naturalne relacji S i T
- **Dekompozycja nieodwracalna** – złączenie naturalne nie daje relacji pierwotnej – powodem jest niewłaściwie przeprowadzona dekompozycja

Projektowanie baz danych - dekompozycja

R		
A	***B***	***C***
1	4	2
3	4	3



S	
A	***B***
1	4
3	4

U

T	
B	***C***
4	2
4	3

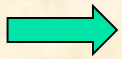
S×T			
A	***B***	***B***	***C***
1	4	4	2
3	4	4	2
1	4	4	3
3	4	4	3

S⋈T		
A	***B***	***C***
1	4	2
3	4	2
1	4	3
3	4	3

Dekompozycja nieodwracalna

Projektowanie baz danych - dekompozycja

R		
A	***B***	***C***
1	4	2
3	4	3



S	
A	***B***
1	4
3	4

U

T	
A	***C***
1	2
3	3

S×T			
A	***B***	***A***	***C***
1	4	1	2
3	4	1	2
1	4	3	3
3	4	3	3

S⋈T		
A	***B***	***C***
1	4	2
3	4	3

Dekompozycja odwracalna

Dane wyjściowe

Dane na fakturze:

• Numer faktury:	123/15		
• Data wystawienia:	2015-09-20		
• Numer klienta:	125		
• Nazwa klienta:	Zakład Mechaniczny Tamech		
• Adres klienta:	Tarnów, ul. Fabryczna 5		
• Nr pozycji faktury:	1	2	3
• Kod towaru:	SM210	PM210	NM210
• Nazwa:	śruba M10	podkładka M10	nakrętka M10
• Cena jedn. zł/kg:	30	30	35
• Ilość kg:	2	1	2
• Wartość zł:	60	30	70
• Suma zł:	160		

Tabela nieznormalizowana

Nr faktury	Data wystawienia	Nr klienta	Nazwa klienta	Adres klienta	Nr pozycji	Kod towaru	Nazwa	Cena	Ilość	Wartość	Suma
123/18	2018-06-21	235	Zakład Mechaniczny Lemko	Jasło, ul. Długa 4	1	SM210	śruba M10	30	2	60	160
					2	PM210	podkładka M10	30	1	30	
					3	NM210	nakrętka M10	35	2	70	

Pierwsza postać normalna 1NF

Relacja jest w pierwszej postaci normalnej jeżeli każdy atrybut niekluczowy jest funkcyjnie zależny od klucza głównego.

Wnioski odnośnie 1NF:

- tabela posiada klucz
- wszystkie wartości kolumn muszą być atomowe (niepodzielne)
- kolumny nie mogą zawierać kolekcji wartości
- nawet w przypadku kolumn atomowych nie mogą występować powtarzające się grupy wartości
- 1NF dotyczy powtarzających się grup danych

Tabela znormalizowana do 1NF

Nr faktury	Data wystawienia	Nr klienta	Nazwa klienta	Adres klienta	Nr pozycji	Kod towaru	Nazwa	Cena	Ilość	Wartość	Suma
123/18	2018-06-21	235	Zakład Mechaniczny Lemko	Jasło, ul. Długa 4	1	SM210	śruba M10	30	2	60	160
123/18	2018-06-21	235	Zakład Mechaniczny Lemko	Jasło, ul. Długa 4	2	PM210	podkładka M10	30	1	30	160
123/18	2018-06-21	235	Zakład Mechaniczny Lemko	Jasło, ul. Długa 4	3	NM210	nakrętka M10	35	2	70	160

Tabela znormalizowana do 1NF

Numer faktury
Data wystawienia
Nr klienta
Nazwa klienta
Adres klienta
Numer pozycji
Kod towaru
Nazwa
Cena
Ilość
Wartość
Suma

Klucz złożony zbudowany z dwóch atrybutów:
- numer faktury
- numer pozycji

Dane są atomowe (nie dzielimy adresu)

Występują zależności częściowe:
- data wystawienia
- numer klienta
- nazwa klienta
- adres klienta

zależą tylko od numeru faktury

Druga postać normalna 2NF

Relacja jest w drugiej postaci normalnej wtedy i tylko wtedy, gdy jest w pierwszej postaci normalnej i każdy atrybut niekluczowy, czyli nie należący do zadanego klucza, jest w pełni funkcyjnie zależny od klucza głównego.

Wnioski odnośnie 2NF:

- tabela jest już w 1NF
- każda kolumna nienależąca do żadnego klucza potencjalnego jest całkowicie zależna od całego klucza głównego
- tabele powinny przechowywać informacje o tylko jednej "rzeczy", opisywaną w całości przez jej klucz główny
- tabela będąca w 1NF może nie być w 2NF tylko jeśli posiada złożony klucz główny

Tabela znormalizowana do 2NF

#	Numer faktury
	Data wystawienia
	Nr klienta
	Nazwa klienta
	Adres klienta
	Suma

#	Numer faktury
#	Numer pozycji
	Kod towaru
	Nazwa
	Cena
	Ilość
	Wartość

Występują zależności tranzytywne:

numer faktury→nr klienta→nazwa klienta

numer faktury→nr klienta→adres klienta

numer faktury,numer pozycji→kod towaru→nazwa

numer faktury,numer pozycji→kod towaru→cena

Trzecia postać normalna 3NF

Relacja jest w trzeciej postaci normalnej wtedy i tylko wtedy, gdy jest w drugiej postaci normalnej i każdy niekluczowy atrybut jest bezpośrednio, a nie przechodnio, zależny od klucza głównego.

Wnioski odnośnie 3NF:

- tabela jest już w 2NF
- żadna informacja w kolumnie, która nie jest kluczem podstawowym, nie może zależeć od niczego innego, jak tylko od klucza podstawowego
- wszystkie kolumny nienależące do żadnego klucza potencjalnego są wzajemnie niezależne
- 3NF jest najwyższym poziomem wymaganym przez większość aplikacji

Tabela znormalizowana do 3NF

#	Numer faktury
	Data wystawienia
	Nr klienta
	Suma

#	Numer faktury
#	Numer pozycji
	Kod towaru
	Ilość
	Wartość

#	Nr klienta
	Nazwa klienta
	Adres klienta

#	Kod towaru
	Nazwa
	Cena

wartość i suma nie muszą być w tabeli – można je obliczyć na podstawie innych danych w widoku

Postać normalna Boyce'a-Codda BCNF

Relacja jest w postaci normalnej Boyce'a-Codda wtedy i tylko wtedy, gdy wyznacznik każdej nietrywialnej zależności funkcyjnej jest nadkluczem.

Wnioski odnośnie BCNF:

- tabela jest już w 3NF
- żaden atrybut nie może zależeć od wyznacznika niebędącego nadkluczem
- tabela będąca w postaci normalnej BCNF nie zawiera danych nadmiarowych
- nie każdą tabelę da się doprowadzić do postaci BCNF
- w praktyce zwykle poprzestajemy na 3NF

Czwarta postać normalna 4NF

Relacja jest w czwartej postaci normalnej wtedy i tylko wtedy, gdy wyznacznik każdej nietrywialnej zależności wielowartościowej jest nadkluczem.

Wnioski odnośnie 4NF:

- 4NF stanowi rozszerzenie BCNF na zależności wielowartościowe
- ponieważ zależności funkcyjne są również zależnościami wielowartościowymi, tabela będąca w 4NF jest już w BCNF
- w 4NF występuje tylko jedna nietrywialna zależność wielowartościowa
- w większości przypadków praktycznych taka postać nie jest wymagana

Piąta postać normalna 5NF

Relacja jest w piątej postaci normalnej wtedy i tylko wtedy, gdy jest w postaci 4NF i nie zawiera zależności złączeniowych, czyli nie istnieje jej odwracalny rozkład na mniejsze tabele.

Wnioski odnośnie 5NF:

- 5NF dotyczy sytuacji w której występują więcej niż dwie zależności wielowartościowe
- w większości przypadków praktycznych taka postać nie jest wymagana

Zależności funkcyjne atrybutów

A→B oznacza, że każdej wartości atrybutu A przyporządkowana jest jednoznacznie wartość atrybutu B. Zależność taka działa w jedną stronę, np. PESEL→NAZWISKO ale nie NAZWISKO→PESEL

Zależności funkcyjne mogą dotyczyć zbiorów atrybutów:

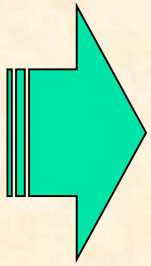
UCZELNIA,WYDZIAŁ→NAZWISKO_DZIEKANA

$A_1,A_2,\ldots,A_n \rightarrow B_1$

$A_1,A_2,\ldots,A_n \rightarrow B_2$

...

$A_1,A_2,\ldots,A_n \rightarrow B_m$



$A_1,A_2,\ldots,A_n \rightarrow B_1,B_2,\ldots,B_m$

$A_1,A_2,\ldots,A_n$ – zbiór determinujący (wyznacznik)

$B_1,B_2,\ldots,B_m$ – zbiór zależny

$A_1,A_2,\ldots,A_n$ jest kluczem głównym relacji R jeśli wszystkie pozostałe atrybuty są funkcyjne zależne od atrybutów klucza

Zależności funkcyjne atrybutów

Aksjomaty Armstronga

Założenie: A,B,C są podzbiorami atrybutów relacji R

- zwrotność – jeżeli $B \subseteq A$ to $A \rightarrow B$ *(zależność trywialna)*
- rozszerzenie – jeżeli $A \rightarrow B$ to $A,C \rightarrow B,C$
- przechodniość – jeżeli $A \rightarrow B$ i $B \rightarrow C$ to $A \rightarrow C$

Z aksjomatów tych wynikają następujące **reguły Armstronga**

- samookreślenie – $A \rightarrow B$
- rozkład – jeżeli $A \rightarrow B,C$ to $A \rightarrow B$ i $A \rightarrow C$
- suma – jeżeli $A \rightarrow B$ i $A \rightarrow C$ to $A \rightarrow B,C$
- złożenie – jeżeli $A \rightarrow B$ i $C \rightarrow D$ to $A,C \rightarrow B,D$

Zależności funkcyjne atrybutów

Domknięcie zbioru atrybutów

Niech dany będzie **zbiór atrybutów A** = $(A_1,A_2,...,A_n)$ oraz zbiór **zależności funkcyjnych F** = $(F_1,F_2,...,F_n)$.

Domknięciem A^+ zbioru atrybutów A nad zbiorem zależności F nazywamy taki zbiór atrybutów B, w którym dla każdego atrybutu B_i, należącego do pewnej relacji R spełniającej zależności F, spełniona jest zależność $A_1,A_2,...,A_n \rightarrow B_i$.

Zbiór A^+ zawiera wszystkie atrybuty zależne funkcyjnie od zbioru atrybutów A

Algorytm wyznaczania domknięcia

Wyznaczanie domknięcia zbioru atrybutów $\{A_1,A_2,...,A_n\}$ oznaczanego $\{A_1,A_2,...,A_n\}^+$

1. Z zależności trywialnej przyjmujemy, że domknięciem jest początkowy zbiór atrybutów, tzn.:
   $\{A_1,A_2,...,A_n\}^+ = \{A_1,A_2,...,A_n\}$
2. Przechodzimy przez wszystkie zależności funkcyjne i do domknięcia dodajemy wszystkie atrybuty stojące z prawych stron zależności wynikających z $A_1,A_2,...,A_n$
3. Postępowanie powtarzamy tak długo jak to jest możliwe
4. Procedurę kończymy jeśli nie są już dodawane kolejne atrybuty

Algorytm wyznaczania domknięcia

Przykład

Relacja ma schemat R(A,B,C,D) i zbiór F zależności funkcyjnych:

F: {A,B→C, C→D, D→A}

1. $\{A\}^+ = \{A\}$,
   $\{B\}^+ = \{B\}$,
   $\{C\}^+ = \{C,D,A\}$, *(nowa zależność C→A)*
   $\{D\}^+ = \{D,A\}$
2. $\{A,B\}^+ = \{A,B,C,D\}$, *(nowa zależność A,B→D)*
   $\{A,C\}^+ = \{A,C,D\}$, *(nowa zależność A,C→D)*
   $\{A,D\}^+ = \{A,D\}$,
   $\{B,C\}^+ = \{B,C,D,A\}$, *(nowa zależności B,C→D i B,C→A)*
   $\{B,D\}^+ = \{B,D,A,C\}$, *(nowa zależności B,D→A i B,D→C)*
   $\{C,D\}^+ = \{C,D,A\}$, *(nowa zależność C,D→A)*

Algorytm wyznaczania domknięcia

3. $\{A,B,C\}^+ = \{A,B,C,D\}$, *(nowa zależność $A,B,C \rightarrow D$)*
   $\{A,B,D\}^+ = \{A,B,D,C\}$, *(nowa zależność $A,B,D \rightarrow C$)*
   $\{A,C,D\}^+ = \{A,C,D\}$,
   $\{B,C,D\}^+ = \{B,C,D,A\}$, *(nowa zależność $B,C,D \rightarrow A$)*
4. $\{A,B,C,D\}^+ = \{A,B,C,D\}$

Klucze:

$\{A,B\}$, $\{B,C\}$, $\{B,D\}$ klucze kandydujące

$\{A,B,C\}$, $\{A,B,D\}$, $\{A,B,C,D\}$ nadklucze

Domknięcie zbioru zależności funkcyjnych:

F^+: $\{A,B \rightarrow C,\ C \rightarrow D,\ D \rightarrow A,\ C \rightarrow A,\ A,B \rightarrow D,\ A,C \rightarrow D,\ B,C \rightarrow A,\ B,C \rightarrow D,$
$B,D \rightarrow A,\ B,D \rightarrow C,\ C,D \rightarrow A,\ A,B,C \rightarrow D,\ A,B,D \rightarrow C,\ B,C,D \rightarrow A\}$

Równoważność pokrycia zbioru zależności

Domknięcie zbioru zależności funkcyjnych F^+

jest to zbiór wszystkich zależności funkcyjnych, które można wyprowadzić z zależności należących do zbioru F

Równoważność zbiorów zależności funkcyjnych

Dwa zbiory zależności funkcyjnych E i F są równoważne jeśli ich domknięcia są takie same, czyli $E^+ = F^+$

Minimalne pokrycie zbioru zależności funkcyjnych F (baza minimalna relacji)

jest to minimalny zbiór zależności funkcyjnych równoważny F

Baza minimalna relacji

Jeżeli **B** jest **minimalną bazą relacji**, to zachodzą następujące warunki:

1. wszystkie zależności w zbiorze B mają jednoelementowe prawe strony
2. usunięcie dowolnej zależności z B powoduje, że B przestaje być bazą (nie jest już równoważne z F)
3. jeśli z dowolnej zależności z B usuniemy jakieś atrybuty z lewej strony zależności, to B przestaje być bazą

Ustalenie bazy minimalnej jest podstawą definiowania poprawnych tablic

Algorytm znajdowania bazy minimalnej

Dla zbioru zależności F szukamy minimalnej bazy G:

1. przyjmujemy, że G = F
2. wszystkie zależności funkcjonalne typu $X \rightarrow A_1,A_2,\ldots,A_n$ rozbijamy na zbiory zależności prostych: $X \rightarrow A_1$, $X \rightarrow A_2$, …, $X \rightarrow A_n$
3. sprawdzamy kolejno, czy po usunięciu dowolnego atrybutu z wyznacznika (tj. lewej strony) zależności dana zależność pozostanie zachowana. Jeżeli tak, to taki atrybut z wyznacznika usuwamy
4. usuwamy z G powtarzające się zależności

Algorytm znajdowania bazy minimalnej

Przykład

Relacja:

R(A,B,C,D,E,F,G,H,I)

Zbiór zależności F:

1. A,B→C,D,E,F,G,H
2. A,C→D,E
3. A→C,D,E
4. B→F,G
5. D→I
6. B,F→G

Chcemy wyznaczyć minimalną bazę G

Algorytm znajdowania bazy minimalnej

Rozkładamy zależności 1,2,3,4 na zależności proste

1. A,B→C
2. A,B→D
3. A,B→E
4. A,B→F
5. A,B→G
6. A,B→H
7. A,C→D
8. A,C→E
9. A→C
10. A→D
11. A→E
12. B→F
13. B→G
14. D→I
15. B,F→G

Algorytm znajdowania bazy minimalnej

Usuwamy zależności zbędne

badamy zależności o wyznacznikach dłuższych od 1

1. zależność A,B→C (1) jest zbędna, bo mamy A→C (9)
2. zależność A,B→D (2) jest zbędna, bo mamy A→D (10)
3. zależność A,B→E (3) jest zbędna, bo mamy A→E (11)
4. zależność A,B→F (4) jest zbędna, bo mamy B→F (12)
5. zależność A,B→G (5) jest zbędna, bo mamy B→G (13)
6. zależność A,C→D (7) jest zbędna, bo mamy A→D (10)
7. zależność A,C→E (8) jest zbędna, bo mamy A→E (11)
8. zależność B,F→G (15) jest zbędna, bo mamy B→G (13)

Zostają zależności: 6,9,10,11,12,13,14

Baza minimalna:

G: {A,B→H, A→C, A→D, A→E, B→F, B→G, D→I}

Algorytm dekompozycji do 3NF

1. wyznaczamy bazę minimalną relacji R
2. dla każdego wyznacznika tworzymy oddzielny schemat relacji, w którym wyznacznik jest kluczem, a pozostałe atrybuty są w bazie przez ten klucz wyznaczane
3. pozostałe atrybuty, które nie znalazły się w żadnej relacji utworzonej w drugim kroku, umieszczamy w nowym schemacie

- wszystkie relacje wynikowe są w 3NF
- dekompozycja zachowuje zależności
- dekompozycja jest odwracalna – istnieje dla niej złączenie bezstratne

Algorytm dekompozycji do 3NF

Rozważam relację R z poprzedniego przykładu

R(A,B,C,D,E,F,G,H,I)

Używam wyznaczonej poprzednio bazy minimalnej:

G: {A,B→H, A→C, A→D, A→E, B→F, B→G, D→I}

Mamy 4 różne wyznaczniki, więc tworzę 4 relacje:

R1(**A**,**B**,H)

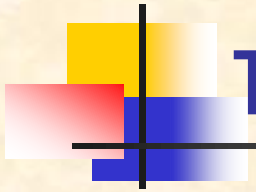
R2(**A**,C,D,E)

R3(**B**,F,G)

R4(**D**,I)

Projektowanie metodą modelowania danych

- Ustalenie potrzebnych encji
- Ustalenie atrybutów encji
- Identyfikowanie cech związków zachodzących między encjami
  - stopień związku
  - typ związku
  - uczestnictwo
- Konwersja związków wieloznacznych (M:N) do postaci związków prostych
- Transformacja diagramu związków encji do modelu relacyjnego
- Implementacja bazy danych (np. SQL)



Transformacja modelu ERD w model relacyjny

- Transformacja encji prostych
- Transformacja hierarchii encji
- Transformacja związków 1:1
- Transformacja związków 1:N
- Transformacja związków M:N
- Transformacja związków n-arnych

Transformacja encji prostych

- Dla każdej encji nie uczestniczącej w hierarchii tworzę relację o nazwie będącej nazwą encji w liczbie mnogiej
- Atrybuty encji przenosimy odpowiednio na atrybuty relacji uwzględniając odpowiednio ich typy
- Unikalny identyfikator encji przenosimy na klucz podstawowy relacji
- Obligatoryjność atrybutu encji przenosimy jako ograniczenie **not null** atrybutu relacji
- Opcjonalność atrybutu encji przenosimy jako **własność null** atrybutu relacji
- Pozostałe ograniczenia integralnościowe atrybutów encji przenosimy na ograniczenia integralnościowe atrybutów relacji

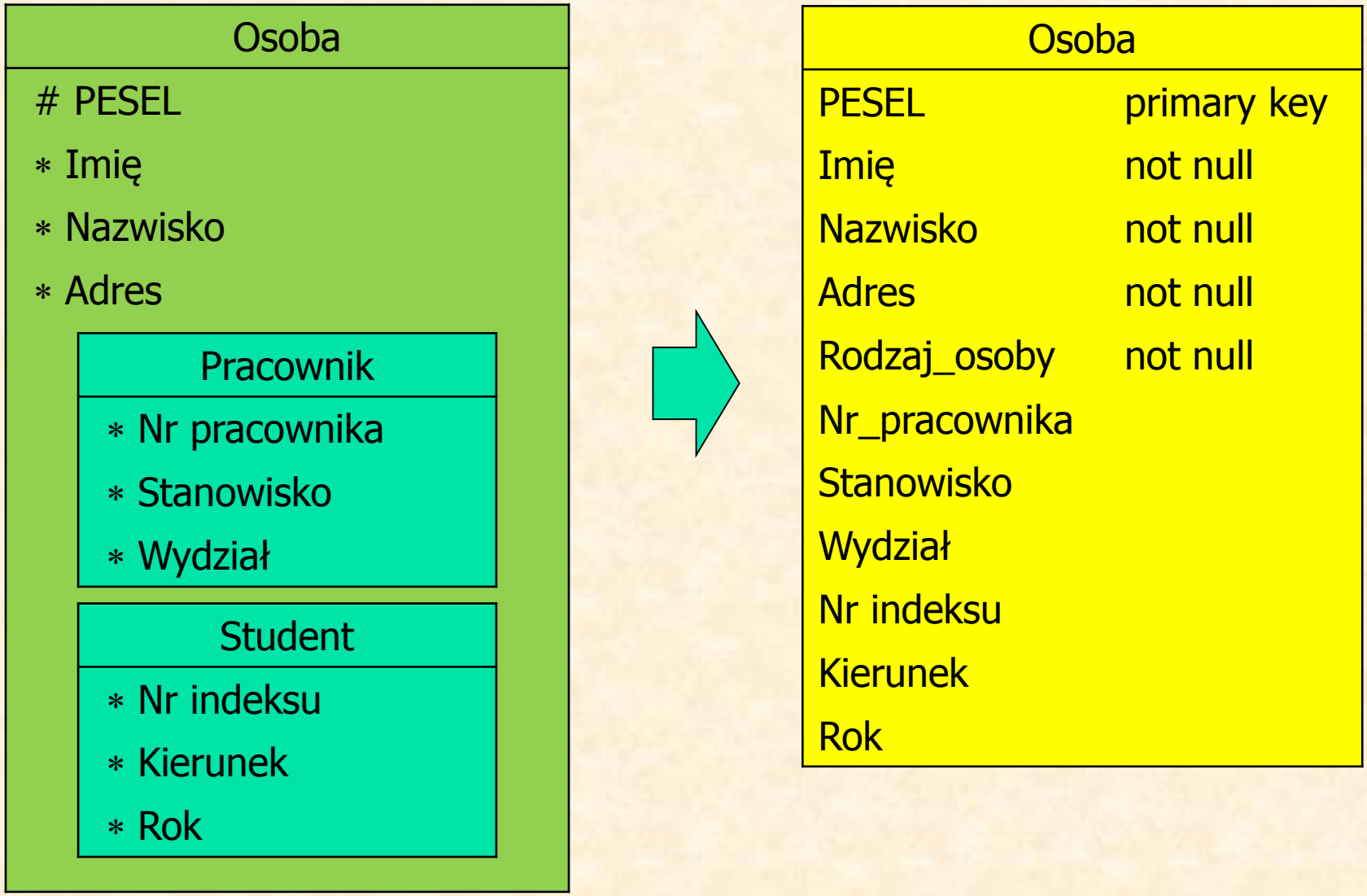
Transformacja hierarchii encji

Transformacja do jednej relacji

- Tworzymy relację zawierającą atrybuty wspólne nadklasy, atrybuty specyficzne wszystkich podklas i atrybut określający typ specjalizacji (do jakiej podklasy dany obiekt należy)
- Wszystkim atrybutom specyficznym poszczególnych podklas nadajemy **własność null**

- Rozwiązanie stosujemy tylko wtedy, gdy podklasy różnią się między sobą minimalnie (np. pojedynczymi atrybutami), a wystąpienie nadklasy należy przynajmniej do jednej z podklas
- W przeciwnym razie w relacji może występować wiele wartości NULL

Transformacja do jednej relacji



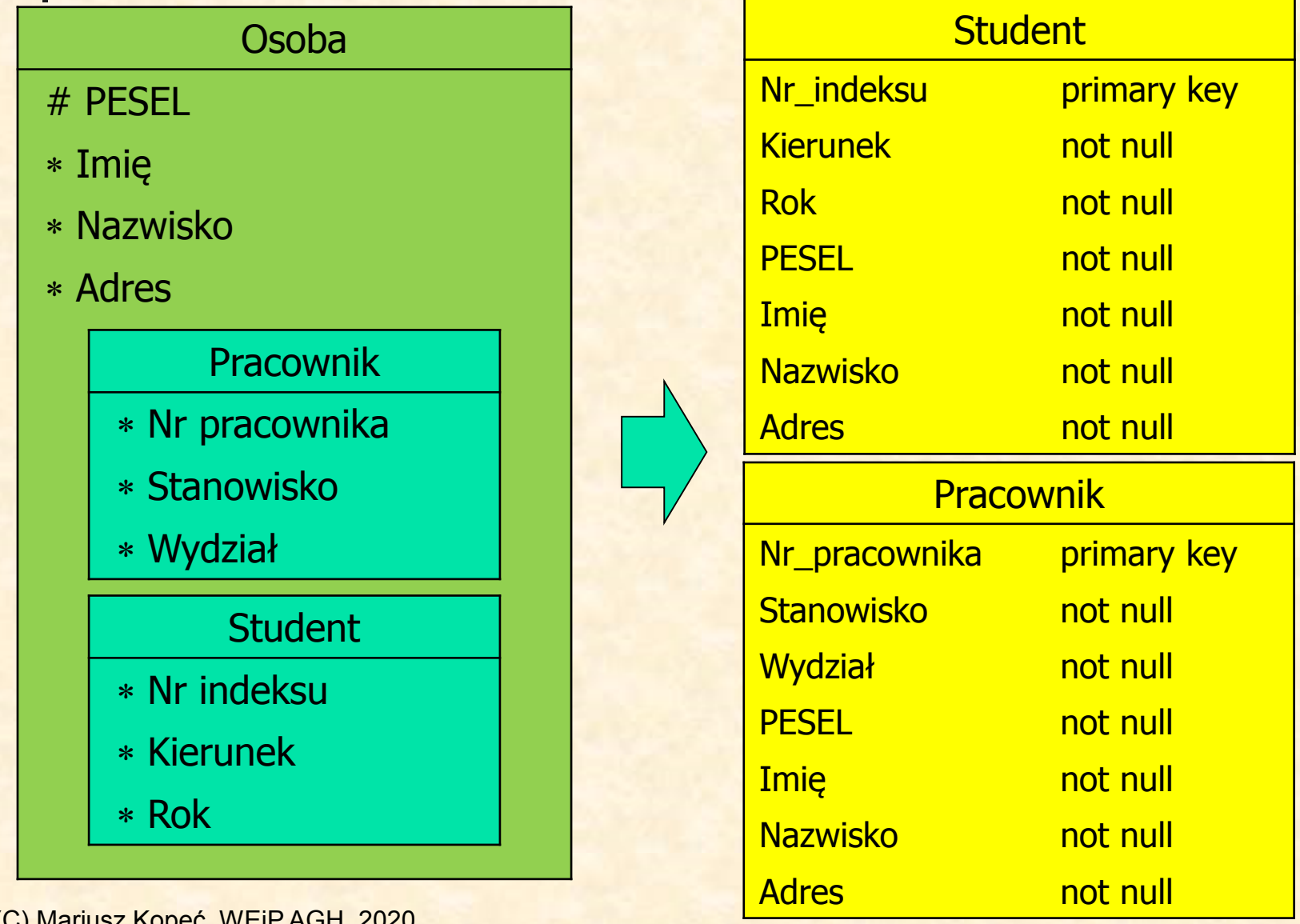
Transformacja hierarchii encji

Transformacja do dwóch relacji

- Dla każdej podklasy tworzymy relację zawierającą atrybuty podklasy oraz atrybuty nadklasy

- Rozwiązanie stosujemy tylko wtedy, gdy podklasy różnią się między sobą znacznie, a wystąpienie nadklasy należy przynajmniej do jednej z podklas
- Transformacja taka prowadzi to utraty korzyści wynikających ze stosowania hierarchii
- Przetwarzanie jest wydajne ze względu na brak konieczności robienia złączeń

Transformacja do dwóch relacji



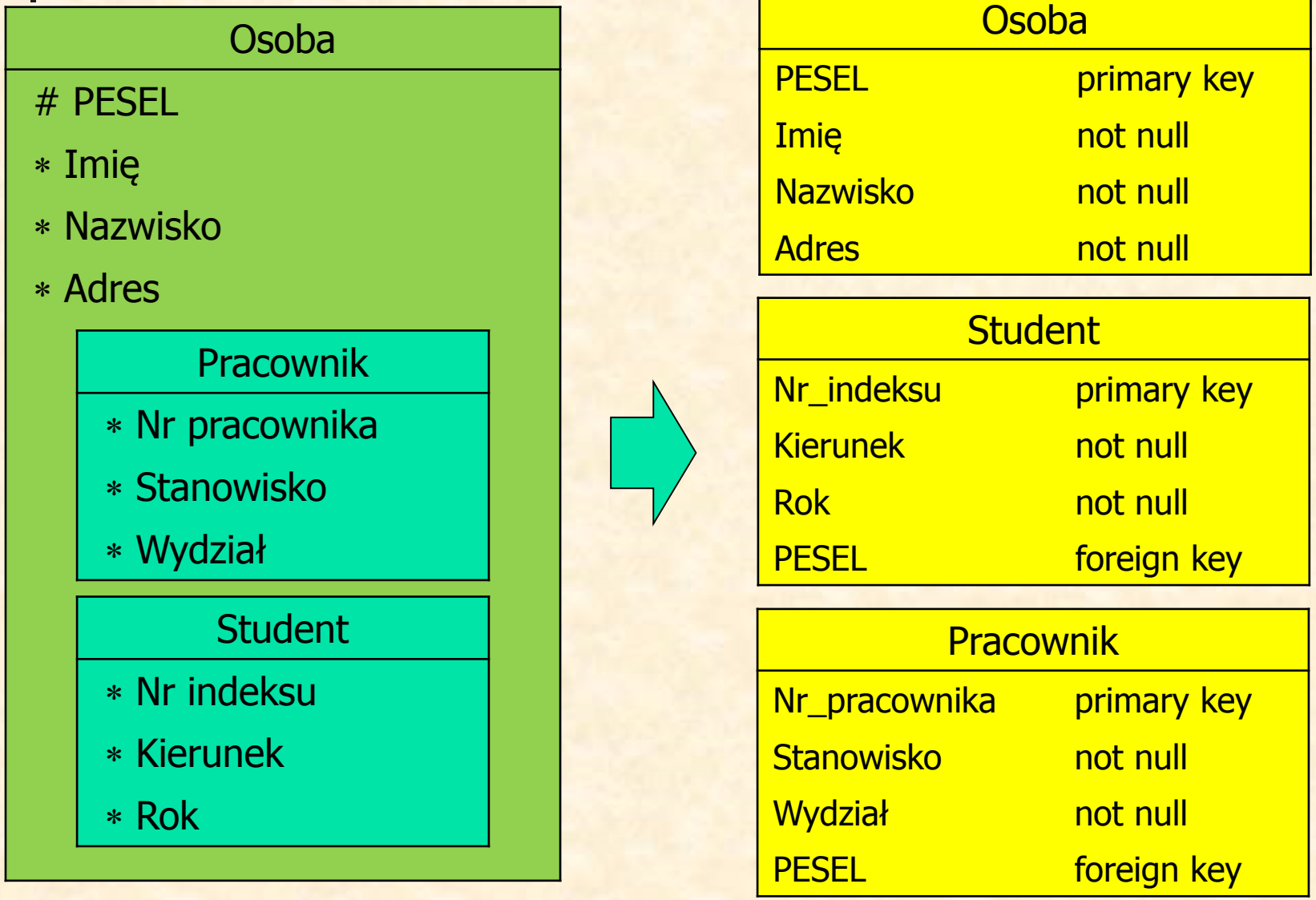
Transformacja hierarchii encji

Transformacja do trzech relacji

- Tworzymy jedną relację zawierającą atrybuty nadklasy (wspólne) i atrybut określający typ podklasy
- Dla każdej podklasy tworzymy relację zawierającą jej atrybuty oraz klucz obcy nadklasy

- Rozwiązanie daje najlepsze wyniki z punktu widzenia normalizacji
- Rozwiązanie jest korzystne jeśli podklasy mają wiele różniących się atrybutów
- Przetwarzanie może być mało wydajne przy dużej ilości złączeń

Transformacja do dwóch relacji



Atrybut a encja

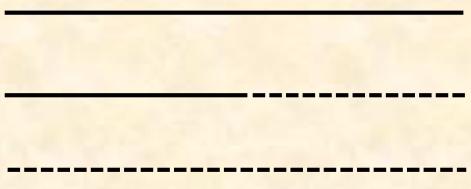
- Atrybut powinien być modelowany jako encja jeśli jest istotną "rzeczą" która posiada własne atrybuty i związki
- Atrybuty mają opisywać encje przy których są umieszczone a nie związki między encjami
- Nazwy atrybutów nie powinny zawierać w sobie nazw encji
- Niektóre dane można modelować na wiele sposobów wymiennie używając, encji, atrybutów i związków. Na przykład możemy stworzyć encję **kupno** z atrybutami ***kupujący***, ***sprzedający*** i ***cena***, albo dwie encje **sprzedający** i **kupujący** połączone związkiem **kupno**
- Związki posiadające atrybuty są modelowane jako encje
- Wybierany sposób modelowania powinien być optymalny z punktu widzenia projektowanego systemu

Kontrola encji

- Każda instancja musi być wyraźnie odróżnialna od innych instancji tej encji
- Każda encja powinna być związana przynajmniej jednym związkiem
- Każda encja powinna posiadać przynajmniej dwa atrybuty
- **Encja podstawowa** – oznacza byty mające zdolność samodzielnego istnienia. Na diagramie z encji podstawowej wychodzą wyłącznie związki opcjonalne
- **Encja zależna (słaba)** – oznacza byty zależne, mogące istnieć wyłącznie w powiązaniu z innymi encjami. Na diagramie z encji zależnej wychodzi co najmniej jeden związek obligatoryjny. Encje zależne są identyfikowane przez związki z innymi encjami

Transformacje związków encji

Kombinacje związków 1:1

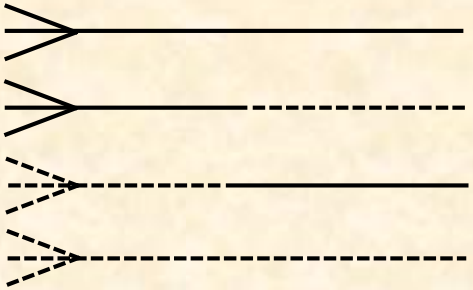


mało prawdopodobna

rzadka

rzadka

Kombinacje związków 1:N



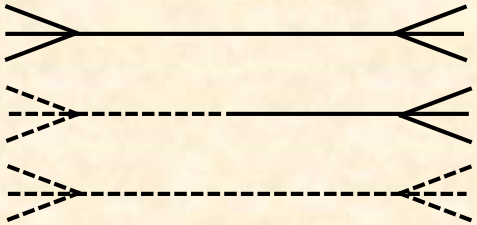
mało prawdopodobna

najczęściej występująca

bardzo rzadka

rzadka

Kombinacje związków M:N



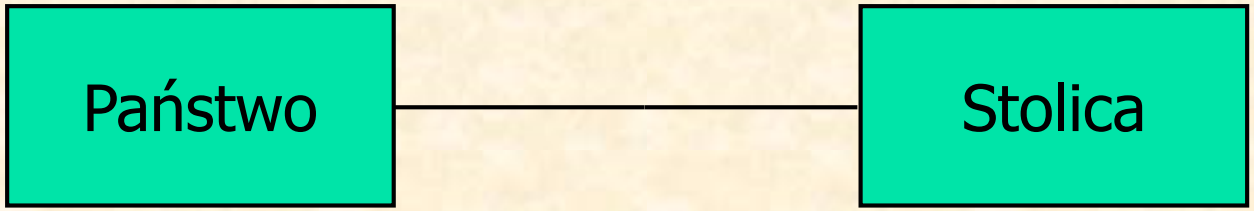
mało prawdopodobna

dość częsta

dość częsta

Szczegóły związków encji

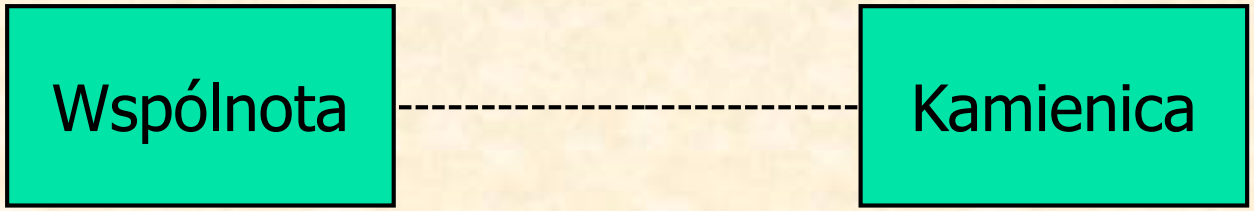
Związek jedno-jednoznaczny obustronnie obowiązkowy



- Związek praktycznie nieprawdopodobny
- Zwykle błędny
- Prawie zawsze przedstawia dwa punkty widzenia na ten sam byt

Szczegóły związków encji

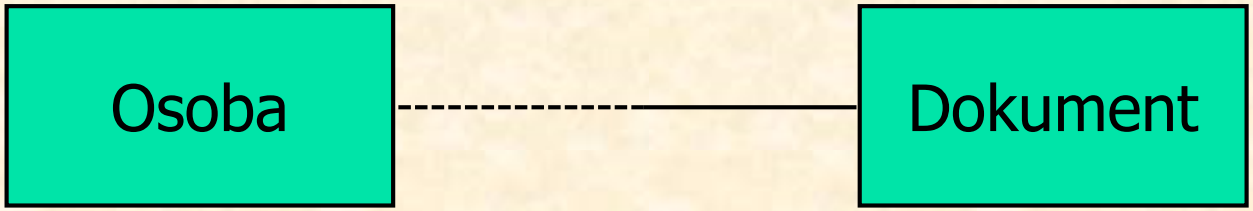
Związek jedno-jednoznaczny obustronnie opcjonalny



- Związek występujący rzadko
- Oba byty mogą istnieć samodzielnie
- Związek łączy w parę dwa samodzielne byty

Szczegóły związków encji

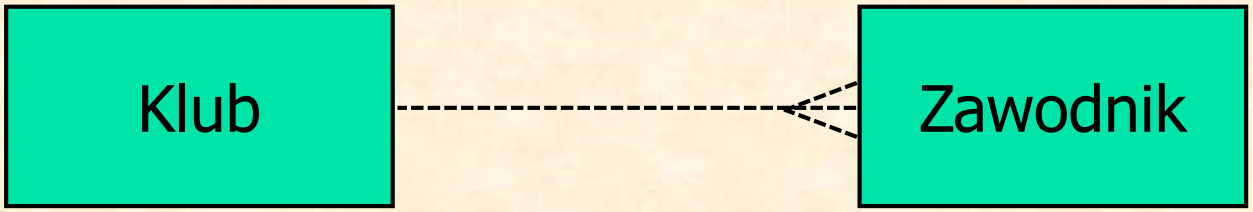
Związek jedno-jednoznaczny opcjonalno-obowiązkowy



- Związek występujący rzadko
- Połączenie w parę bytu niezależnego i zależnego
- Może przedstawiać zestaw opcjonalnych atrybutów wydzielonych w postaci encji zależnej

Szczegóły związków encji

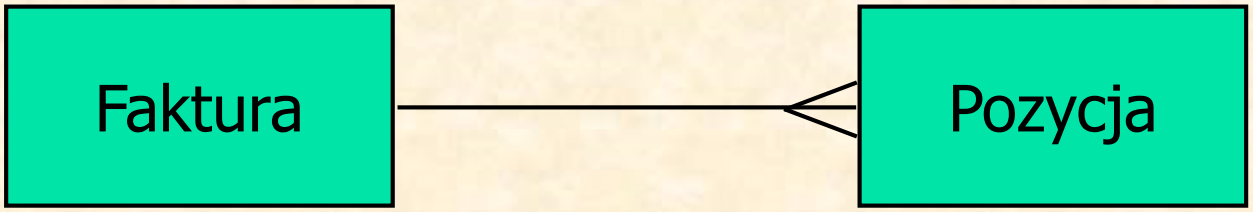
Związek jednoznaczny obustronnie opcjonalny



- Związek występujący rzadko
- Oba byty mogą istnieć samodzielnie
- Pomiędzy bytami istnieje zależność hierarchiczna

Szczegóły związków encji

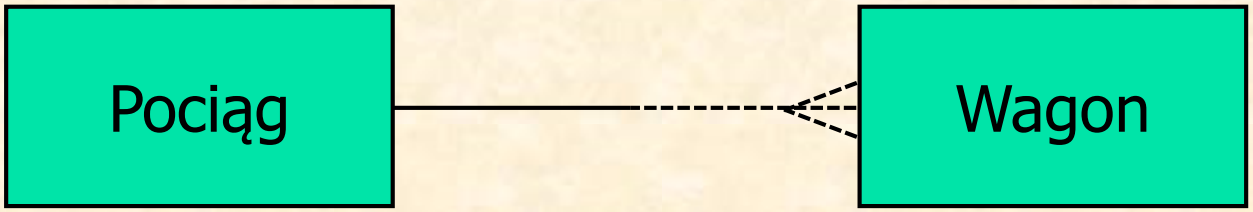
Związek jednoznaczny obustronnie obowiązkowy



- Mało prawdopodobny
- Jeżeli występuje, opisuje obiekt złożony o strukturze hierarchicznej

Szczegóły związków encji

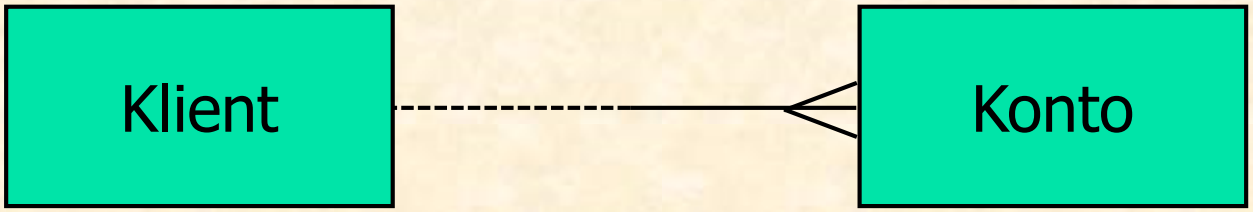
Związek jednoznaczny obowiązkowo-opcjonalny



- Związek występujący bardzo rzadko
- Pomiędzy bytami istnieje zależność hierarchiczna
- Tylko byty podrzędne z punktu widzenia hierarchii mogą istnieć samodzielnie

Szczegóły związków encji

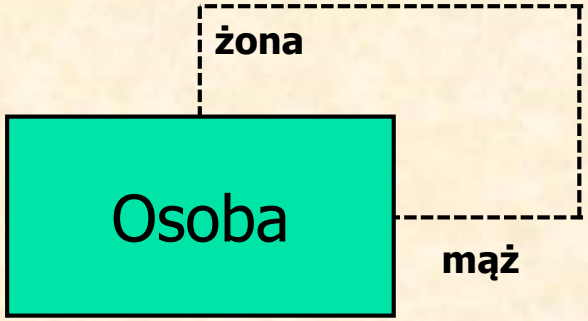
Związek jednoznaczny opcjonalno-obowiązkowy



- Związek występujący często
- Pomiędzy bytami istnieje zależność hierarchiczna
- Tylko byt nadrzędny z punktu widzenia hierarchii może istnieć samodzielnie

Szczegóły związków encji

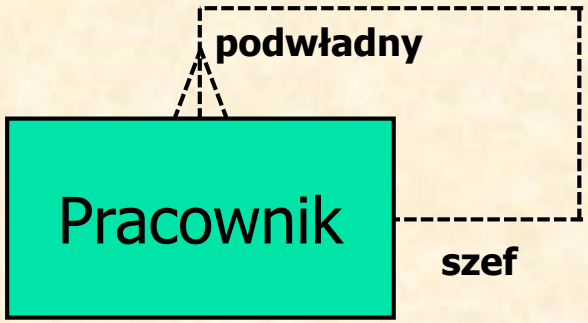
Związek rekurencyjny jedno-jednoznaczny obowiązkowy lub opcjonalny



- Związek występujący bardzo rzadko
- Oznacza połączenie w pary bytów tej samej encji

Szczegóły związków encji

Związek rekurencyjny
jednoznaczny opcjonalny



- Związek występujący często
- Określa hierarchię bytów tej samej encji

Transformacja związków binarnych 1:1

Związek obustronnie obowiązkowy

- Unikalny niepusty klucz obcy umieszczony w relacji o mniejszej przewidywanej liczbie krotek

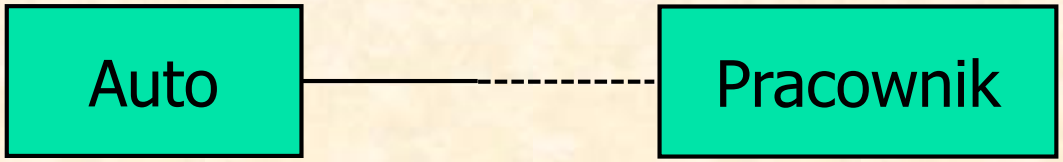
Mężowie	
PESEL	primary key
Imię	not null
Nazwisko	not null
PESEL_ż	not null foreign key references Żony(PESEL)

Żony	
PESEL	primary key
Imię	not null
Nazwisko	not null

Transformacja związków binarnych 1:1

Związek jednostronnie obowiązkowy

- Unikalny niepusty klucz obcy umieszczony w relacji o mniejszej przewidywanej liczbie krotek



Auta	
Nr_rej	primary key
Marka	not null
PESEL_p	not null
foreign key (PESEL_p) references Pracownicy(PESEL)	

Pracownicy	
PESEL	primary key
Imię	not null
Nazwisko	not null

Transformacja związków binarnych 1:N

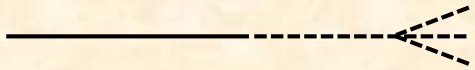
Związek obustronnie obowiązkowy

- W relacji po stronie "wiele" dodajemy niepusty klucz obcy do relacji obowiązkowej

Grupy	
Nr	primary key
Kierunek	not null
Rok	not null

Studenci	
PESEL	primary key
Imię	not null
Nazwisko	not null
Nr	not null
foreign key (Nr) references Grupy(Nr)	

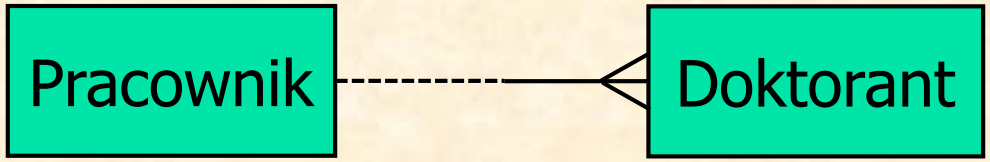
Tak samo dla



Transformacja związków binarnych 1:N

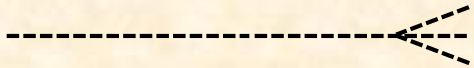
Związek jednostronnie obowiązkowy

- W relacji po stronie "wiele" dodajemy opcjonalny klucz obcy do relacji obowiązkowej



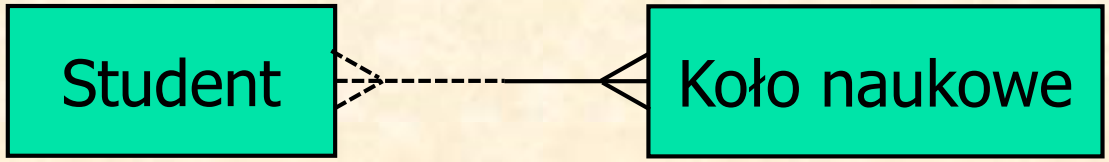
Pracownicy	
Nr_p	primary key
Wydział	not null
Stanowisko	not null

Doktoranci	
Nr_indeksu	primary key
Imię	not null
Nazwisko	not null
Nr_p	
foreign key (Nr_p) references Pracownicy(Nr_p)	

Tak samo dla 

Transformacja związków binarnych M:N

- Tworzymy nową relację Q reprezentującą związek między relacjami R i S (odpowiada encji słabej)
- W Q dodajemy klucze obce do relacji R i S



Studenci	
Nr_i	primary key
Imię	not null
Nazwisko	not null

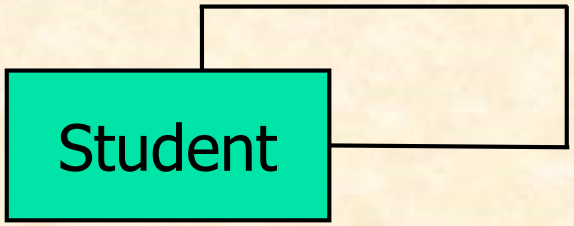
Koła	
Nr_k	primary key
Nazwa	not null

Uczestnictwa	
Nr_i	*not null?*
Nr_k	*not null?*
unique (Nr_i,Nr_k)	
foreign key (Nr_i) references Studenci(Nr_i)	
foreign key (Nr_k) references Koła(Nr_k)	

Transformacja związków unarnych 1:1

Związek obowiązkowy lub opcjonalny

- Do relacji dodajemy klucz obcy (not null lub null) wskazujący na klucz podstawowy tej samej relacji



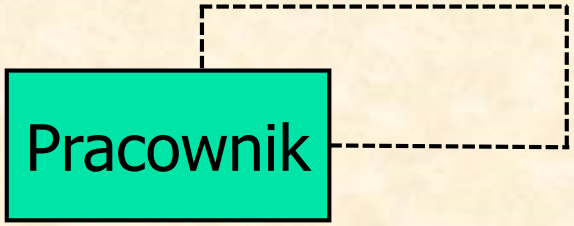
studenci tworzą zespoły dwuosobowe, ale osoby bez pary pracują same

Studenci	
Nr_ind	primary key
Imię	not null
Nazwisko	not null
Para	*not null?*
foreign key (Para) references Studenci(Nr_ind)	

Transformacja związków unarnych 1:N

Związek opcjonalny

- Do relacji dodajemy klucz obcy opcjonalny wskazujący na klucz podstawowy tej samej relacji (not null nie możliwe)



pracownicy mają przełożonego

Pracownicy	
Nr_prac	primary key
Imię	not null
Nazwisko	not null
Szef	
foreign key (Szef) references Pracownicy(Nr_prac)	

Transformacja związków n-arnych

- Dla relacji $R_1,R_2,\ldots,R_n$ będących w związku n-arnym tworzymy relację Q zawierającą klucze obce wskazujące na klucze główne relacji R_i. Jej kluczem głównym jest złożenie wszystkich kluczy obcych.

Studenci	
Nr_ind	primary key
Imię	not null
Nazwisko	not null

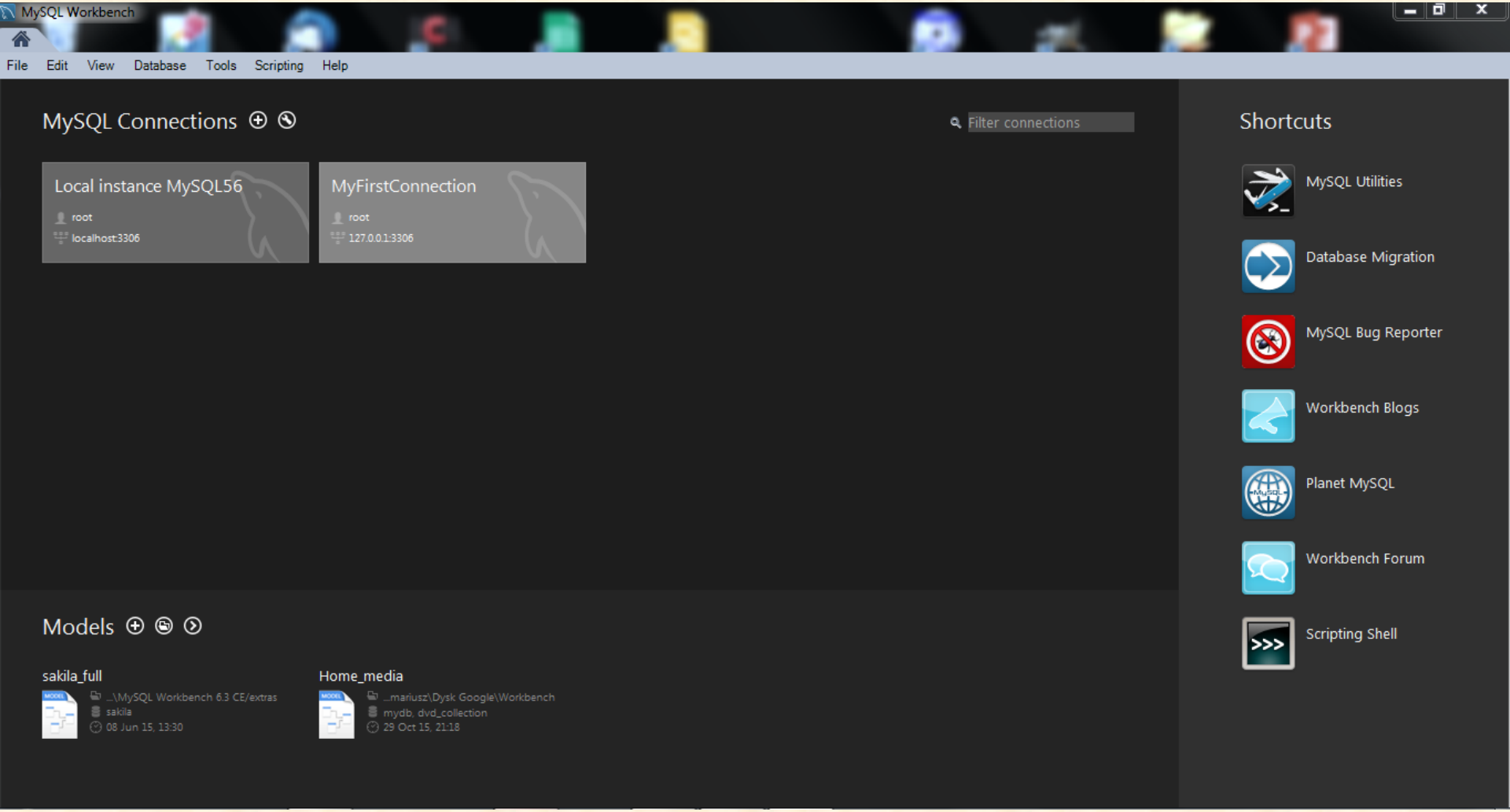
Przedmioty	
Nr_prz	primary key
Nazwa	not null

Wykładowcy	
Nr_wyk	primary key
Imię	not null
Nazwisko	not null

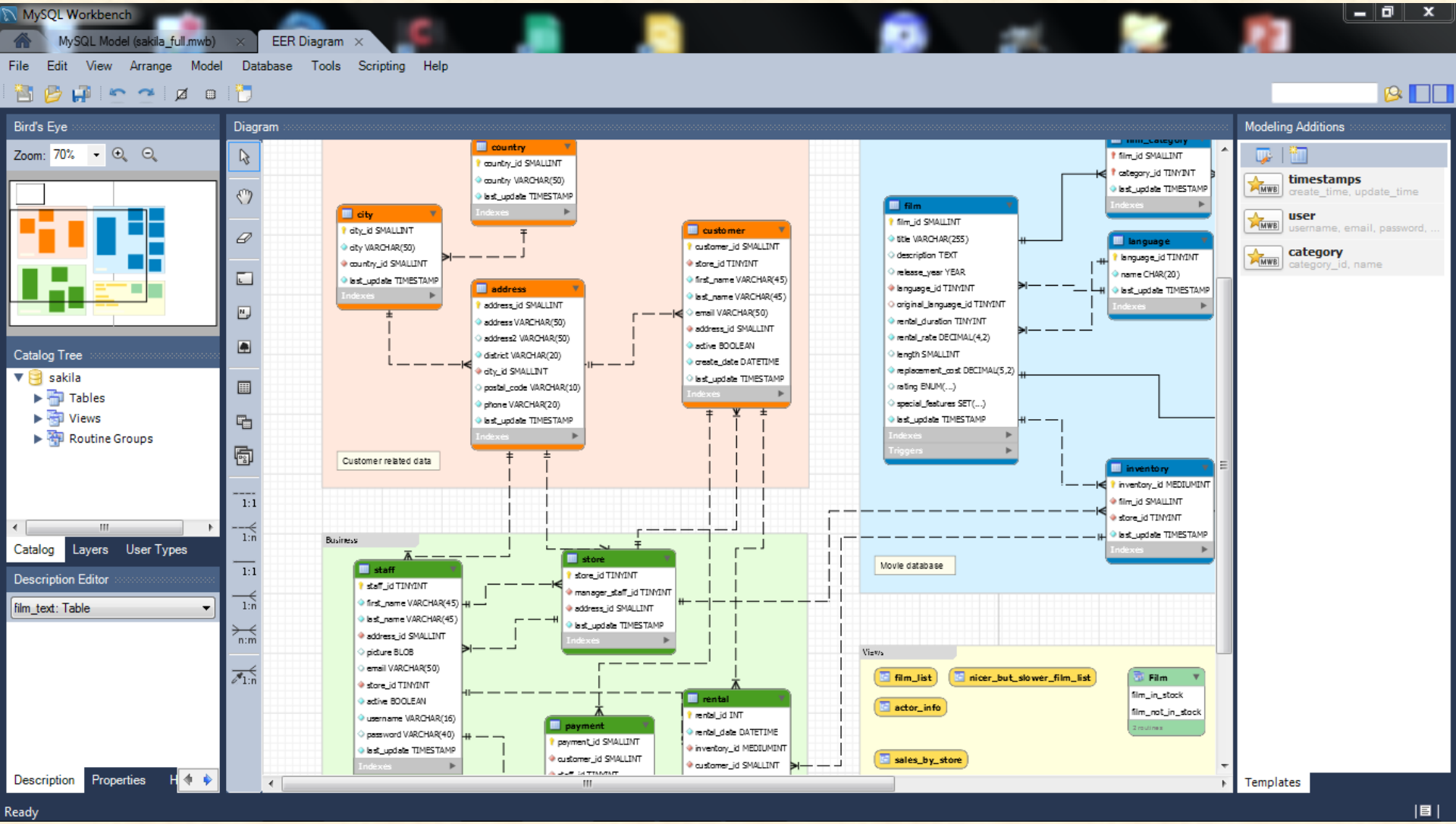
student zdaje egzamin z przedmiotu u wykładowcy

Egzaminy	
Nr_ind	
Nr_prz	not null
Nr_wyk	not null
Egzamin	not null
primary key (Nr_ind,Nr_prz,Nr_Wyk)	
foreign key (Nr_ind) references Studenci(Nr_ind)	
foreign key (Nr_prz) references Przedmioty(Nr_prz)	
foreign key (Nr_wyk) references Wykładowcy(Nr_wyk)	

Workbench


MySQL Workbench
File Edit View Database Tools Scripting Help
MySQL Connections
Filter connections
Local instance MySQL56
root
localhost:3306
MyFirstConnection
root
127.0.0.1:3306
Shortcuts
MySQL Utilities
Database Migration
MySQL Bug Reporter
Workbench Blogs
Planet MySQL
Workbench Forum
Scripting Shell
Models
sakila_full
...\MySQL Workbench 6.3 CE/extras
sakila
08 Jun 15, 13:30
Home_media
...mariusz\Dysk Google\Workbench
mydb, dvd_collection
29 Oct 15, 21:18

Workbench


MySQL Workbench
MySQL Model (sakila_full.mwb)
EER Diagram
File Edit View Arrange Model Database Tools Scripting Help
Bird's Eye
Zoom: 70%
Catalog Tree
sakila
Tables
Views
Routine Groups
Catalog Layers User Types
Description Editor
film_text: Table
Description Properties
Diagram
country
city
address
customer
film
language
inventory
staff
store
rental
payment
Customer related data
Business
Movie database
Views
film_list
nicer_but_slower_film_list
actor_info
sales_by_store
Film
Modeling Additions
timestamps
user
category
Templates
Ready

Bezpłatnie dostępne bazy danych

- ASDL chemia analityczna
  http://home.asdlib.org/
- Arnetminer informatyka
  https://aminer.org/
- arXiv fizyka, matematyka, informatyka, statystyka
  http://arxiv.org/
- ADS astrofizyka, fizyka, geofizyka
  http://adswww.harvard.edu/
- BASE multidyscyplinarna
  https://www.base-search.net/
- Chemxseer chemia
  http://chemxseer.ist.psu.edu/
- CHBD medycyna
  http://www.aina.ucalgary.ca/chbd/

Bezpłatnie dostępne bazy danych

- Citebase matematyka, fizyka, informatyka
  http://adsabs.harvard.edu/
- CiteUlike informatyka
  http://www.citeulike.org/
- CiteSeerX matematyka, informatyka, statystyka
  http://citeseerx.ist.psu.edu/index
- DBLP informatyka
  http://www.dblp.org/search/
- Directory of Open Access Journals multidyscyplinarna
  https://doaj.org/
- DTIC obronność
  http://www.dtic.mil/dtic/
- GOLM spektrometria masowa
  http://gmd.mpimp-golm.mpg.de/

Bezpłatnie dostępne bazy danych

- Index Copernicus multidyscyplinarna
  http://www.indexcopernicus.com/
- MedlinePlus medycyna
  https://www.nlm.nih.gov/medlineplus/
- NEA-DB fizyka jądrowa
  http://www.oecd-nea.org/dbdata/
- NIST fizyka
  http://www.nist.gov/pml/
- OSTI informacje techniczne DoE
  http://www.osti.gov/home/
- PubChem chemia
  http://pubchem.ncbi.nlm.nih.gov/
- PubMed biomedycyna
  http://www.ncbi.nlm.nih.gov/pubmed/
- Science.gov multidyscyplinarna
  http://www.science.gov/