

## Bazy danych

**Jacek Rumiński**



**Adam Bujnowski**

wykład 5



## Plan wykładu

1. Struktura danych w modelu relacyjnym: krotki i relacje
2. Warunki integralności modelu relacyjnego

Model relacyjny oraz modele z nim powiązane są obecnie najczęściej implementowane w oprogramowaniu systemów zarządzania bazami danych (SZBD).

Model relacyjny (w odniesieniu do definicji modelu danych) jest opisywany poprzez trzy podstawowe elementy:

- strukturę danych (relacje, zwane również tabelami),
- reguły integralności danych,
- operatory przetwarzania danych (algebra relacyjna).

Model relacyjny został sformułowany przez pracownika firmy IBM **Edgara Franka Codd**a w 1969. W kolejnych latach zarówno Codd jak i inne osoby wprowadziły szereg zmian w definicjach elementów modelu relacyjnego.

Zacznijmy jednak od samej nazwy modelu.

Model relacyjny związany jest z podstawową strukturą danych określaną poprzez **relację**. Zanim rozważymy czym jest relacja wprowadźmy pojęcie **krotki**.

Krotka reprezentuje uporządkowaną listę elementów. Zapisywana jest z użyciem nawiasów „( )” oraz przecinkiem jako znakiem rozdzielającym elementy, np. (1, 3, 5, 34). Krotka  $n$  elementowa ( $n$ -krotka) to ogólnie  $n$ -tka uporządkowana, którą dla  $n \geq 2$  możemy zdefiniować jako

1.  $(a1, a2) = \{\{a1\}, \{a1, a2\}\}$ ,
2.  $(a1, a2, \dots, an+1) = ((a1, a2, \dots, an), an+1)$ .

Przykłady krotek:

**(‘Jacek’, ‘Rumiński’, ‘dr inż.’, 40),  
(3453, (‘Jacek’, ‘Rumiński’), ‘adiunkt’).**

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

W teorii zbiorów krotka może mieć charakter funkcji wiążącej indeks z wartością, np.  $(a, b, c) = \{ (1, a), (2, b), (3, c) \}$ .

W analogiczny sposób możemy zapisać krotkę jako:

**( nazwisko : 'Rumiński', wiek : 40 ),**

która wiąże nazwę atrybutu z wartością.

Wiedząc czym jest krotka (*n-tka*) podajmy teraz uproszczoną definicję relacji:

**Relacją** *n*-członową ( $n \geq 2$ ) nazywamy dowolny podzbiór zbioru uporządkowanych *n*-tek.

Liczba członów w relacji określa jej rodzaj jako relację binarną ( $n=2$ ), tetrarną ( $n=3$ ), czy ogólnie *n*-arną.

**Przykład relacji binarnej:**  $\{ ('Jacek', 'Adam'), ('Wiesław', 'Michał'), ('Adam', 'Wiesław') \}$ . W przypadku relacji tetrarnej będziemy mieli 3 elementy w krotce, itd. Dowolny podzbiór zbioru krotek to również zbiór pusty (relacja pusta).

Dla relacji można określić dziedzinę, przeciwdziedzinę oraz pole relacji.  
Rozważając relację binarną  $R$  dziedziną będą wszystkie pierwsze elementy uporządkowanych krotek, które są powiązane z innymi elementami (drugimi w relacji binarnej) przez relację  $R$ :

$$D = \{a1 : \exists a2 (a1Ra2)\}.$$

Analogicznie przeciwdziedziną relacji binarnej  $R$  będzie zbiór:

$$D^* = \{a2 : \exists a1 (a1Ra2)\}.$$

Pole relacji to unia zbiorów dziedziny i przeciwdziedziny, czyli:

$$F = D \cup D^*.$$

Analizując poprzedni przykład relacji binarnej

**$\{('Jacek', 'Adam'), ('Wiesław', 'Michał'), ('Adam', 'Wiesław')\}$ ,**

określimy jej dziedzinę jako zbiór:

**$D = \{ 'Jacek', 'Wiesław', 'Adam' \},$**

przeciwdziedzinę jako:

**$D^* = \{ 'Adam', 'Michał', 'Wiesław' \}$**

oraz pole jako:

**$F = \{ 'Jacek', 'Wiesław', 'Adam', 'Michał' \}.$**

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

Znając podstawowe pojęcia, możemy przytoczyć definicję relacji podaną przez twórcę modelu relacyjnego, Codda:

***Mając zbiory  $S_1, S_2, \dots, S_n$ , (nie koniecznie różne) wówczas  $R$  jest relacją na tych  $n$  zbiorach jeśli jest zbiorem  $n$ -krotek, takich że każda ma pierwszy element ze zbioru  $S_1$ , drugi element ze zbioru  $S_2$ , itd.***

Zbiór (ang. set)  $S_j$  jest  $j$ -tą dziedziną relacji  $R$ , gdzie  $n$  jest stopniem relacji.

***Relację  $R$  może reprezentować macierz, która będzie miała następujące własności:***

- (1) każdy wiersz reprezentuje  $n$ -krotkę relacji  $R$ ,
- (2) porządek wierszy nie jest istotny,
- (3) każdy wiersz jest inny,
- (4) kolejność kolumn jest istotna, odpowiada uporządkowaniu dziedzin  $S_1, S_2, \dots, S_n$ , na których zdefiniowana jest relacja  $R$ ,
- (5) znaczenie każdej kolumny jest częściowo wyrażane poprzez etykietę opisującą dziedzinę.



# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

Poniżej zaprezentowano macierz, jako formę reprezentacji relacji **R** o nazwie „dostawa”:

<b>dostawca</b> (etykieta dziedziny/zbioru $S_1$ )	<b>kod towaru</b> (etykieta dziedziny/zbioru $S_2$ )	<b>cena jednostkowa</b> (etykieta dziedziny/zbioru $S_3$ )	<b>ilość</b> (etykieta dziedziny/zbioru $S_4$ )
Firma A	1	12	100
Firma B	3	10000	20
Firma C	2	89	40

Analizując powyższy przykład możemy wyróżnić 4 zbiory:

***Dostawca*** = {‘Firma A’, ‘Firma B’, ‘Firma C’};

***Kod towaru*** = {1, 3, 2},

***Cena jednostkowa*** = {12, 10000, 89},

***Ilość*** = {100, 20, 40}.

# Podstawy modelu relacyjnego



Przedmiot: *Bazy danych*

Politechnika Gdańska, *Inżynieria Biomedyczna*

Relacja R ukazana w formie macierzy jest **podzbiorem** iloczynu kartezyjskiego czterech zbiorów  $R \subseteq \text{dostawca} \times \text{kod towaru} \times \text{cena jednostkowa} \times \text{ilość}$ .

**Iloczyn kartezyjski wygenerowałby wszystkie możliwe kombinacje 4 elementowe ze zbiorów.**

Przykład iloczynu (relacja = tabela):

tabela T

Nazwisko
Kowalski
Nowa
Kamieńska

tabela T1

Miejsce pracy
Szpital Kliniczny
Akademia Medyczna

**iloczyn  
T i T1**

tabela T2

Nazwisko	Miejsce pracy
Kowalski	Szpital Kliniczny
Kowalski	Akademia Medyczna
Nowa	Szpital Kliniczny
Nowa	Akademia Medyczna
Kamieńska	Szpital Kliniczny
Kamieńska	Akademia Medyczna

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

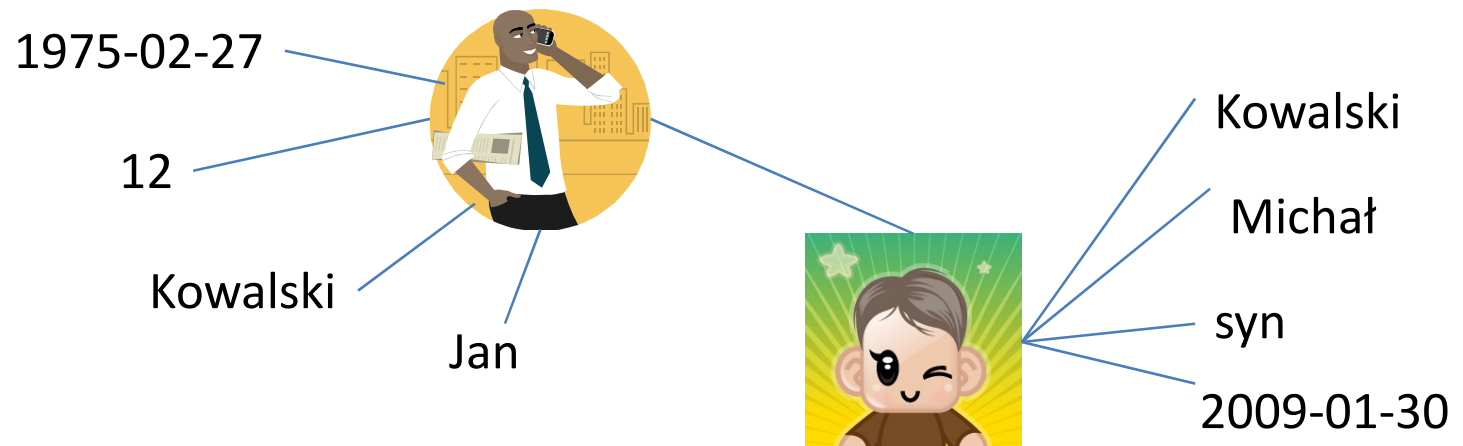
Relacja **R** ukazana w formie macierzy jest podzbiorem iloczynu kartezyjskiego czterech zbiorów  $R \subseteq \text{dostawca} \times \text{kod towaru} \times \text{cena jednostkowa} \times \text{ilość}$ . Iloczyn kartezyjski wygenerowałby wszystkie możliwe kombinacje 4 elementowe ze zbiorów.

Relacja, której dziedziny są proste może być reprezentowana przez dwuwymiarową macierz.

***Co jednak, jeśli dziedzina nie jest prosta, czyli nie zawiera atomowych (pojedynczych) wartości lecz np. krotki?***

Rozpatrzmy przykład krotki opisującej pracownika firmy A:

***KROTKA 1: (12, 'Kowalski', 'Jan', ('Kowalski', 'Michał', 'syn', 2009-01-30), 1975-02-27).***



# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

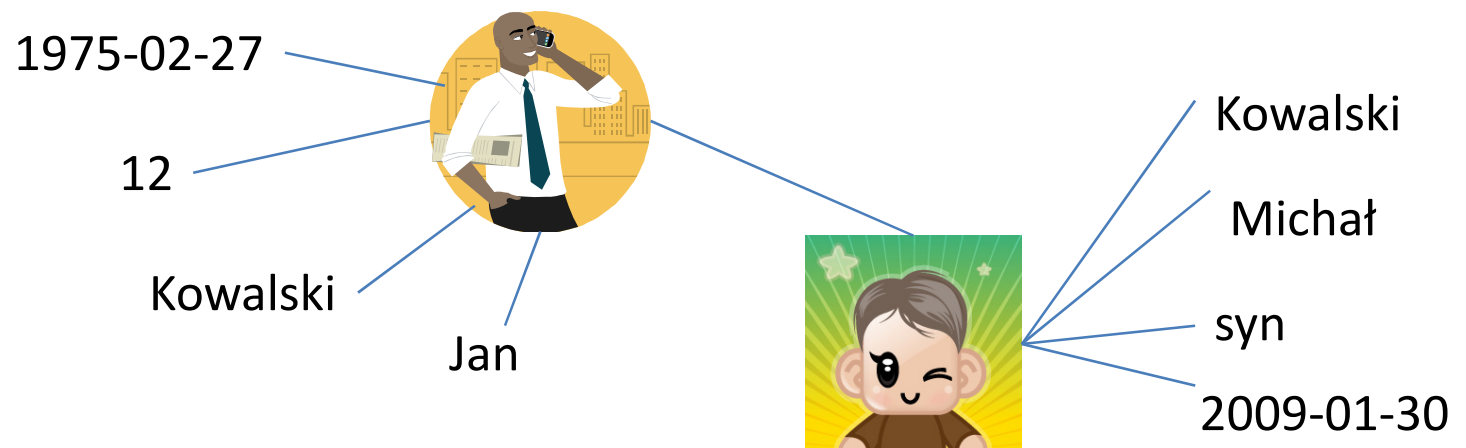
Etykiety dziedzin tej przykładowej krotki to

***{identyfikator, nazwisko, imię, dziecko, data urodzenia}.***

Łatwo zauważyć, że dziedzina „dziecko” nie jest prosta, składa się z dziedzin o etykietach:

***{nazwisko, imię, pokrewieństwo, data urodzenia}.***

**Oczywiście krotki 1 nie da się przedstawić w formie dwuwymiarowej macierzy. Co zrobić?**



Konieczna jest transformacja schematu relacji obejmującej krotkę (KROTKA1) do takiej postaci, aby wszystkie dziedziny były atomowe (czyli inaczej, aby nie było krotek złożonych). W naszym przykładzie musimy utworzyć nową relację jako zbioru krotek opisujących dzieci, a stosowne krotki ojca i syna powiązać związkiem (np. używając identyfikatorów).

Proces taki został nazwany przez Codd normalizacją.

Ponieważ w późniejszym czasie wypracowano (również Codd) szereg form (stopni) normalizacji, tę podstawową transformację dziedzin, na dziedziny atomowe nazwano pierwszą postacią normalną.

**Pamiętajmy!**

**W modelu relacyjnym krotki składają się z atomowym dziedzin!**

## Podsumujmy:

1. Relacja jest zbiorem krotek.
2. Relacja może być reprezentowana poprzez tabelę, każdy wiersz tabeli to zapis krotki.
3. W tabeli każda dziedziną jest reprezentowana przez kolumnę (nazwa i typ).
4. Wartości danej dziedziny muszą być niezłożone (atomowe), czyli nie mogą być krotkami.
5. Każda krotka (wiersz) musi być unikalna (jak to zapewnić?)
6. Można utworzyć kilka relacji do reprezentowania rzeczywistości oraz powiązać (jak?) krotki (wiersze) poszczególnych relacji (tabel).

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

## **KROTKA 1:**

**(12,'Kowalski', 'Jan', ('Kowalski', 'Michał', 'syn', 2009-01-30), 1975-02-27).**

id.	nazwisko	imię	nazwisko	imię	pokrewieństwo	Data urodzenia	Data urodzenia
12	Kowalski	Jan	Kowalski	Michał	syn	2009-01-30	1975-02-27



**Powtarzające się grupy elementów -> krotka. Tworzymy nową relację do przechowywania krotek tego rodzaju!**

id.	nazwisko	imię	Id dziecka	Data urodzenia
12	Kowalski	Jan	1	1975-02-27

Id dziecka	nazwisko	imię	pokrewieństwo	Data urodzenia
1	Kowalski	Michał	syn	2009-01-30

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

id.	nazwisko	imię	Id dziecka	Data urodzenia
12	Kowalski	Jan	1	1975-02-27



**Co to za identyfikatory?**

Id dziecka	nazwisko	imię	pokrewieństwo	Data urodzenia
1	Kowalski	Mic hał	syn	2009-01-30

Model relacyjny według Codd zakładał, że każda krotka (wiersz) jest unikalna. Prowadzi to nas do drugiej części definicji modelu danych, czyli do warunków integralności w modelu relacyjnym!



## Plan wykładu

1. Struktura danych w modelu relacyjnym: krotki i relacje
2. Warunki integralności modelu relacyjnego

## Warunki integralności w modelu relacyjnym

- 1. Integralność encji** – Każda encja, reprezentowana w modelu relacyjnym przez krotkę (wiersz) jest unikalna. Unikalność krotki w relacji możemy osiągnąć poprzez zastosowaniem pojedynczej dziedzin (attributu, kolumny) lub zbioru dziedzin, których wartości są niepowtarzalne.

Często dla danej grupy encji (bytów) można użyć wiele takich dziedzin (attributów). Przykładowo dla grupy encji STUDENT (liczba pojedyncza w nazwie, ponieważ STUDENT jest abstrakcyjnym typem dla każdego, konkretnego studenta) możemy rozpatrzyć następujące atrybuty:

- Nr indeksu,
- Nr dowodu,
- PESEL, ...

**Atrybuty takie (dziedziny) określane są mianem KLUCZY!**

Spośród kluczy (kluczy kandydujących) wybieramy jeden **klucz główny (PRIMARY KEY)**, którego wartości będą unikalnymi identyfikatorami (niekoniecznie liczby!) krotek (wierszy).

**Każda krotka jest unikalna, ponieważ wartość klucza głównego jest unikalna!**

Czasami nie można w sposób naturalny zdefiniować pojedynczej dziedziny jako klucza. Przykładowo rozpatrzmy tabelę przechowującą informacje o małżeństwach. Jeśli przyjmiemy, że małżeństwo jest związkiem dwóch osób (które zawierają między sobą związek małżeński tylko raz) wówczas kluczem będzie **klucz złożony: (id\_kobiety\_żona, id\_mężczyzny\_mąż)**.

Identyfikatorami osób mogą być np. numery PESEL.

Zawsze można wprowadzić atrybut dodatkowy (sztuczny, tzn. nie wynika bezpośrednio z obserwacji świata i danej grupy encji), który będzie miał charakter identyfikatora czy liczby porządkowej (np. id\_małżeństwa, albo id\_księgi\_małżeństw). Z tych samych powodów wprowadzono najpierw imiona, nazwiska, a później numery PESEL, itp.

# Podstawy modelu relacyjnego



Przedmiot: *Bazy danych*

Politechnika Gdańska, *Inżynieria Biomedyczna*

W celu zilustrowania problemu posłużymy się językiem, który został opracowany przez firmę IBM specjalnie do zarządzania relacyjnymi bazami danych. Językiem tym jest **Structure Query Language (SQL)**. Stosuje on wyrażenia podobne do wydawania poleceń w języku angielskim, np.:

"Utwórz tabelę o nazwie studenci" -> **CREATE TABLE studenci**, itp.

Jednocześnie norma SQL definiuje typy danych atrybutów (dziedzin) oraz szereg własności modelu danych i sposobów ich implementacji za pomocą poleceń (wyrażeń) i oznaczeń. Rozkazy definiowane w SQL są interpretowane przez interfejsy systemów zarządzania bazami danych. Wykonane wyrażenie tworzy bazy danych, tabele, dodaje dane do tabel, zmienia dane, usuwa dane, nadaje uprawnienia, odczytuje dane zgodnie z kryteriami, itd. Język SQL będzie szczegółowo omówiony później.

Teraz wykorzystamy go w celu pokazania możliwości definiowania schematy relacji (budowy tabeli) oraz obsługi warunku integralności (klucze).

# Podstawy modelu relacyjnego



Przedmiot: *Bazy danych*

Politechnika Gdańska, *Inżynieria Biomedyczna*

Założmy, że utworzona jest baza danych. W bazie chcemy zdefiniować relację (tabelę). W języku SQL wydamy wówczas polecenie CREATE TABLE podając odpowiednie parametry. Składnia polecenia CREATE TABLE posiada standardowe ramy (które są czasem różnie realizowane przez produkty systemów baz danych).

Uproszczona forma składni polecenia:

**CREATE TABLE <name>**  
**( <columnDefinition> [, ...] [, <constraintDefinition>...] );**

Nazwa tabeli

**<columnDefinition>:**

**columnname Datatype [[NOT] NULL] [IDENTITY] [PRIMARY KEY]**

Definicja atrybutu. Po przecinku kolejne...

**<constraintDefinition>:**

**[CONSTRAINT <name>] UNIQUE ( <column> [,<column>...] ) | PRIMARY KEY ( <column> [,<column>...] )**

Definicja dodatkowych wymagań dla tabeli

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

Przykład dla krotki pracownika:

id.	nazwisko	imię	Id dziecka	Data urodzenia
12	Kowalski	Jan	1	1975-02-27

```
CREATE TABLE pracownik (  
  id int,  
  nazwisko char(45),  
  imie char(25),  
  id_dziecka int,  
  data_urodzenia date,  
  PRIMARY KEY(id)  
)
```

LUB

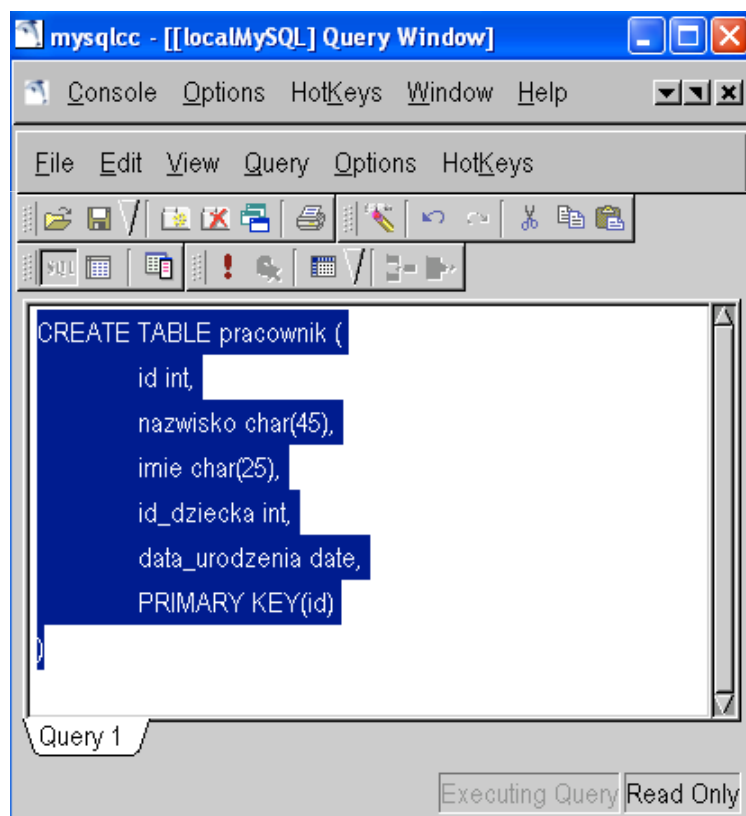
```
CREATE TABLE pracownik (  
  id int PRIMARY KEY,  
  nazwisko char(45),  
  imie char(25),  
  id_dziecka int,  
  data_urodzenia date  
)
```

KLUCZ GŁÓWNY

DEMO ->

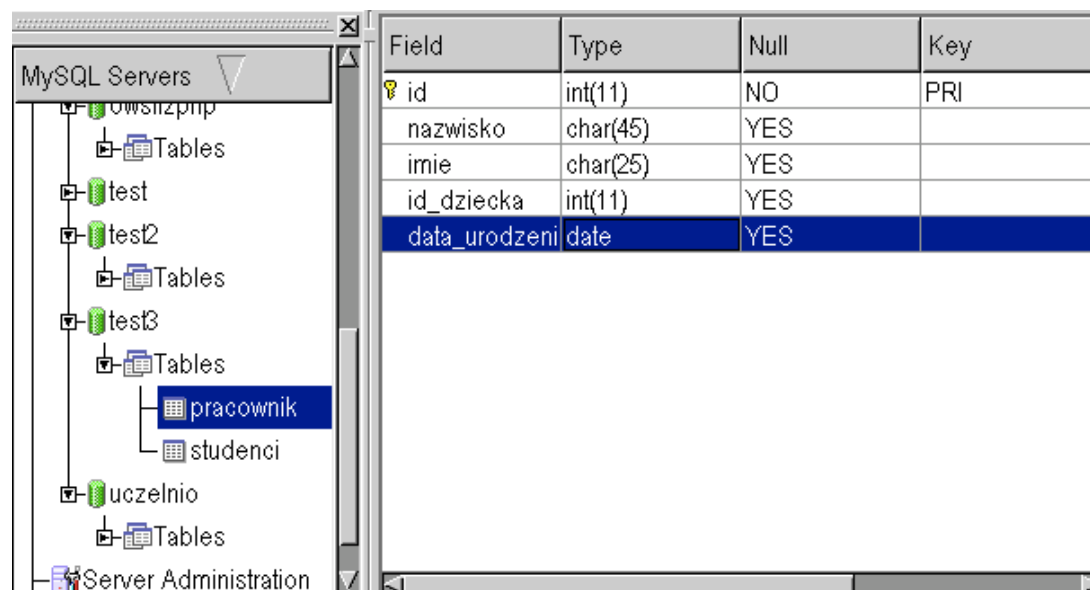
## DEMO 6

# Tworzenie tabel: CREATE TABLE



```

CREATE TABLE pracownik (
  id int,
  nazwisko char(45),
  imie char(25),
  id_dziecka int,
  data_urodzenia date,
  PRIMARY KEY(id)
)
    
```



Field	Type	Null	Key
id	int(11)	NO	PRI
nazwisko	char(45)	YES	
imie	char(25)	YES	
id_dziecka	int(11)	YES	
data_urodzeni	date	YES	

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

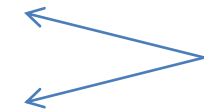
W celu przeprowadzenia kolejnych przykładów wprowadźmy jeszcze dwa, uproszczone wyrażenia języka SQL:

-Dodanie danych do tabeli:

***INSERT INTO tablename [( column [,...]] VALUES ( expression [,...])***

Przykładowo:

***INSERT INTO pracownik (id, nazwisko, imie, id\_dziecka, data\_urodzenia)***  
***VALUES (12, 'Kowalski', 'Jan', 1, '1975-02-27');***



KROTKA

***INSERT INTO pracownik VALUES (12, 'Kowalski', 'Jan', 1, '1975-02-27');***



# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

-Pobranie i wyświetlenie danych z tabeli:

***SELECT [\* | columnname [,...]] FROM tablename [ WHERE condition ]***

Przykładowo:

***SELECT \* FROM pracownik;***

***Wynik:***

	id	nazwisko	imie	id_dziecka	data_urodzenia
1	12	Kowalski	Jan	1	1975-02-27

← KROTKA

Sprawdźmy teraz czym jest warunek integralności encji.  
Każdy wiersz jest unikalny; wartość klucza głównego jest unikalna i musi być określona (nie może mieć wartości NULL – nic).

Wykonajmy polecenia:

## KROTKI

```
INSERT INTO pracownik VALUES (1, 'Nowak', 'Jerzy', 2, '1945-01-27');
```



```
INSERT INTO pracownik VALUES (2, 'Rumiński', 'Jacek', 3, '1980-03-30');
```



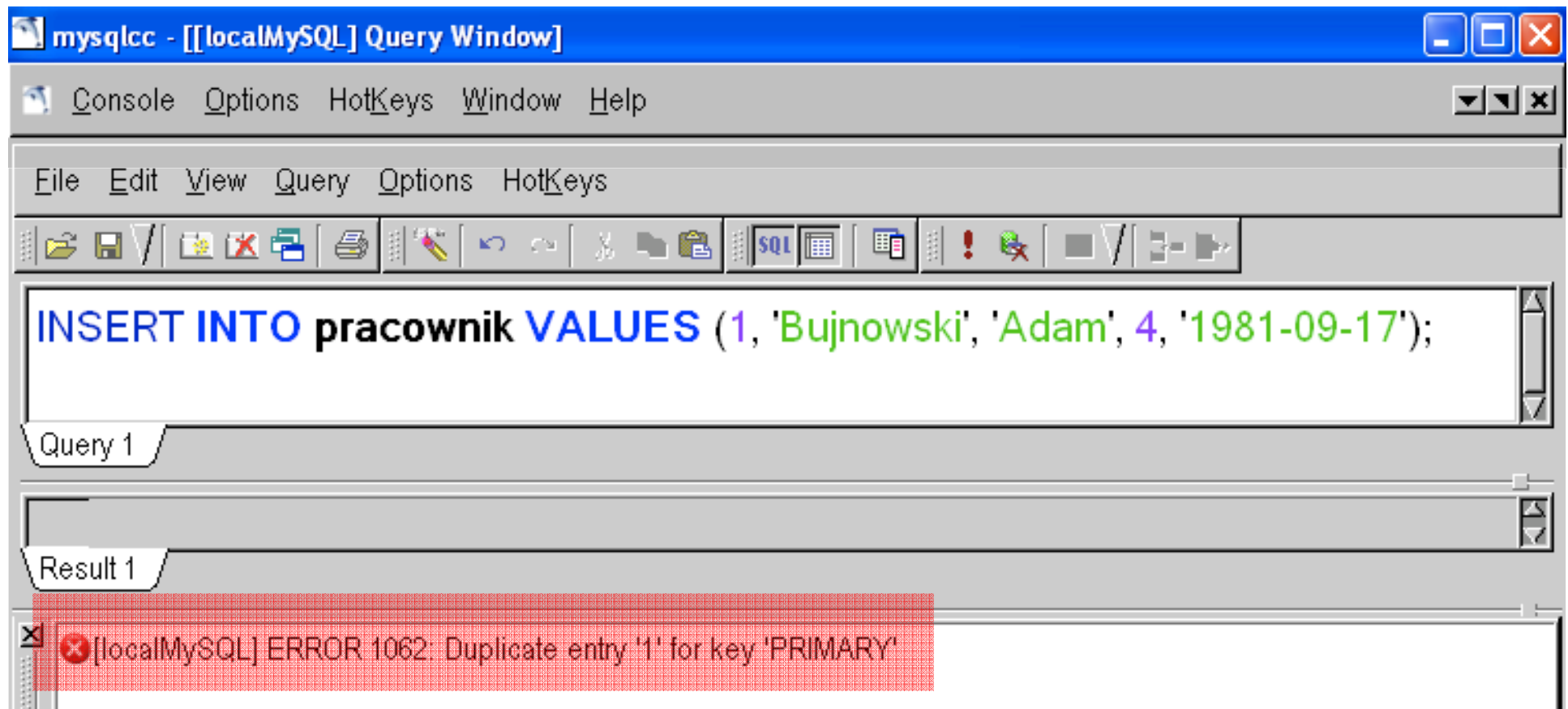
```
INSERT INTO pracownik VALUES (1, 'Bujnowski', 'Adam', 4, '1981-09-17');
```



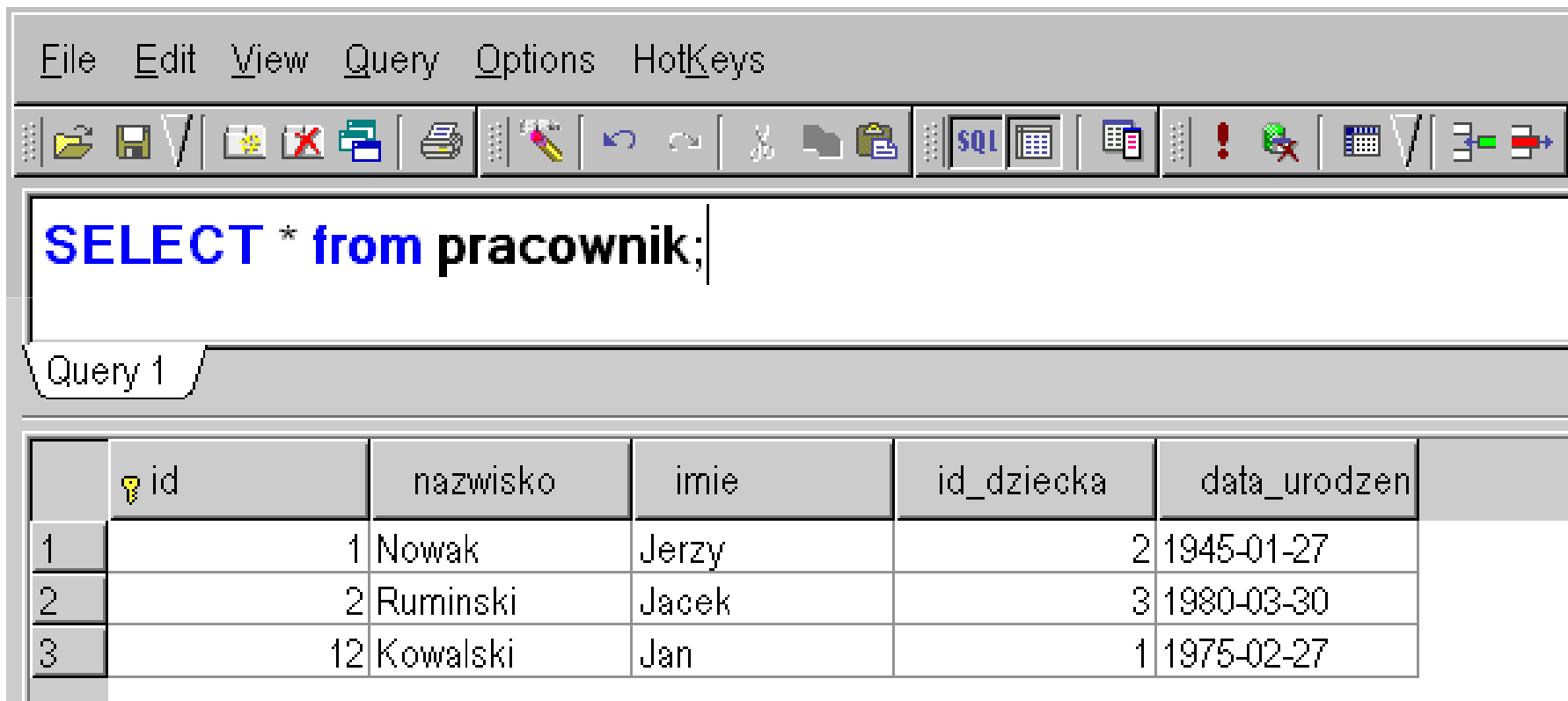
Wartość klucza głównego (PRIMARY KEY) została już użyta w relacji (tabeli).  
Krotka nie jest unikalna!

# DEMO 7

## Sprawdzanie integralności encji



## Przykład – zastosowanie wyrażenia SELECT:



The screenshot shows a database query tool interface. The menu bar includes File, Edit, View, Query, Options, and HotKeys. The toolbar contains various icons for file operations, editing, and querying. The query editor displays the SQL statement: **SELECT \* from pracownik;**. Below the query editor, a tab labeled "Query 1" is visible. The results are displayed in a table with the following columns: id (marked with a primary key icon), nazwisko, imie, id\_dziecka, and data\_urodzen.

	id	nazwisko	imie	id_dziecka	data_urodzen
1	1	Nowak	Jerzy	2	1945-01-27
2	2	Ruminski	Jacek	3	1980-03-30
3	12	Kowalski	Jan	1	1975-02-27

Użyta postać wyrażenia SELECT zwraca kopię relacji (tabeli)

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

Wyrażenie CREATE TABLE umożliwia dodatkowo nałożenie warunku na atrybut, aby jego wartości były unikalne (klucz). W tym celu stosuje słowo kluczowe UNIQUE, np.

```
CREATE TABLE student (  
  nr_indeksu int PRIMARY KEY,  
  pesel char(11) UNIQUE,  
  nazwisko char(45),  
  imie char(25)  
);
```

Każdy atrybut ma przypisany określony typ danych. W normie SQL zdefiniowano szereg typów danych. Podstawowe z nich to:

**int** – liczby całkowite,

**char(N)** – ciąg N znaków o stałej długości (rezerwuje się N bajtów)

**varchar(N)** – ciąg do N znaków (zajmuje tyle bajtów ile podana wartość + 1 bajt na określenie rozmiaru bądź końca danego ciągu znaków) (CHARACTER VARYING)

# Podstawy modelu relacyjnego



Przedmiot: *Bazy danych*

Politechnika Gdańska, *Inżynieria Biomedyczna*

**float, real, double precision** – liczby rzeczywiste, zmiennoprzecinkowe,

**date** – data

**time** – czas (godzina, minuty, sekundy,...),

**timestamp** – stempel czasowy (data+czas),

**bit** – 0 lub 1,

**boolean** – typ logiczny true lub false,

**BLOB** – Binary Large Object – duży obiekt binarny – sekwencja bajtów (np. 1GB),

**CLOB** – Character Large Object – duży obiekt tekstowy – sekwencja znaków (większa niż 255, zwykle maksymalny rozmiar dla varchar i char), oraz inne.

Nowe wersje normy SQL umożliwiają definiowanie własnych typów danych za pomocą polecenia **CREATE TYPE**. Taki typ może być prostą strukturą atrybutów (np. do przechowywania liczby zespolonej: część rzeczywista i część urojona) lub może tworzyć klasę na wzór modelu obiektowego (posiadać atrybuty i funkcje). Możliwość tworzenia własnych typów danych wykorzystano do zbudowania standardowych rozszerzeń do SQL wprowadzając multimedialne typy danych, np. StillImage, Polygon, itp.

Warunki integralności w modelu relacyjnym

**2. Integralność referencyjna** – Dziedzina w danej relacji (tabeli) R1 jest określana przez dziedzinę (zbiór użytych wartości) innej relacji (tabeli) R2 włącznie z wartością NULL (nic).

Jaki jest sens takiej integralności?

Rozpatrzmy przykład z tabelą "pracownik". Użyliśmy tam atrybutu "Id dziecka", aby wskazać powiązanie pomiędzy pracownikiem, a dzieckiem. Tylko jaką wartość możemy wpisać w dane pole? Na pewno wartość typu "int", ale czy każdą? Jeśli wpiszemy wartość '83', a takiej wartości nie ma w tabeli przechowującej krotki dzieci?

id.	nazwisko	imię	Id dziecka	Data urodzenia
12	Kowalski	Jan	1	1975-02-27

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

Oczywistym rozwiązaniem jest ograniczenie zakresu możliwych wartości (dziedziny) do tych wartości, które już zostały wykorzystane w tabeli "dziecko" do identyfikacji (klucz główny) dzieci.

Tworząc tabelę "pracownik" należałoby wskazać, że dany atrybut (kolumna) może mieć tylko takie wartości, które już wystąpiły w kolumnie innej, wskazywanej tabeli ("dziecko").

Id dziecka	nazwisko	imię	pokrewieństwo	Data urodzenia
1	Kowalski	Michał	syn	2009-01-30

id.	nazwisko	imię	Id dziecka	Data urodzenia
12	Kowalski	Jan	1	1975-02-27



Warunek wymuszający możliwe wartości atrybutu (kolumny) w relacji R1 w odniesieniu do atrybutu (kolumny) innej relacji R2 nazywany jest kluczem obcym (FOREIGN KEY).

Zanim przedstawimy praktyczną realizację klucza obcego w SQL rozpatrzmy pewien problem.

Co będzie jeśli pracownik będzie miał kolejne dziecko? Niestety trzeba będzie zmienić strukturę tabeli "pracownik". Tak samo w przypadku coraz większej liczby dzieci. Oznacza to, że naturalny proces opisu encji (krotki):

**(12,'Kowalski', 'Jan', ('Kowalski', 'Michał', 'syn', 2009-01-30), ('Kowalska', 'Anna', 'córka', 2009-11-30), 1975-02-27).**

czy:

**(12,'Kowalski', 'Jan', 1, 2, 1975-02-27).**

trzeba będzie zmienić do innej postaci!

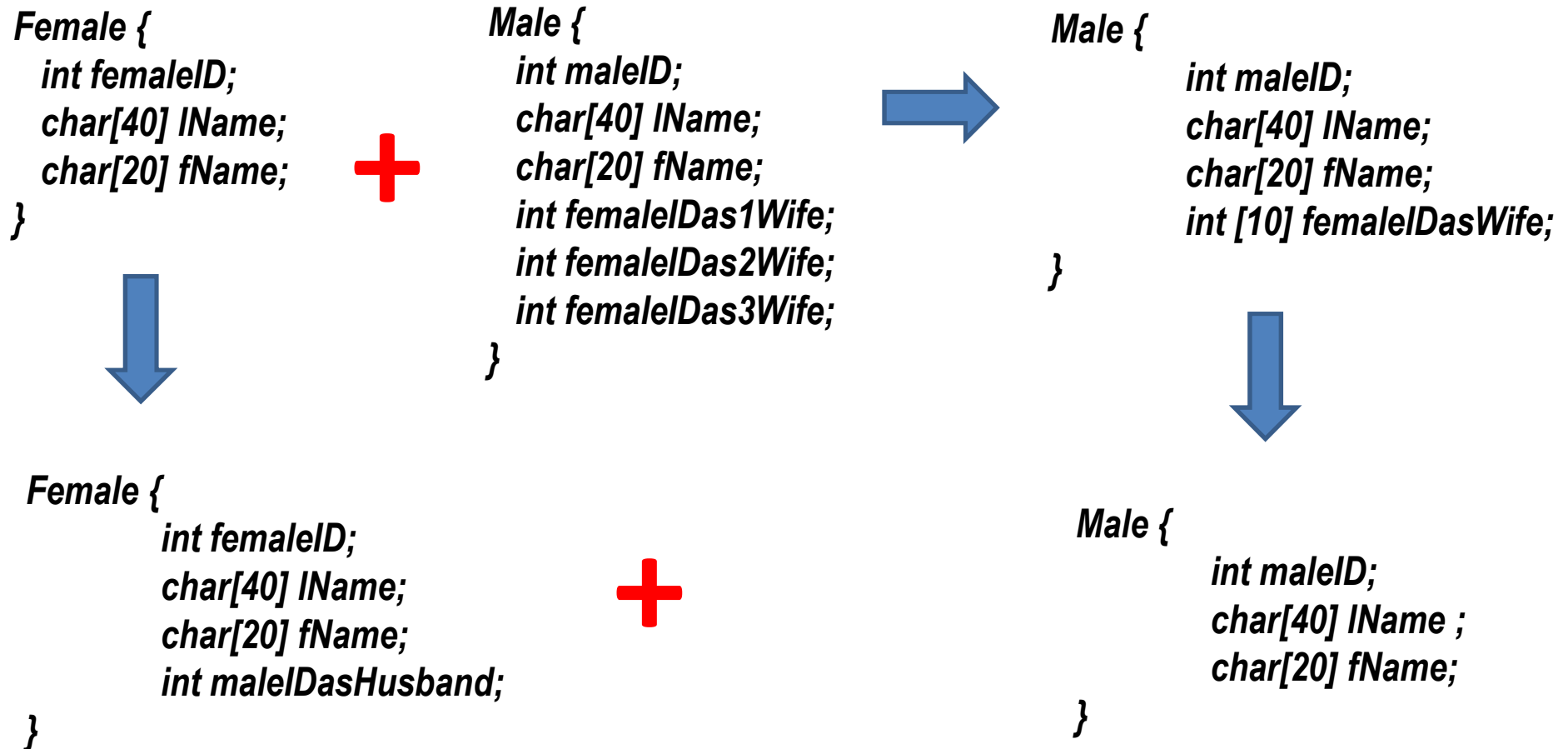
# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

Wymagana zmiana jest analogiczna do wcześniej omówionego przykładu (mężczyzna ma wiele żon):



# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

Oznacza to, że w przypadku przykładu z tabelami "pracownik" i "dziecko" musimy zmienić ich schematy do:

id.	nazwisko	imię	Data urodzenia
12	Kowalski	Jan	1975-02-27

Id dziecka	nazwisko	imię	pokrewieństwo	Data urodzenia	Id_pracownika
1	Kowalski	Michał	syn	2009-01-30	12

W ten sposób można zrealizować związek jeden do wielu (jeden pracownika ma wiele dzieci), bez względu na to, ile ich będzie miał.

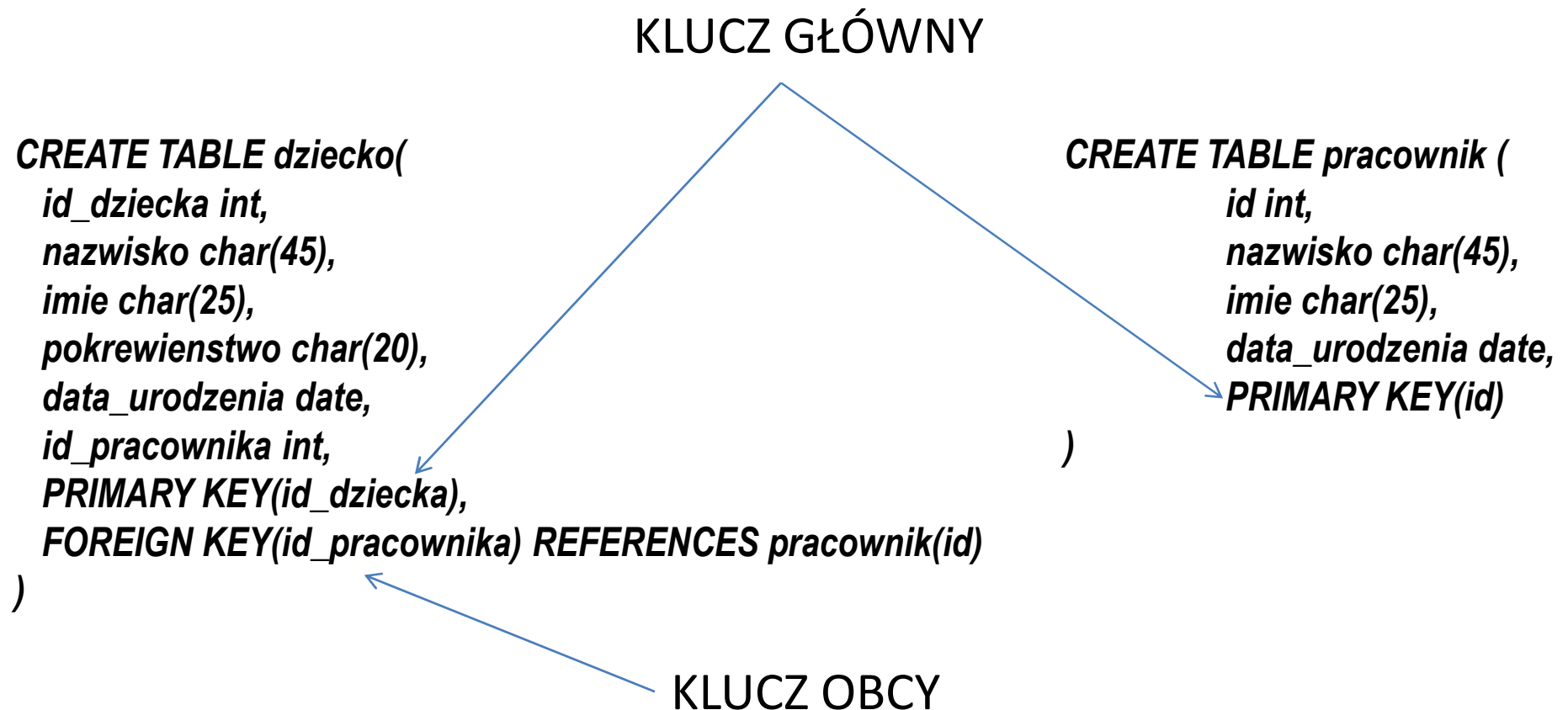
Zapiszmy teraz wyrażenia CREATE TABLE dla obu tabel z uwzględnieniem integralności referencyjnej (stosując klucz obcy).

# Podstawy modelu relacyjnego



Przedmiot: *Bazy danych*

Politechnika Gdańska, *Inżynieria Biomedyczna*



Wykonując wyrażenia CREATE należy najpierw utworzyć tabelę "pracownik" dopiero później tabelę "dziecko" (ta wskazuje powiem na tabelę pracownik).

# Podstawy modelu relacyjnego



Przedmiot: *Bazy danych*

Politechnika Gdańska, *Inżynieria Biomedyczna*

Wykonanie wyrażeń CREATE w tej samej bazie może spowodować błąd, ponieważ w bazie istnieje już tabela "pracownik" zdefiniowana według wcześniejszego schematu. W celu usunięcia tabeli wykonujemy polecenie DROP język SQL:

***DROP TABLE pracownik;***

Następnie wykonujemy polecenia CREATE tworząc tabele "pracownik" i "dziecko". Teraz możemy dodać krotki (wiersze sprawdzając warunek integralności referencyjnej).

***INSERT INTO pracownik VALUES (12, 'Kowalski', 'Jan', '1975-02-27');***

***INSERT INTO pracownik VALUES (2, 'Ruminski', 'Jacek', '1980-03-30');***

***INSERT INTO dziecko VALUES (1, 'Kowalski', 'Michal', 'syn', '2009-01-30', 12);***

***INSERT INTO dziecko VALUES (2, 'Kowalska', 'Anna', 'corka', '2009-11-30', 12);***

***INSERT INTO dziecko VALUES (3, 'Ruminska', 'Aleksandra', 'corka', '2002-02-02', 2);***

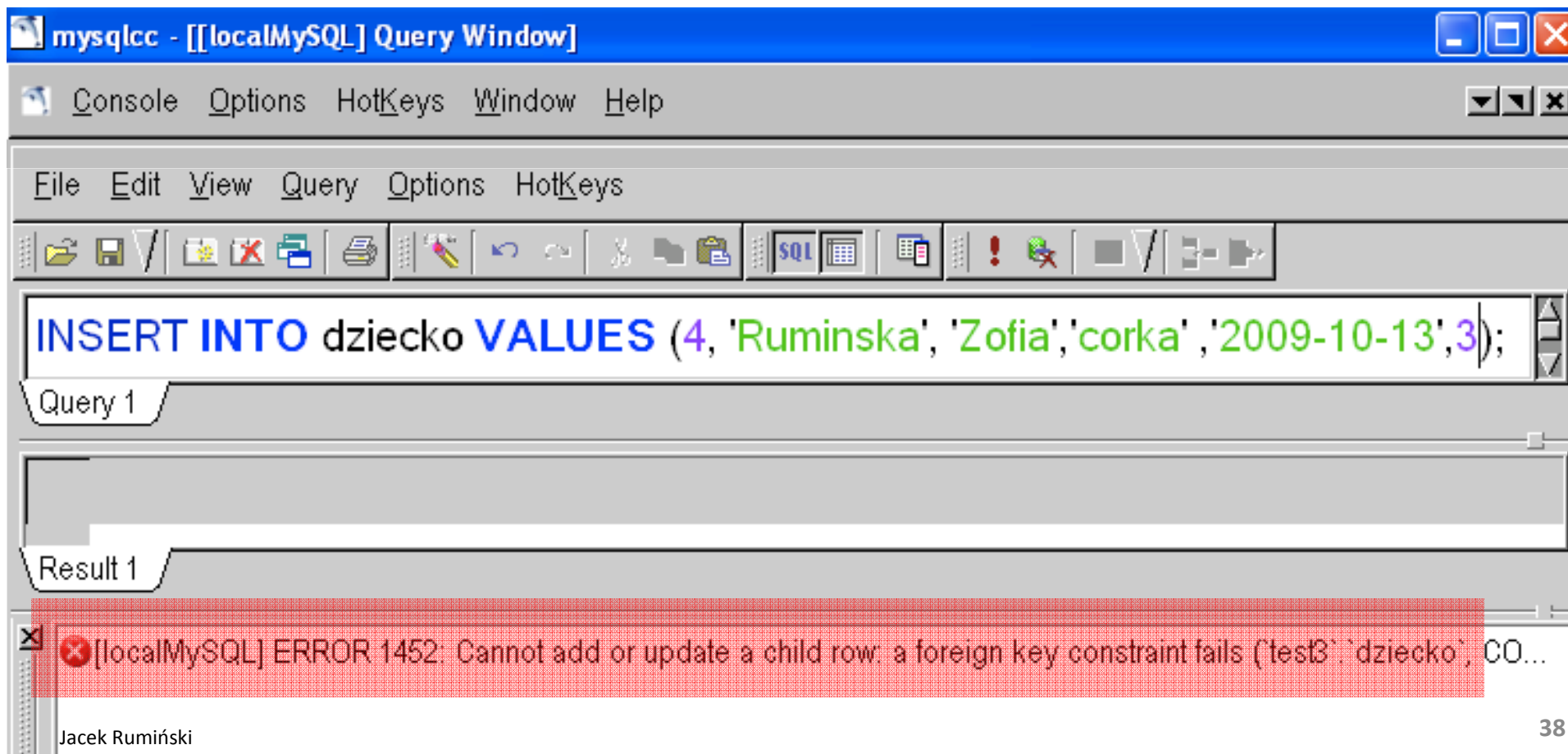
***INSERT INTO dziecko VALUES (4, 'Ruminska', 'Zofia', 'corka', '2009-10-13', 3);***



**Nie ma takiego pracownika!**

# DEMO 8

## Sprawdzanie integralności referencyjnej



The screenshot shows the 'mysqlcc - [[localMySQL] Query Window' application. The menu bar includes 'Console', 'Options', 'HotKeys', 'Window', and 'Help'. The toolbar contains various icons for file operations, editing, and database management. The main text area displays the SQL query: `INSERT INTO dziecko VALUES (4, 'Ruminska', 'Zofia', 'corka', '2009-10-13', 3);`. Below the query, there is a 'Query 1' tab. The 'Result 1' tab is active, showing a red error message: `[localMySQL] ERROR 1452: Cannot add or update a child row: a foreign key constraint fails ('test3`.`dziecko`, CO...`. The error message is partially obscured by a red rectangular box.

# Podstawy modelu relacyjnego



Przedmiot: **Bazy danych**

Politechnika Gdańska, **Inżynieria Biomedyczna**

Udało nam się zbudować silne powiązanie pomiędzy krotkami różnych relacji (wierszami różnych tabel). Co będzie jednak w przypadku, gdy z tabeli nadrzędnej (u nas "pracownik") będę chciał usunąć krotkę? Co się stanie z krotkami z tabeli potomnej (u nas "dziecko"), które wskazywały usuniętą krotkę?

Możliwości jest kilka. W języku SQL definiując klucz obcy można (często zależy od oprogramowania realizującego SQL) wyróżnić następujące warunki nałożone na klucz obcy:

**[CONSTRAINT *[symbol]*] FOREIGN KEY**

**[*index\_name*] (*index\_col\_name*, ...)**

**REFERENCES *tbl\_name* (*index\_col\_name*,...)**

**[ON DELETE *reference\_option*]**

**[ON UPDATE *reference\_option*]**

Co zrobić, gdy usuwamy wiersz

Co zrobić, gdy zmieniamy wiersz

***reference\_option*:**

**RESTRICT / CASCADE / SET NULL / NO ACTION**

Warunki / rodzaj działania

**RESTRICT** – nie wykonuj operacji usunięcia lub aktualizacji wiersza w tabeli nadrzędnej (próba usunięcia będzie odrzucona)

**NO ACTION** - nie wykonuj operacji usunięcia lub aktualizacji wiersza w tabeli nadrzędnej jeśli w tabeli podrzędnej są wiersze wskazujące na usuwany wiersz (próba usunięcia będzie odrzucona)

**CASCADE** – kaskadowo usuń/aktualizuj wiersz z tabeli nadrzędnej oraz wiersze tabeli podrzędnej wskazujące przez klucz obcy usuwany wiersz tabeli nadrzędnej

**SET NULL** – usuń/aktualizuj wiersz w tabeli nadrzędnej. W tabeli podrzędnej ustaw wartości klucza obcego na NULL (operacja możliwa tylko wtedy, gdy klucz obcy nie zawierał ograniczenia NOT NULL)

Jeśli nic nie podamy (tak jak w poprzednich slajdach) wówczas wykonywana jest akcja domyślna (według ustawień oprogramowania).



W celu ilustracji wykorzystania przedstawionych rodzajów działań wprowadźmy kolejne polecenie języka SQL:

usuwanie wierszy z tabeli:

***DELETE FROM tablename [WHERE condition]***

Jeśli nie podamy opcjonalnego warunku, wówczas polecenie żąda usunięcia wszystkich wierszy tabeli.

Założmy, że w naszym przykładzie chcemy usunąć krotkę z tabeli "pracownik" o wartości atrybutu **id** = 2:

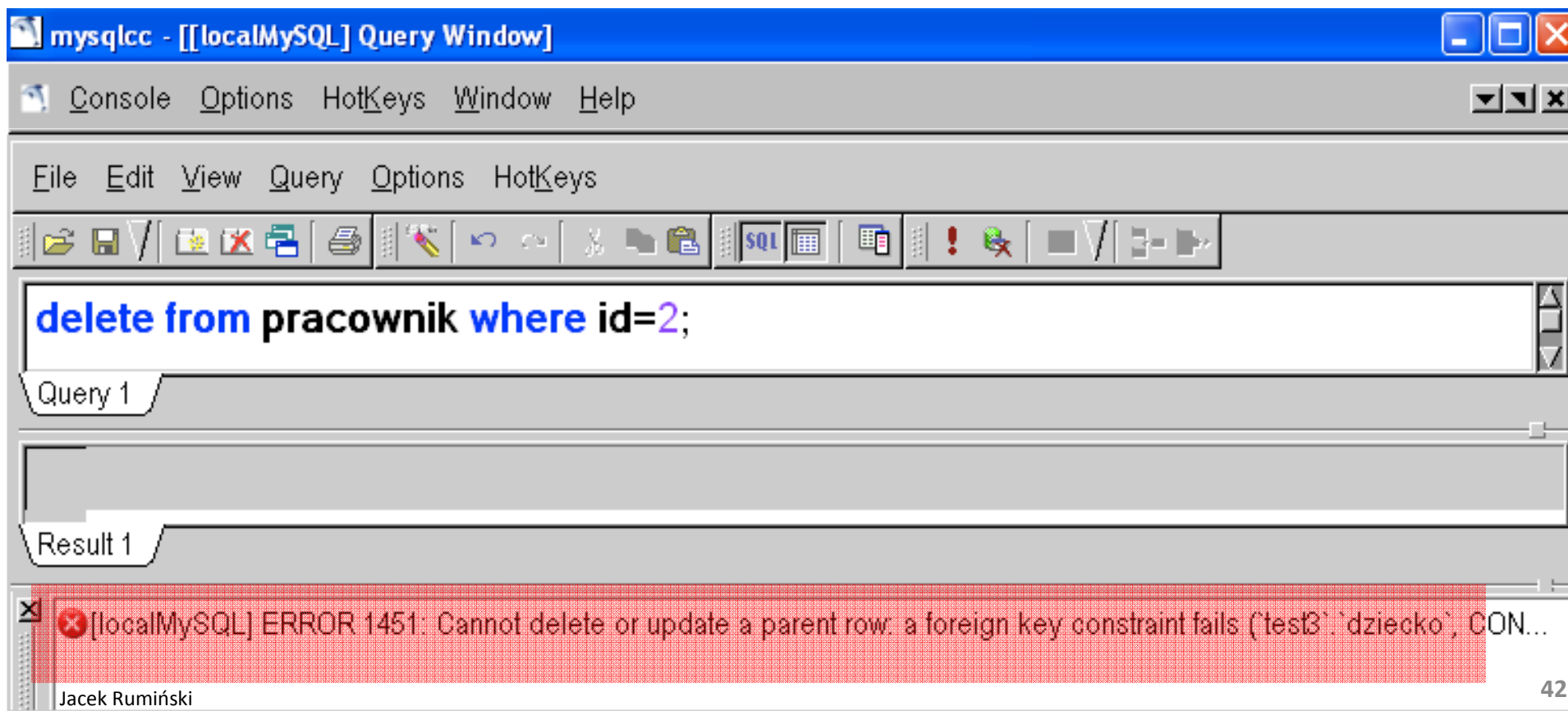
***DELETE FROM pracownik WHERE id=2;***

Wykonanie tego polecenia nie powiedzie się, ponieważ w stosowanym oprogramowaniu (MySQL, model tabeli InnoDB) domyślne ustawienie warunku to RESTRICT/NO ACTION (w MySQL to ten sam warunek).

DEMO->

# DEMO 9

## Sprawdzanie integralności referencyjnej



The screenshot shows a MySQL command-line interface window titled "mysqlcc - [[localMySQL] Query Window". The window has a menu bar with "Console", "Options", "HotKeys", "Window", and "Help". Below the menu bar is a toolbar with various icons for file operations, editing, and execution. The main text area contains the SQL query: `delete from pracownik where id=2;`. Below the query area, there is a tab labeled "Query 1". The results area, labeled "Result 1", is empty. At the bottom of the window, a red error message is displayed: `[localMySQL] ERROR 1451: Cannot delete or update a parent row: a foreign key constraint fails ('test3`.`dziecko`, CON...`. The name "Jacek Rumiński" is visible in the bottom left corner of the window.

Posługując się kluczami obcymi można zbudować szereg związków pomiędzy grupami encji.

Poznaliśmy już możliwość zbudowania związku jeden-do-wielu i wiele-do-jednego.

Jak wymóc związek jeden-do-jednego?

Najprostsza realizacja to nałożenie na klucz obcy dodatkowego warunku typu UNIQUE lub PRIMARY KEY. Wówczas wartość klucza obcego może tylko raz wystąpić w relacji (tabeli), czyli otrzymamy związek jeden-do-jednego.

Przykład ->

# Podstawy modelu relacyjnego



Przedmiot: *Bazy danych*

Politechnika Gdańska, *Inżynieria Biomedyczna*

## Przykład

```
CREATE TABLE dziecko(  
  id_dziecka int,  
  nazwisko char(45),  
  imie char(25),  
  pokrewienstwo char(20),  
  data_urodzenia date,  
  id_pracownika int UNIQUE, ← Pojedyncze wystąpienie  
  PRIMARY KEY(id_dziecka),    – jeden-do-jednego )  
  FOREIGN KEY(id_pracownika) REFERENCES pracownik(id)  
)
```

KLUCZ OBCY

```
CREATE TABLE pracownik (  
  id int,  
  nazwisko char(45),  
  imie char(25),  
  data_urodzenia date,  
  PRIMARY KEY(id)
```

```
INSERT INTO pracownik VALUES (12, 'Kowalski', 'Jan', '1975-02-27');
```

Wartość klucza  
została już użyta!

```
INSERT INTO dziecko VALUES (1, 'Kowalski', 'Michał', 'syn', '2009-01-30', 12);  
INSERT INTO dziecko VALUES (2, 'Kowalska', 'Anna', 'córka', '2009-11-30', 12);
```



Poznaliśmy realizacje w modelu relacyjnym następujących związków

- jeden-do-wielu ,
- wiele-do-jednego,
- jeden-do-jednego.

Problemy i rozwiązania związane z wykorzystaniem w modelu relacyjnym związku wiele-do-wielu poznamy w czasie kolejnego wykładu.

## PODSUMUJMY:

- **Strukturą danych w modelu relacyjnym jest relacja (tabela),**
- **Relacja jest zbiorem krotek,**
- **Każda krotka jest unikalna, co zapewnia warunek integralności encji (klucz główny),**
- **Dziedzinę atrybutu można ograniczyć poprzez realizację integralności referencyjnej (klucz obcy).**

# Bazy danych – koniec wykładu 5



Przedmiot: *Bazy danych*

Politechnika Gdańska, *Inżynieria Biomedyczna*

## Co dalej?

Zapoznaliśmy się podstawami modelu relacyjnego. Omówiliśmy podstawy dotyczące struktury danych oraz warunków integralności. Na kolejnym wykładzie zajmiemy się projektowaniem baz danych, po czym omówimy operacje na danych. W ten sposób poznamy podstawy relacyjnego modelu danych (struktura+warunki+operacje).

# ZAPRASZAMY NA WYKŁAD 6