# Nierelacyjne bazy danych

Aleksander Wójcik
informatyka III rok
spec. inżynieria oprogramowania

WSEI

19 października 2014

# Plan prezentacji

1. Wstęp
2. Model relacyjny
   - Rozwój historyczny
   - Opis modelu relacyjnego
   - Przykład w relacyjnym modelu danych
3. Obiektowe bazy danych
   - Założenia obiektowego modelu danych
   - Przykład w obiektowym modelu danych
   - Porównanie dwóch modeli na przykładzie
   - Przykładowe obiektowe bazy danych
4. NoSQL
   - Założenia technologii
   - Ruch NoSQL
   - Typy baz danych NoSQL
   - Przykład modyfikacji dokumentowej bazy danych

## Beamer - biblioteka LaTeX-a

Prezentacja składa się z ok. 650 linijek kodu.

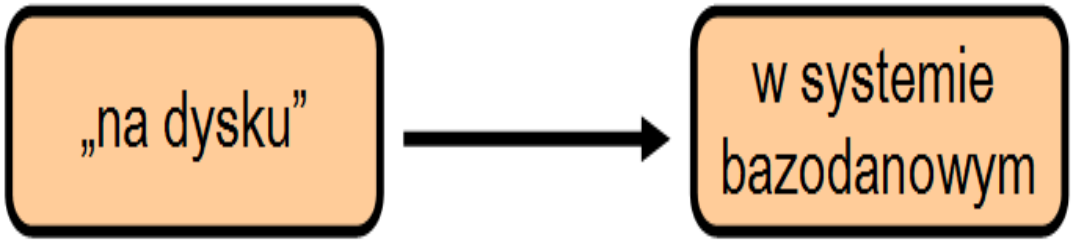
```
\documentclass[brown]{beamer}
%beamerowe
  \usetheme{Madrid}
  \usepackage{polski}
  \usepackage[cp1250]{inputenc}
  \usepackage{graphicx}
%latexowe
\usepackage{listings}
\lstset{
language=SQL,
basicstyle=\tiny\sffamily,
showstringspaces=false,
keywordstyle=\color{blue}
}
\begin{document}
\title{Nierelacyjne bazy danych}
\author{Aleksander Wójcik\\ \\
student II roku informatyki\\
spec. inżynieria oprogramowania}
\institute{WSEI}
\date{\today}
\section{Wstęp}
%\begin{frame}
%\titlepage
%
%\end{frame}
```

```
        \frametitle{Beamer}

        \begin{block}{Beamer – biblioteka \LaTe
Prezentacja składa się z ok. 650 linijek kodu.
        \end{block}
\begin{columns}
                \begin{column}{.5\textwidth}
\lstinputlisting[language=TeX, firstline=1, las
                \end{column}
                \begin{column}{.5\textwidth}
\lstinputlisting[language=TeX, firstline=32, la
                \end{column}
        \end{columns}
\end{frame}
%#############################################
%-------------------------------------------
\section{Model relacyjny}
%-------------------------------------------
\subsection{Rozwój historyczny}
\begin{frame}
        \frametitle{Jak zapisywano dane?}

        \begin{block}{Przyczyny przejścia na sy
                \begin{enumerate}
                        \item fizyczna i logicz
\item centralna kontrola integralności
```

# Jak zapisywano dane?

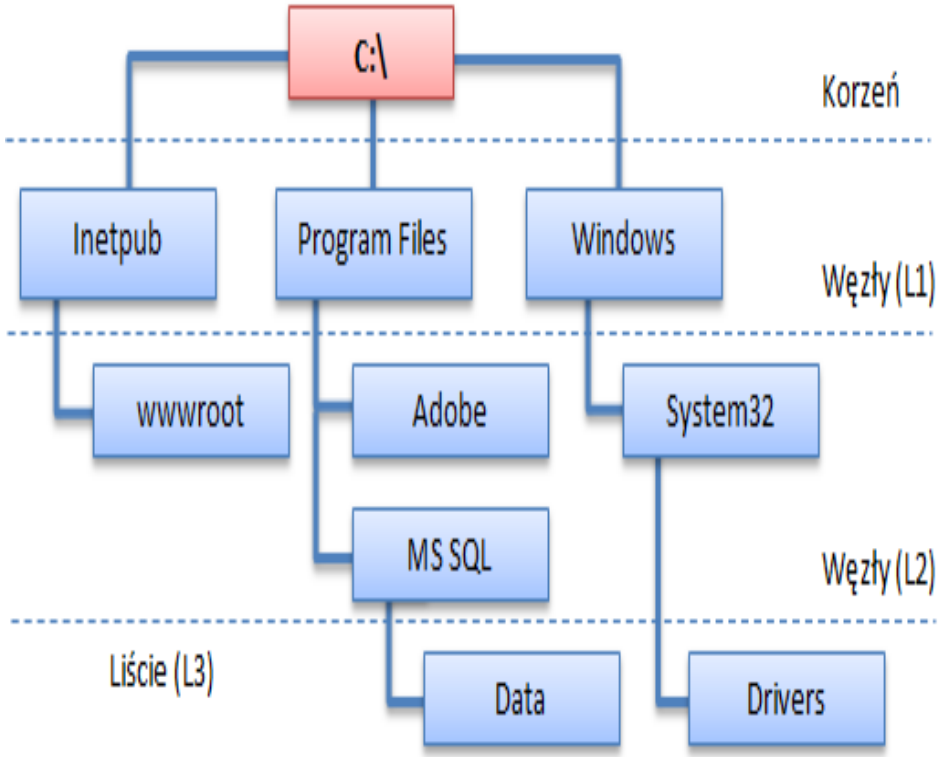
## Przyczyny przejścia na systemy bazodanowe

1. fizyczna i logiczna niezależność danych
2. centralna kontrola integralności
3. języki na wysokim poziomie abstrakcji



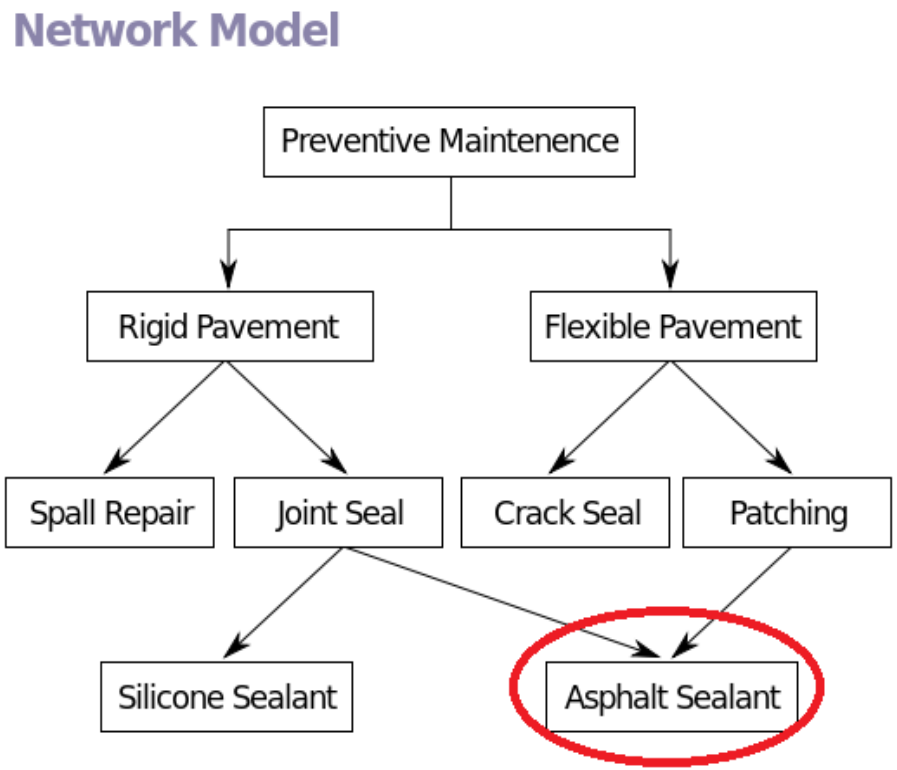

Rysunek: Przejście na systemy bazodanowe

# Model hierarchiczny




Rysunek: Każdy węzeł ma jednego rodzica, baza danych ma formę drzewa

# Model sieciowy




Rysunek: Model sieciowy jest mniej restrykcyjny niż model hierarchiczny

# Model relacyjny

## Druga generacja

`Model relacyjny` oparty o teorię zbiorów powstał w 1970r. Jego twórcą był Edgar Frank Codd. Od 1973r. firmy tj. IBM i Oracle zaczęły wdrażać model relacyjny, który w krótkim czasie zawładnął rynkiem baz danych.


Rysunek: logo firmy Oracle


Rysunek: logo firmy IBM

# Plusy modelu relacyjnego

## Plusy

1. Prostota - użytkownik szybko orientuje się jak baza jest zbudowana
2. Oparcie modelu na solidnych podstawach teoretycznych (matematycznych)
3. Niezależność danych od programu
4. Stabilna pozycja na rynku
5. Intuicyjny interfejs dostępu do bazy: polecenia SELECT, UPDATE, DELETE

# Minusy modelu relacyjnego

## Minusy

1. **Trudność w modelowaniu rzeczywistości**
   1. Brak typów złożonych
   2. Powiązania pomiędzy tabelami tylko poprzez klucze obce
   3. Nieelastyczność modelu, brak możliwości rozszerzenia

# Minusy modelu relacyjnego(2)

## Minusy

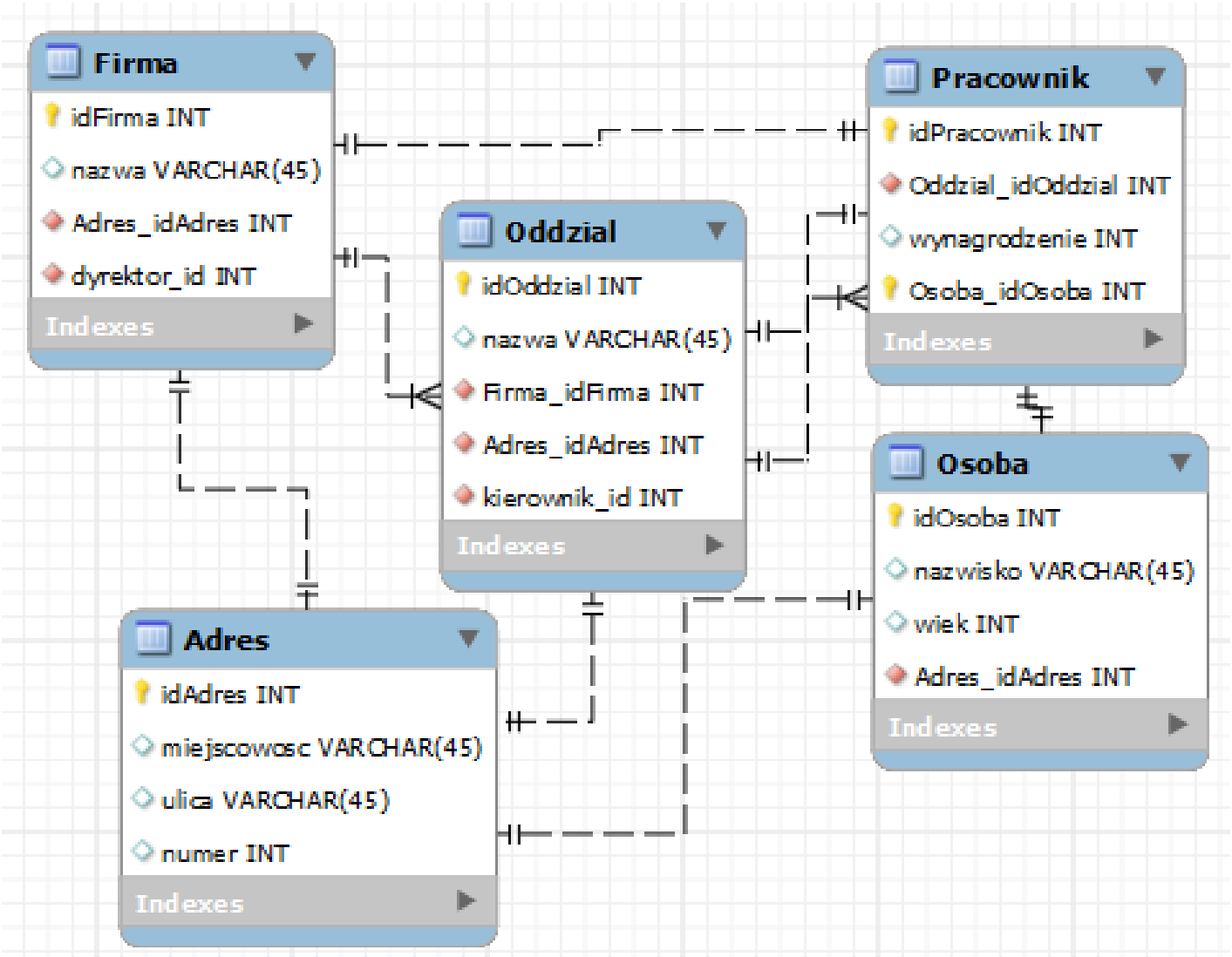
1. Potrzeba definiowania klucza sztucznego, gdy atrybuty nie są wystarczające do uzyskania unikatowej identyfikacji (np.nazwa firmy może się powtarzać)
2. Atrybuty złożone (np. adres) nie mogą być reprezentowane bezpośrednio - składowe muszą być deklarowane indywidualnie jako atrybuty
3. Atrybuty zbiorowe (np. pracownicy) muszą być rozróżnialne od atrybutów jednowartościowych i reprezentowane w innym schemacie relacyjnym
4. Agregacje i specjalizacje nie są w łatwy sposób obsługiwane i wymagają specjalnych więzów integralności

# Przykład

## Założenia

1. **firma** ma nazwę, centralę, kilka oddziałów i dyrektora
2. **oddziały** mają nazwę, biuro, kierownika i pracowników
3. **osoby** mają nazwisko, wiek i miejsce zamieszkania
4. **pracownicy** są osobami posiadającymi wynagrodzenie

# Schemat firmy (**relacyjny**)




Rysunek: Model bazy danych utworzony w programie Workbench

# Kod SQL tworzący tabele

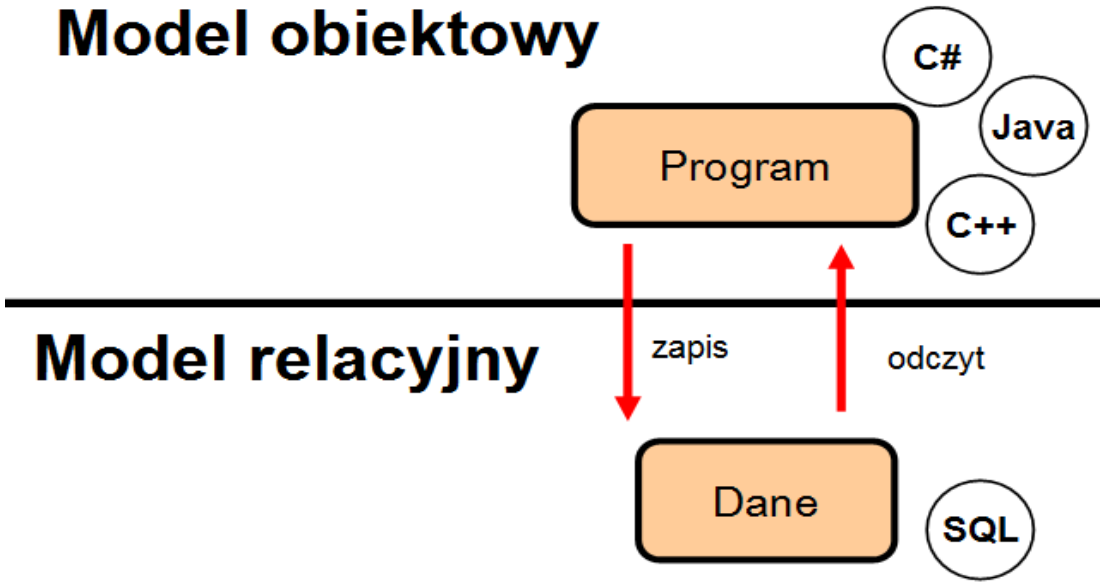
```
CREATE TABLE `Pracownik` (
  `idPracownik` INT NOT NULL,
  `Oddzial_idOddzial` INT NOT NULL,
  `wynagrodzenie` INT NULL,
  `Osoba_idOsoba` INT NOT NULL,
  PRIMARY KEY (`idPracownik`),
  CONSTRAINT `fk_Pracownik_Oddzial1`
    FOREIGN KEY (`Oddzial_idOddzial`)
    REFERENCES `Oddzial` (`idOddzial`)
   CONSTRAINT `fk_Pracownik_Osoba1`
    FOREIGN KEY (`Osoba_idOsoba`)
    REFERENCES `Osoba` (`idOsoba`)
    )

CREATE TABLE `Oddzial` (
  `idOddzial` INT NOT NULL,
  `Firma_idFirma` INT NOT NULL,
  `Adres_idAdres` INT NOT NULL,
  `kierownik_id` INT NOT NULL,
  PRIMARY KEY (`idOddzial`),
  CONSTRAINT `fk_Oddzial_Firma`
    FOREIGN KEY (`Firma_idFirma`)
    REFERENCES `Firma` (`idFirma`)
CONSTRAINT `fk_Oddzial_Adres1`
    FOREIGN KEY (`Adres_idAdres`)
    REFERENCES `Adres` (`idAdres`)
CONSTRAINT `fk_Oddzial_Pracownik1`
    FOREIGN KEY (`kierownik_id`)
    REFERENCES `Pracownik` (`idPracownik`)
)
```

```
CREATE TABLE `Adres` (
  `idAdres` INT NOT NULL,
  `miejscowosc` VARCHAR(45) NULL,
  `ulica` VARCHAR(45) NULL,
  `numer` INT NULL,
  PRIMARY KEY (`idAdres`)
)

CREATE TABLE `Osoba` (
  `idOsoba` INT NOT NULL,
  `nazwisko` VARCHAR(45) NULL,
  `wiek` INT NULL,
  `Adres_idAdres` INT NOT NULL,
  PRIMARY KEY (`idOsoba`),
  CONSTRAINT `fk_Osoba_Adres1`
    FOREIGN KEY (`Adres_idAdres`)
    REFERENCES `Adres` (`idAdres`)
)

CREATE TABLE `Firma` (
  `idFirma` INT NOT NULL,
  `nazwa` VARCHAR(45) NULL,
  `Adres_idAdres` INT NOT NULL,
  `dyrektor_id` INT NOT NULL,
  PRIMARY KEY (`idFirma`),
  CONSTRAINT `fk_Firma_Adres1`
    FOREIGN KEY (`Adres_idAdres`)
    REFERENCES `Adres` (`idAdres`)
    CONSTRAINT `fk_Firma_Pracownik1`
    FOREIGN KEY (`dyrektor_id`)
    REFERENCES `Pracownik` (`idPracownik`)
)
```

# Głowny problem: niekompatybilność

## Niekompatybilność dwóch modeli

- model przechowywania danych: **relacyjny**
- model implementacji programu: **obiektowy**




Rysunek: Pomiędzy dwoma "światami" musi istnieć komunikacja

# Trzy główne podejścia


Rysunek: Próby pogodzenia paradygmatów

## Próby pogodzenia paradygmatów

1. Zbliżenie języka proceduralnego SQL do języków obiektowych
2. Zbliżenie możliwości języków obiektowych do SQL
3. Mapowanie obiektowo-relacyjne

# Brak jednoznacznego paradygmatu

Zarówno standard SQL2 jak i założenia paradygmatu obiektowego są `jednoznaczne`.
Jednak przy próbie "połączenia" paradygmatów i wypracowania jednolitego standardu pojawiają się duże rozbieżności.

## SQL3

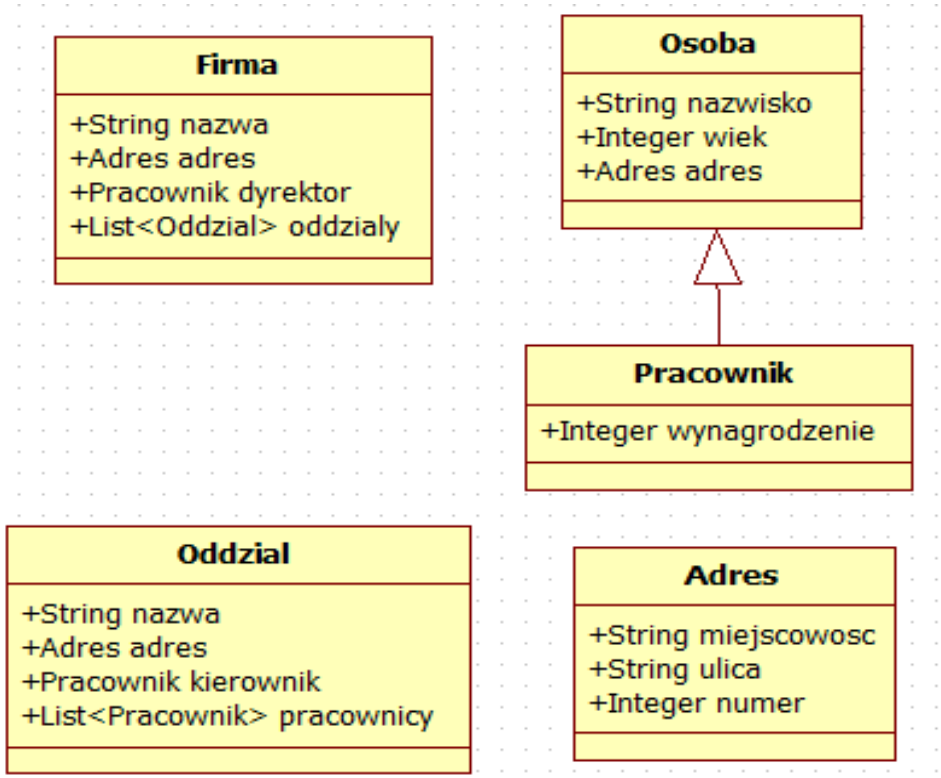
1. SQL2 rozszerzony o możliwości obiektowe
2. **nie spełnia** paradygmatu hermetyzacji:
   brak spefifykatorów:
   1. PUBLIC
   2. PRIVATE
   3. PROTECTED
3. ogromny standard przekraczający 1200 stron

# Przykładowe założenia

## Założenia modelu

1. Deklaracje typów są tak elastyczne jak w językach programowania
2. Obiekty mogą być identyfikowane unikatowo, niezależnie od ich wartości
3. Rozróżnianie typu krotkowego i zbiorowego
4. Możliwe jest ponowne użycie informacji poprzez:
    1. agregację
    2. dziedziczenie

# Schemat firmy (**obiektowy**)




Rysunek: Model obiektowej bazy danych utworzony w programie WhiteStarUML

# Kod definiujący obiektowy model danych

```
Firma {
String nazwa;
Adres adres;
Pracownik dyrektor;
List<Oddzial> oddzialy;
}

Oddzial {
String nazwa;
Adres adres;
Pracownik kierownik;
List<Pracownik> pracownicy;
}
```

```
Adres {
String miejscowosc;
String ulica;
Integer numer;
}

Osoba {
String nazwisko;
Integer wiek;
Adres adres;
}

Pracownik extends Osoba {
Integer wynagrodzenie;
}
```

# Porównanie dwóch modeli na przykładzie firmy

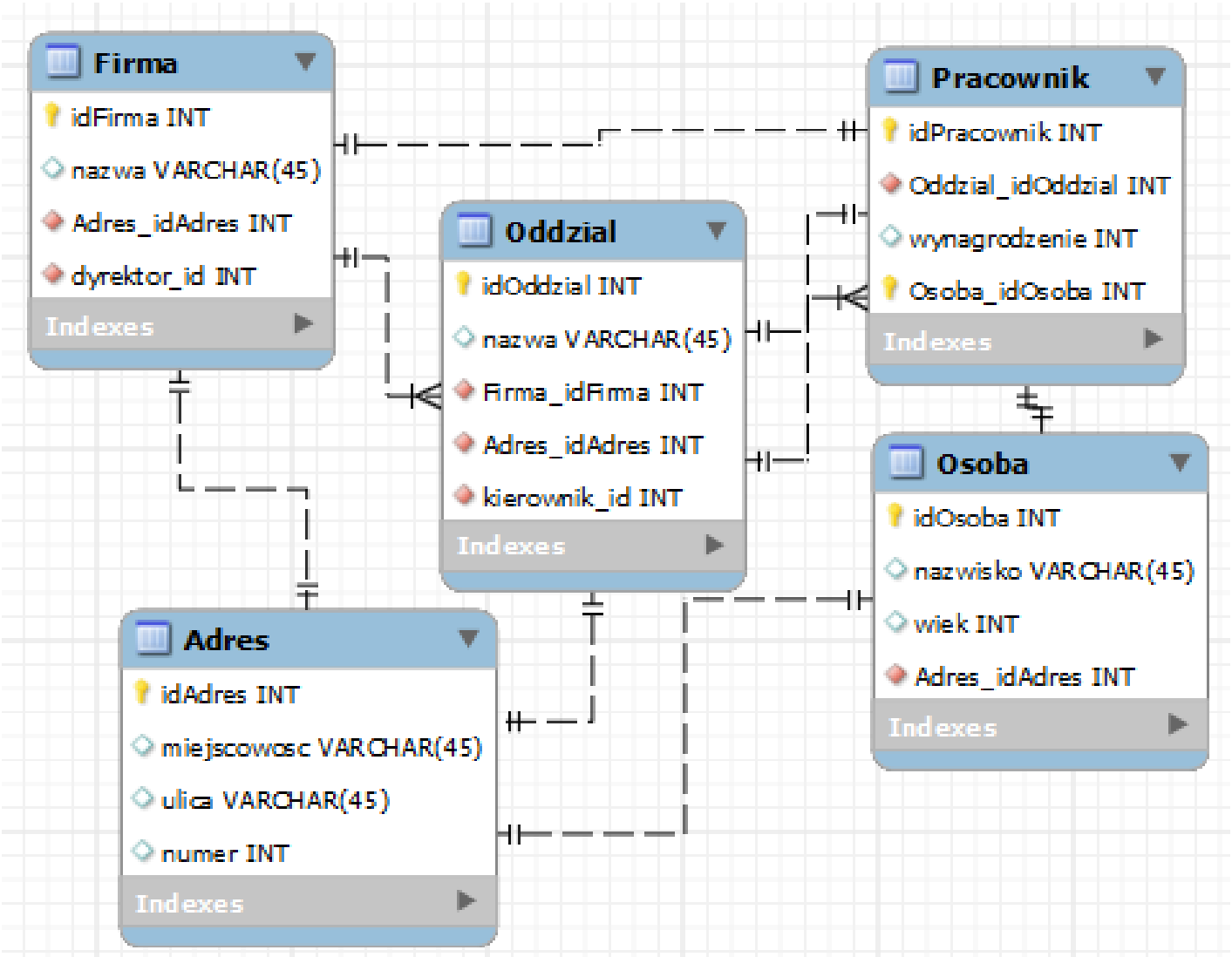
## Polecenie select

Wybierz dyrektorów firm lubelskich, których zarobki przekraczają 10tys. zł

```
SELECT
p.idPracownik
FROM
Firma as "f", Adres as "a",
Pracownik as "p"
WHERE
f.Adres_idAdres=a.idAdres AND
f.dyrektor_id=p.idPracownik AND
p.wynagrodzenie > 10000 AND
a.miejscowosc='Lublin'
```

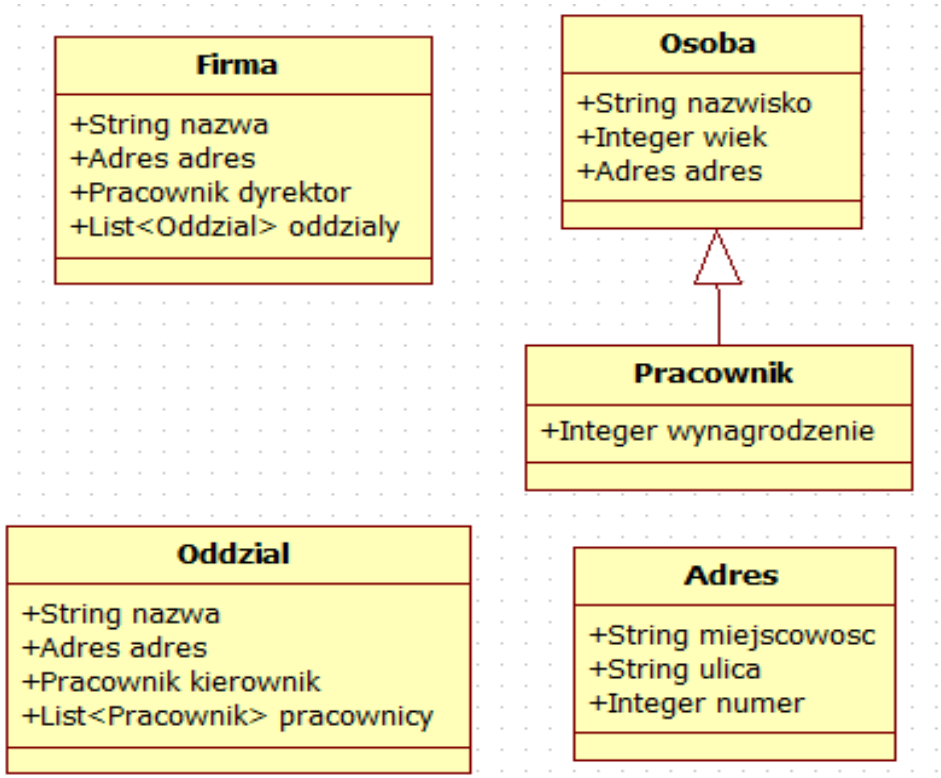
```
SELECT
f.dyrektor
FROM
f IN Firma
WHERE
f.adres.miejscowosc='Lublin'
AND
f.dyrektor.wynagrodzenie
> 10000
```

# Schemat firmy (**relacyjny**)




Rysunek: Model bazy danych utworzony w programie Workbench

# Schemat firmy (**obiektowy**)




Rysunek: Model obiektowej bazy danych utworzony w programie WhiteStarUML

# Przykładowe obiektowe bazy danych

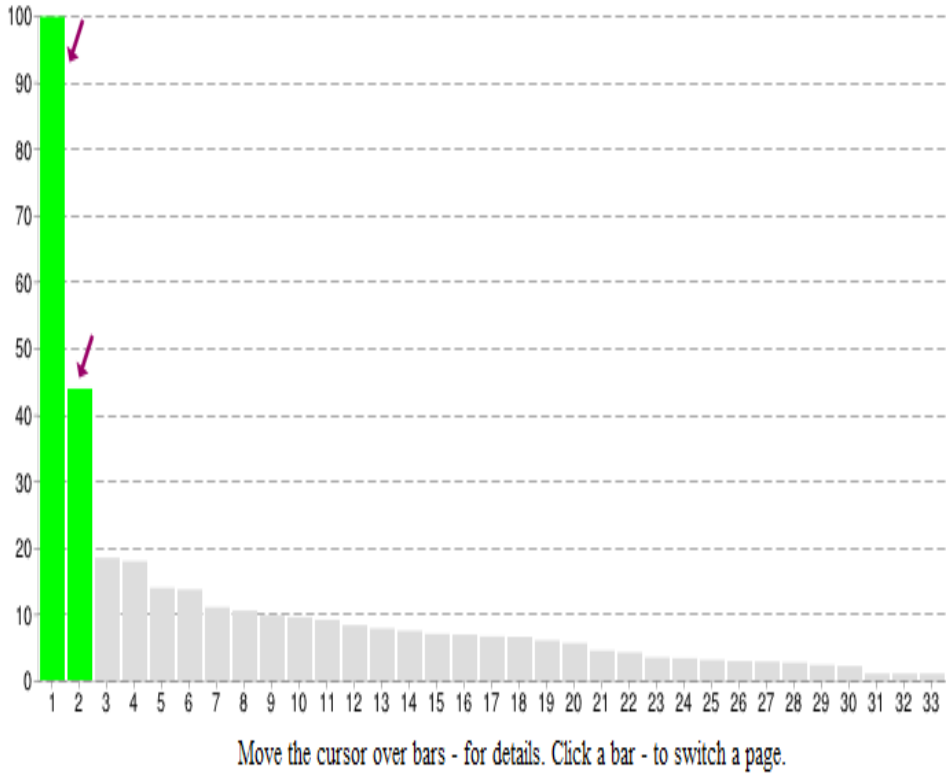
## Uwaga!

Obiektowe bazy danych mają dziś stosunkowo **mały** udział procentowy na rynku. Przytoczę przykłady kilku dostępnych technologii.

## Przykładowe obiektowe bazy danych

- SQL3 (rozszerzenia języka SQL2 o cechy obiektowe)
- Illustra (rozwinięcie Postgresu)
- ObjectStore (rozszerzenie C++)
- GemStone (rozszerzenie Smalltalk-u)
- Oracle

# ObjectDB




Rysunek: Porównanie wydajności obiektowej bazy ObjectDB z relacyjnymi bazami danych wykorzystujących mapowanie obiektowo-relacyjne

# Szauncki wg Forrester Research (2012r.)

## Szauncki wg Forrester Research (2012r.)

1. **35 mld USD** - wartość rynku relacyjnych baz danych na świecie (licencje na oprogramowanie, pomoc techniczna, pensje pracowników, inne usługi)
2. **25%** tyle danych biznesowych jest strukturyzowana
3. **65%** tyle z nich jest przechowywana w relacyjnych bazach danych

## NoSQL

Termin NoSQL czyli Not Only SQL - określa systemy zarządzania, które nie bazują na modelu relacyjnym

# NoSQL (2)

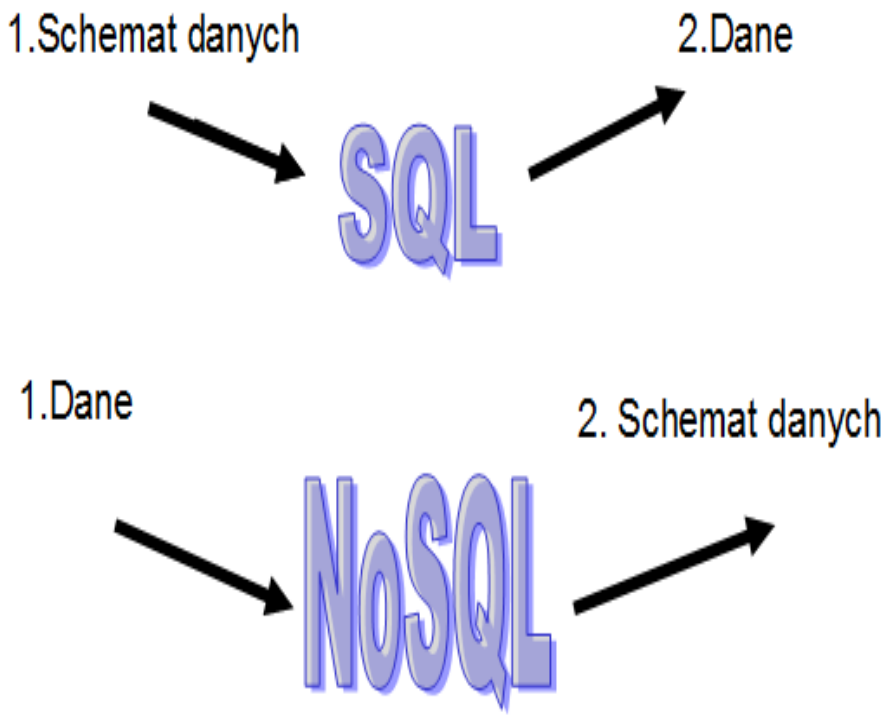
## Rynek NoSQL

Na rynku dostępnych jest aż ok. **50** technologii nierelacyjnych baz danych

## Założenia technologii NoSQL

- Rezygnację z wielu elementów baz relacyjnych
- Zmniejszenie znaczenia schematów danych

# Graficzna ilustracja porównania




Rysunek: Wg założeń NoSQL uwaga powinna być skupiona najpierw na danych, a dopiero potem na schematach

# Ruch NoSQL

## Ruch NoSQL

Ruch NoSQL jest ruchem zajmującym się propagowaniem wiedzy na temat NoSQL

**Mało** jest fachowej literatury poruszającej zagadnienia NoSQL, a temat nie jest poruszany w szkołach

# Założenia ruchu

## Założenia ruchu

- **Standaryzacja interfejsu dostępu do baz NoSQL5**
- **Eliminacja najsłabszych rozwiązań**
- **Problem z budową zapytań**
- **Zagwarantowanie wsparcia dla swoich rozwiązań**

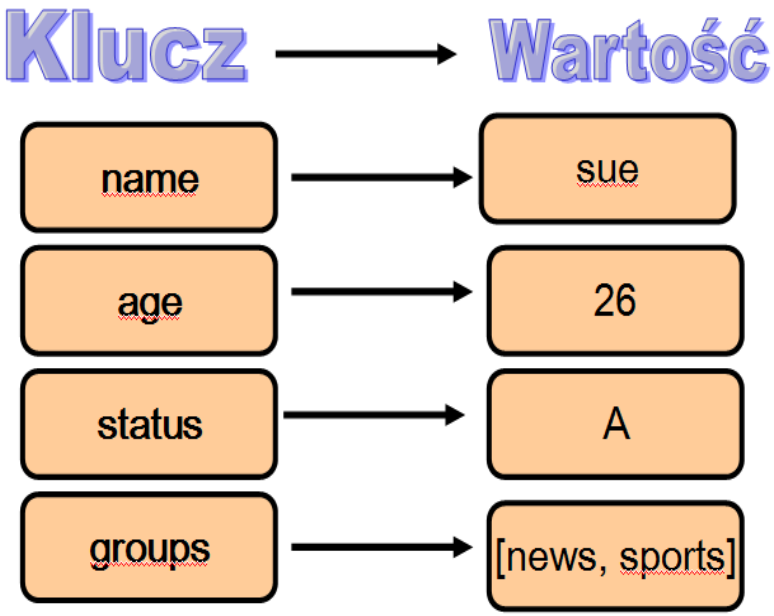
# Typy baz danych NoSQL

## 1. Bazy obiektowe

(omówione w poprzedniej części)

## 2. Bazy klucz-wartość (key-value)

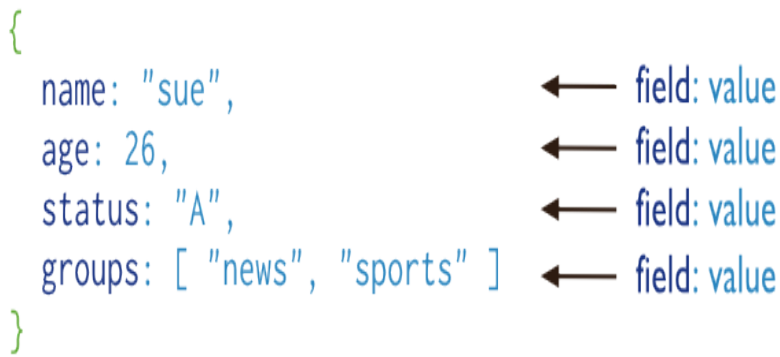
Są to tabele, zawierające dwie kolumny tekstowe: klucz oraz wartość.
Są niezwykle **szybkie**



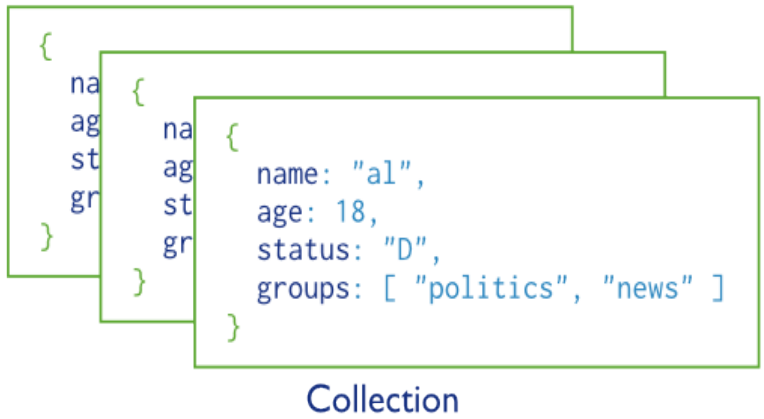
# Typy baz danych NoSQL (2)

## Bazy dokumentowe

Zamiast wierszy używa się pojęcia dokumentu, zawierającego pary klucz-wartość.
Firmy korzystające z w/w rozwiązania: New York Times, Disney, MTV Networks, IGN Entertainment, The Guardian




Rysunek: Dokument w bazie mongoDB




Rysunek: Kolekcja dokumentów

## Modyfikacja bazy

Załóżmy, że chcielibyśmy dodać informację, że John jest niepełnosprawny, a Al jest wyróżniający się.

```
{
name: John
age: 24
status: B
groups: ["news","sports"]
}
{
name: Al
age: 18
status: D
groups: ["politics"]
}
```

| | A | B | C | D |
|---|---|---|---|---|
| 1 | name | age | status | groups |
| 2 | Sue | 26 | A | 112 |
| 3 | John | 24 | B | 113 |
| 4 | Al. | 18 | D | 114 |

Rysunek: W atrybucie groups przechowujemy klucz obcy

# Modyfikacja bazy (2)

## Wniosek

Dokumentowe bazy danych są elastyczniejsze niż relacyjne!

```
{
name: John
age: 24
status: B
groups: ["news","sports"]
disabled: true
}
{
name: Al
age: 18
status: D
groups: ["politics"]
special: true
}
```

| | A | B | C | D | E |
|---|---|---|---|---|---|
| 1 | name | age | status | groups | others |
| 2 | Sue | 26 | A | 112 | null |
| 3 | John | 24 | B | 113 | disabled |
| 4 | Al. | 18 | D | 114 | special |

Rysunek: Definiujemy nowy atrybut np. others varchar(45)
W pozostałych komórkach (może być ich setki!) wpisujemy wartość null

# ORM

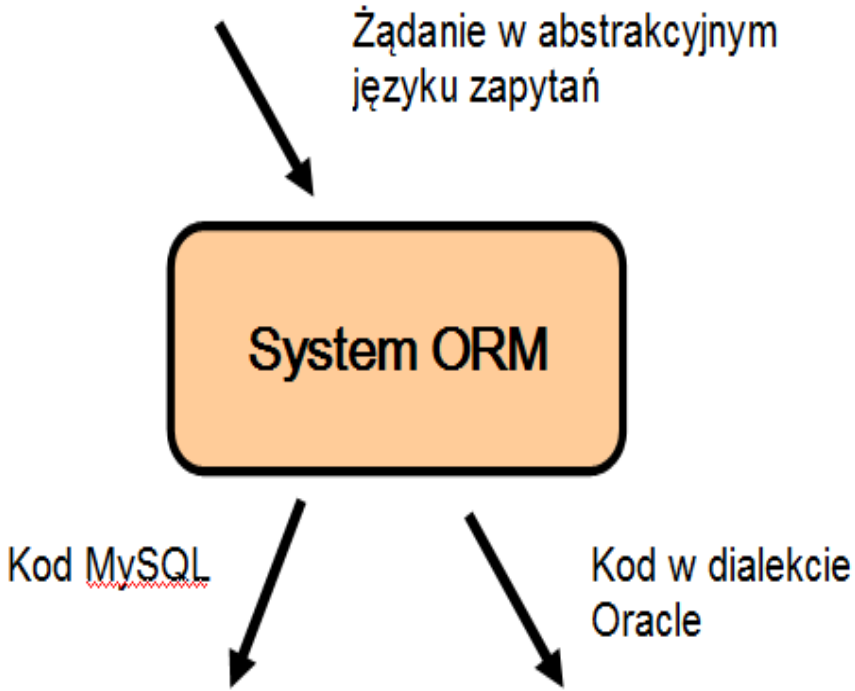
## ORM (ang. *Object Relational Mapping*)

Mapowaniem obiektowo-relacyjnym (ORM) nazywamy technikę odwzorowania obiektów utworzonych w programie w taki sposób, aby mogły być zapisane w postaci tabel

## Systemy takie uzyskały nazwę

**obiektowo-relacyjnych baz danych**

# Przykład systemu mapowania obiektowo-relacyjnego: Hibernate, JPA




Rysunek: Programista operuje na abstrakcyjnym języku zapytań, niezależnym od konkretnego dialektu SQL

# Znaczenie dla programisty

## Znaczenie dla programisty

- Aby zapisać obiekt **zlecamy** systemowi ORM zapisanie obiektu w bazie
- Operuje na abstrakcyjnym języku zapytań, **niezależnym** od konkretnego dialektu SQL

Dzięki takim możliwościom, odnosimy wrażenie pracy jedynie w **środowisku obiektowym**

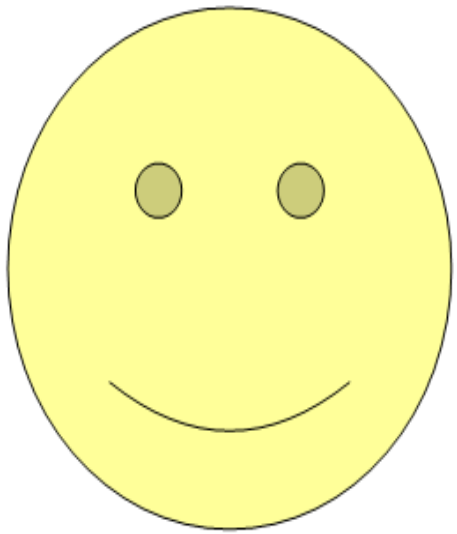
# System ORM

## System ORM odpowiada za:

- niskopoziomową obsługę bazy danych
  - ustanawianie połączenia
  - aspekt sieciowy
- tłumaczenie żądań programu na kod SQL **specyficzny dla konkretnego dialektu SQL**

# Dziękuję za uwagę

**Dziękuję za uwagę!**



Rysunek: ostatni rysunek prezentacji