# Nierelacyjne bazy danych

Wprowadzenie do baz danych typu NoSQL


Grzegorz Gołaszewski, Wojciech Waloszek, Teresa Zawadzka, Michał Zawadzki

# Zasady prowadzenia przedmiotu (1)

Osoby prowadzące:

- mgr inż. Grzegorz Gołaszewski
- dr inż. Wojciech Waloszek
- **dr inż. Teresa Zawadzka**
- dr inż. Michał Zawadzki – gościnnie przedstawiciel EPAM Systems

# Zasady prowadzenia przedmiotu (2)

1. Wykład 1: **Wprowadzenie do baz danych typu NoSQL** (3h wykład/warsztat) - *T. Zawadzka*
2. Blok tematyczny 1: **Dokumentowe bazy danych na przykładzie MongoDB** (3h laboratorium + 3h warsztat/wykład + 3h laboratorium) – *M. Zawadzki*
3. Blok tematyczny 2: **Grafowe bazy danych na przykładzie Neo4J** (3h laboratorium + 3h warsztat/wykład + 3h laboratorium) – *T. Zawadzka*
4. Blok tematyczny 3: **Bazy danych o organizacji kolumnowej na przykładzie HBase** (3h laboratorium + 3h warsztat/wykład + 3h laboratorium) – *W. Waloszek*
5. Blok tematyczny 4: **Bazy danych typu *key-value* na przykładzie Oracle NoSQL Database** (3h laboratorium + 3h warsztat/wykład + 3h laboratorium) – *G. Gołaszewski*
6. Zajęcia dodatkowe

# Zasady zaliczenia przedmiotu (3)

1. Obecność na wszystkich zajęciach jest obowiązkowa.

*(wykład/warsztat stanowi część bloku tematycznego i nieusprawiedliwiona nieobecność na wykładzie/warsztacie powoduje niedopuszczenie do drugiego laboratorium w bloku tematycznym)*

2. Wymagana jest punktualność na zajęciach.

*(dwa spóźnienia dłuższe niż 10 minut skutkują niezaliczeniem jednego bloku tematycznego)*

3. Zaliczenie przedmiotu wymaga zaliczenia wszystkich bloków tematycznych oraz egzaminu na minimum 50%.

*(w przypadku usprawiedliwionej nieobecności prowadzący blok tematyczny określa warunki jego zaliczenia)*

4. Uzyskanie z każdego bloku tematycznego 95% punktów zwalnia z egzaminu z oceną bardzo dobrą z całego przedmiotu.
5. Każdy blok tematyczny kończy się podaniem listy zagadnień obowiązujących na egzaminie.

# Ocena

1. Ocena jest wyznaczana na podstawie punktów uzyskanych z egzaminu i bloków tematycznych.
2. Punkty uzyskane z pojedynczego bloku tematycznego stanowią 20% oceny, zaś pozostałe 20% stanowią punkty z egzaminu 20%.
3. Ocena z przedmiotu:
    - ≥100% - Dokument wyróżnienia
    - ≥ 90% - 5.0
    - ≥ 80% - 4.5
    - ≥ 70% - 4.0
    - ≥ 60% - 3.5
    - ≥ 50% - 3.0

# Kierunkowe efekty kształcenia (1)

[K_U07] analizuje problemy i tworzy właściwe modele, struktury danych oraz algorytmy heurystyczne i numeryczne, ocenia ich złożoność obliczeniową, szacuje błędy otrzymanych rozwiązań

**W ramach przedmiotu studenci uczą się dobierać odpowiednie bazy danych do konkretnych zastosowań biznesowych.**

POLITECHNIKA GDAŃSKA

ETI

# Kierunkowe efekty kształcenia (2)

[K_W04] zna protokoły komunikacji, modele przetwarzania danych, technologie klasycznego i sematycznego Internetu, działanie serwera www, aplikacje internetowe i wzorce programowe oraz budowę aplikacji hostowanych w przeglądarce

**W ramach przedmiotu studenci zapoznają się z modelami przetwarzania danych typu NoSQL: dokumentami, grafami, danymi o organizacji kolumnowej oraz strukturami danych typu klucz-wartość.**

POLITECHNIKA GDAŃSKA

ETI

# Kierunkowe efekty kształcenia (3)

[K_W08] zna problemy współdzielenia stanu, prezentacji i transformacji informacji w systemie rozproszonym, technologie hipermediów i związanych z nimi usług, architektury interaktywnej symulacji rozproszonej oraz metody interakcji agentów

**W ramach przedmiotu studenci poznają metody rozpraszania danych: shardingu i replikacji oraz teorię CAP i BASE.**

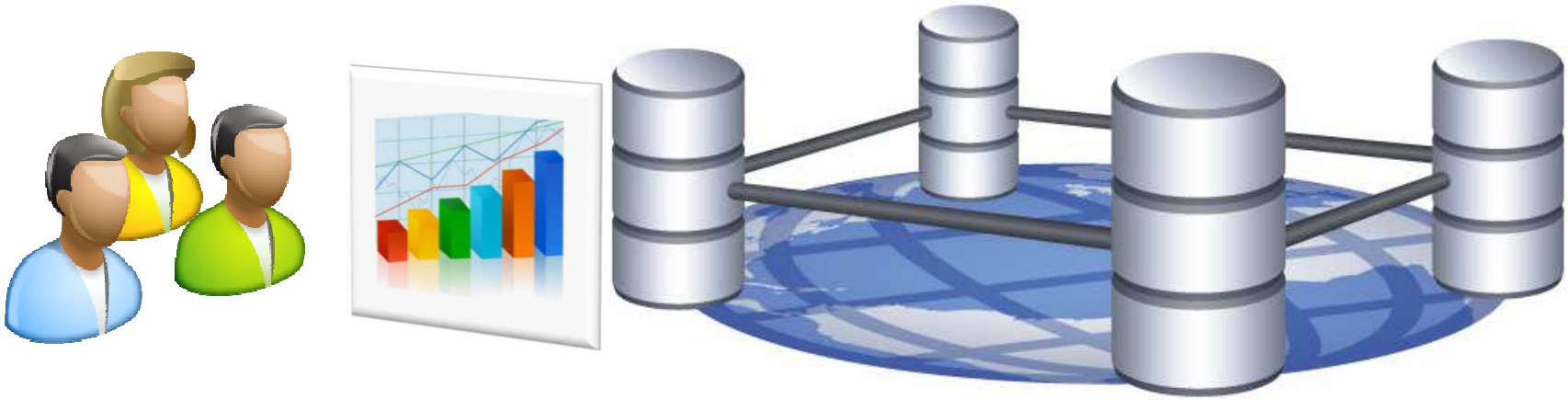
POLITECHNIKA GDAŃSKA

ETI

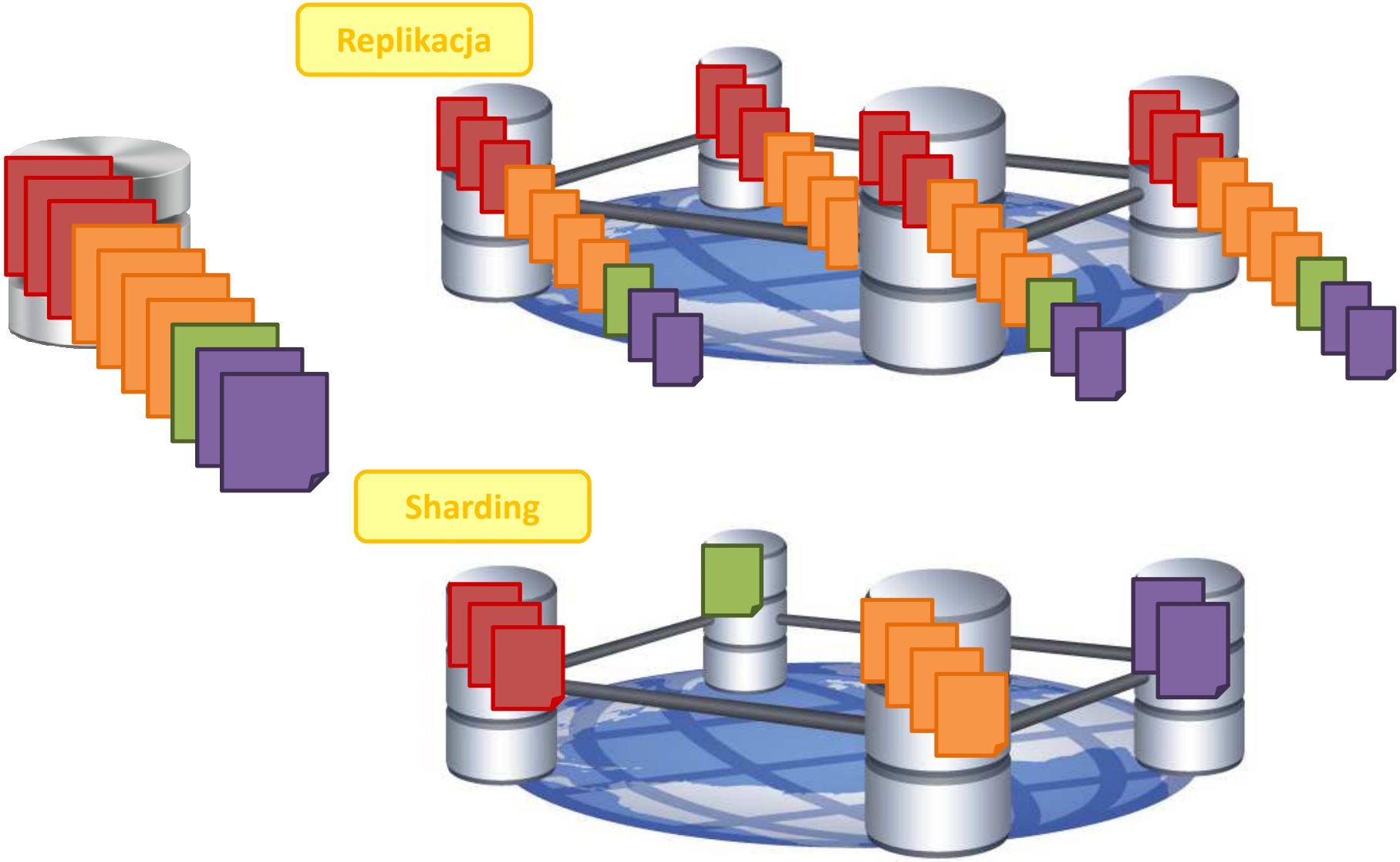
# NoSQL – co to takiego?


- Struktura inna niż relacyjna
- Wydajne działanie w rozproszeniu
- Brak ustalonego schematu danych
- Brak ACID

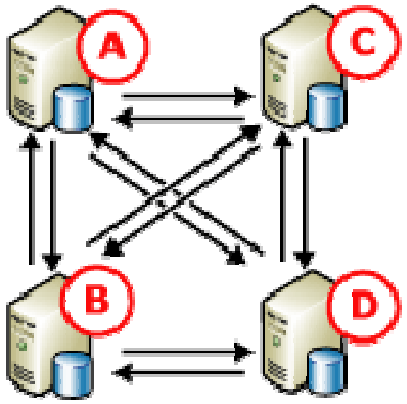
# Co to jest rozproszona baza danych?



# Replikacja i sharding

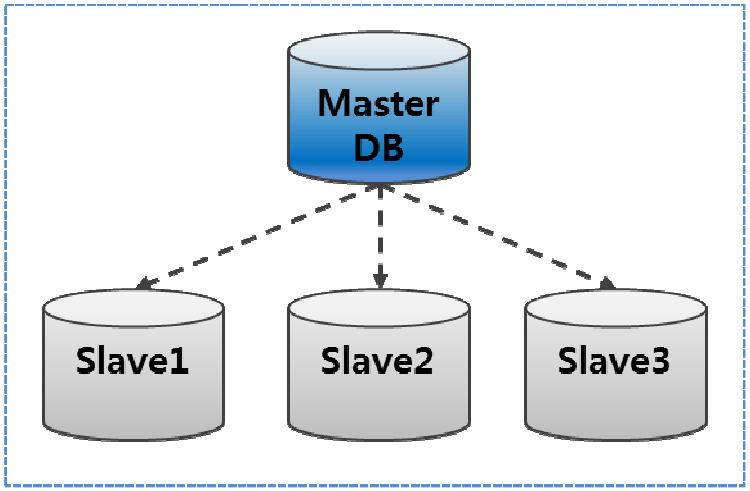


# Replikacja *peer-to-peer*




1. Replikacja peer-to-peer umożliwia zapis do dowolnego węzła.
2. Węzeł koordynuje synchronizację danych z innymi węzłami.

# Replikacja *master-slave*




1. Zapis tyko przez węzeł typu *master*.
2. Węzeł typu *master* jest odpowiedzialny za synchronizację danych pomiędzy węzłami typu *slave*.
3. Odczyt z dowolnego węzła.

# ACID

- *Atomicity* – atomowość
- *Consistency* – spójność
- *Isolation* – izolacja
  1. *read uncommited*
  2. *read commited*
  3. *repeatable read*
  4. *serializable*
- *Durability* - trwałość

# Relacyjne bazy danych - cechy

- Niedopasowane do olbrzymich zbiorów danych z różnymi typami danych (np. zdjęcia, filmy, tekst).
- Problem skalowalności olbrzymich zbiorów danych
  - **Skalowanie pionowe** (ang. *scale-up*): ograniczenie pamięci i CPU,
  - **Skalowanie poziome** (ang. *scale-out*): problem synchronizacji danych pomiędzy węzłami.
- Spójność wąskim gardłem dla skalowalności RDBMSów.

# Teoria CAP (1)

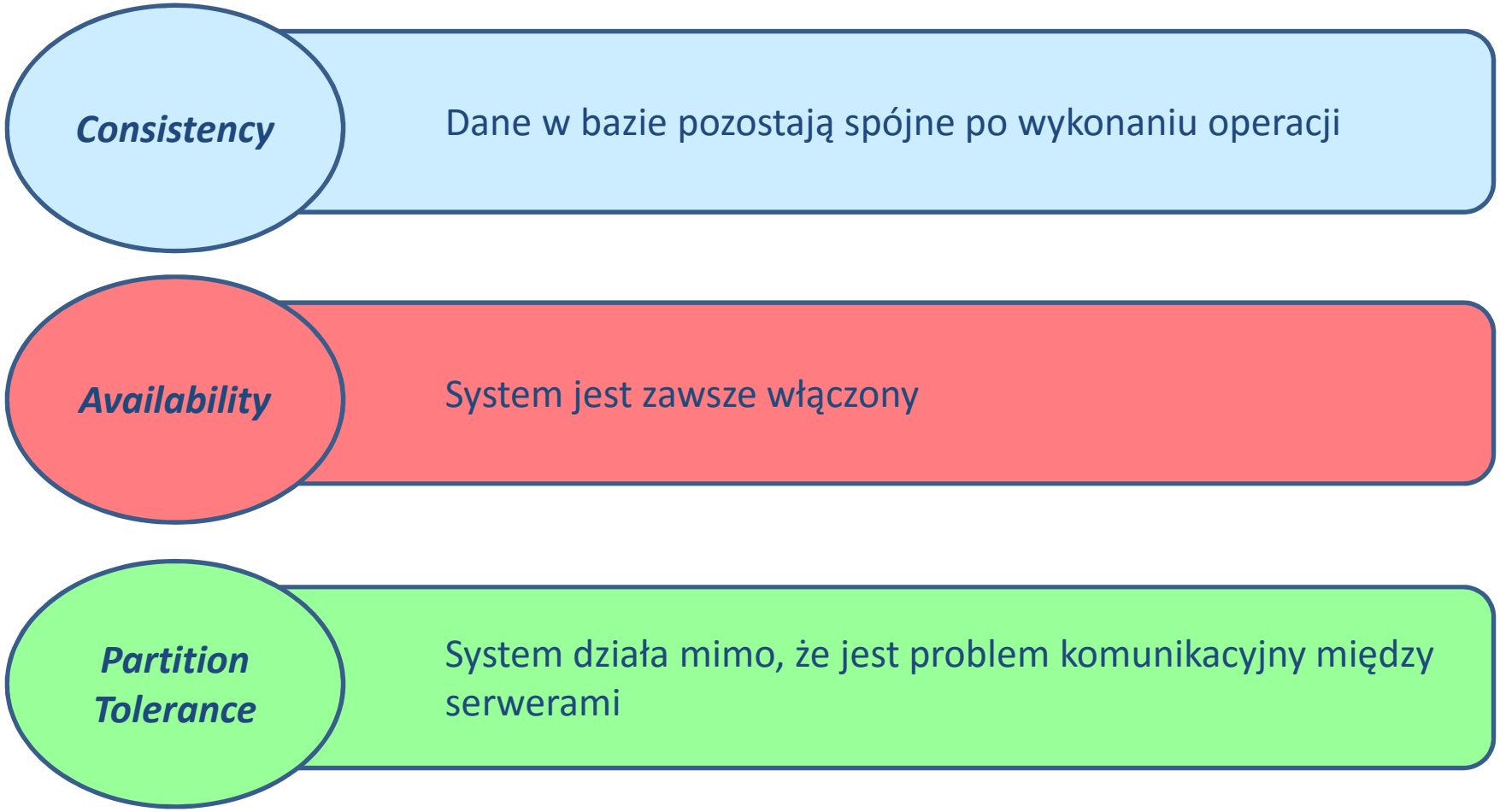
**Warsztat – Korporacja Zapamiętywacz**

***Consistency***: klienci po przekazaniu informacji do zapamiętania, otrzymają przy następnym telefonie zawsze prawidłową informację, bez względu na to jak szybko zadzwonią.

***Availability***: korporacja jest gotowa odbierać telefony w godzinach urzędowania.

***Partition Tolerance***: korporacja działa nawet jeśli żona Jana Niezapominalskiego się do niego nie odzywa.

# Teoria CAP (2)


Consistency
Dane w bazie pozostają spójne po wykonaniu operacji
Availability
System jest zawsze włączony
Partition Tolerance
System działa mimo, że jest problem komunikacyjny między serwerami

# Teoria CAP (3)

CA - wszystkie węzły są zawsze ze sobą połączone, jeśli któryś węzeł jest niedostępny system przestaje działać

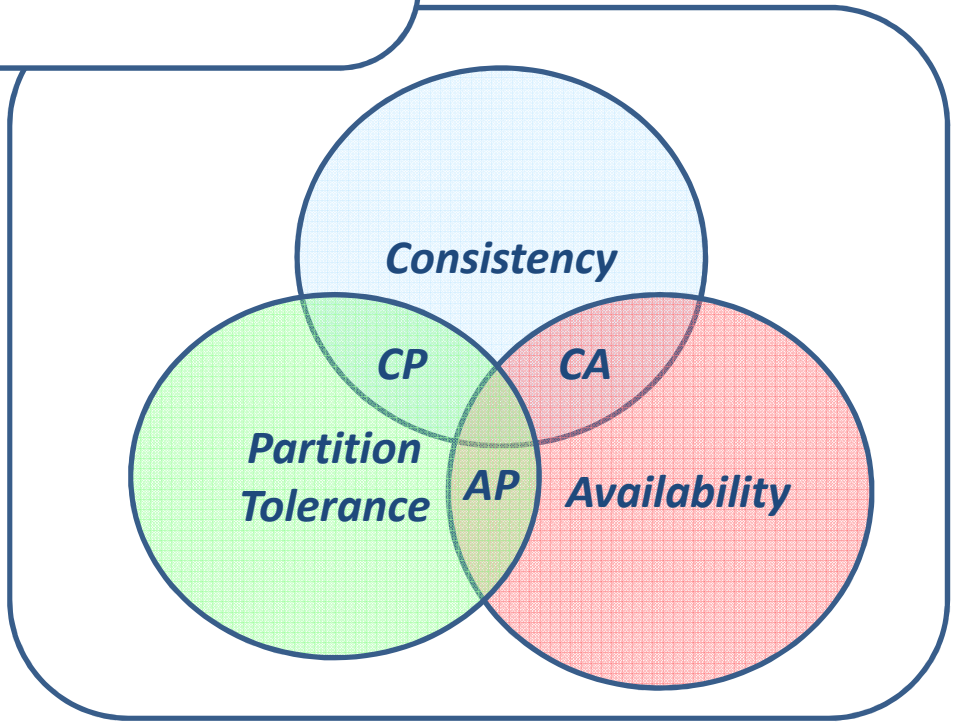
CP - część danych może nie być dostępnych, ale reszta jest cały czas spójna i aktualna

AP - system jest cały czas dostępny, mimo faktu, że występuje partycjonowanie, natomiast niektóre dane mogą nie być poprawne/aktualne

**W praktyce tylko**

**2 z 3**

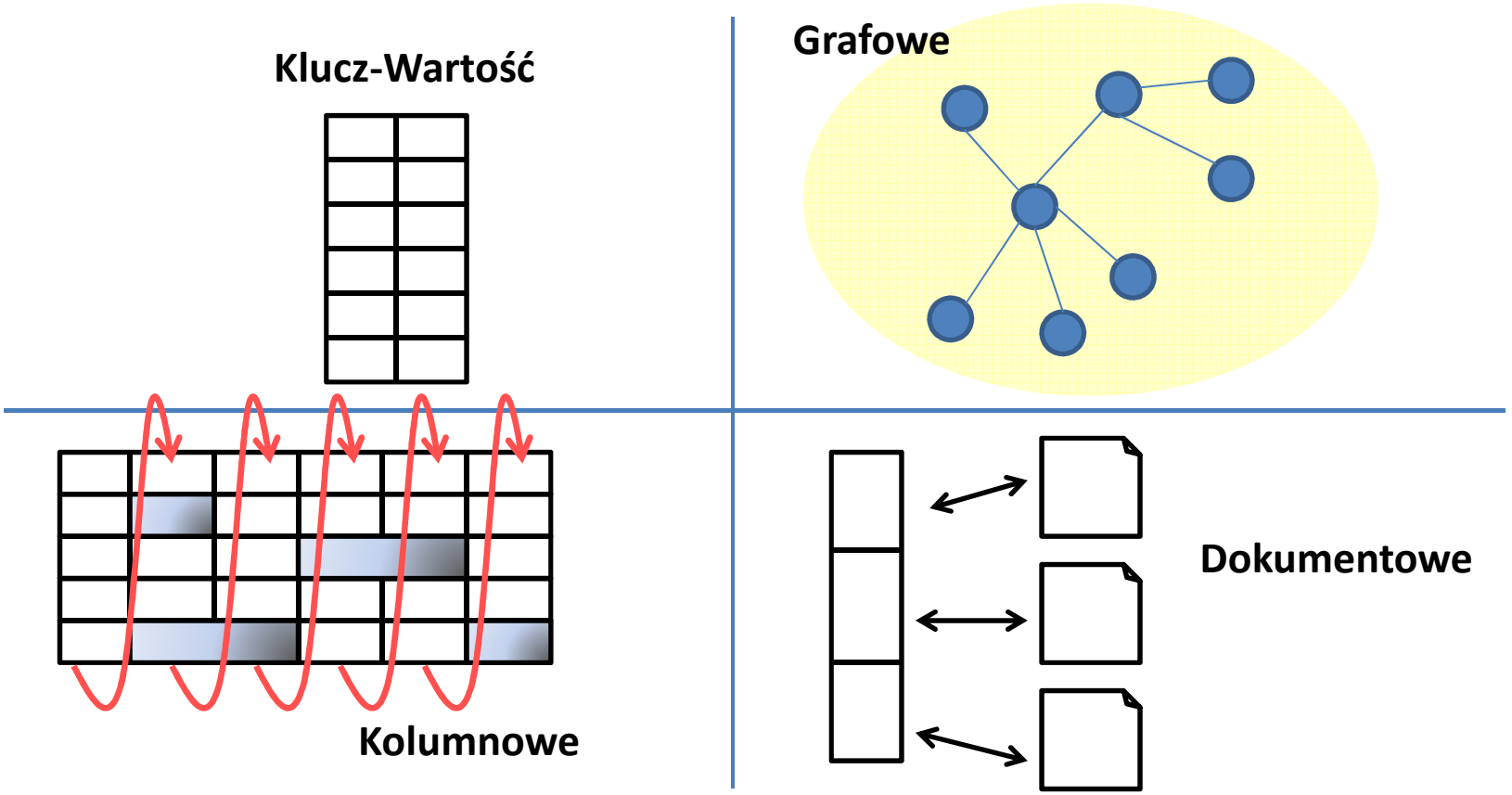
**postulatów CAP mogą być spełnione!!!**



# BASE

1. *Basically Available* – system gwarantuje dostępność w rozumieniu teorii CAP
2. *Soft state* – stan systemu może zmieniać się w czasie, nawet w przypadku braku nowego wejścia.
3. *Eventual consistency* – system będzie spójny w pewnym momencie czasu, zakładając, że w tym czasie system nie otrzyma nowego wejścia.

# Bazy danych typu NoSQL



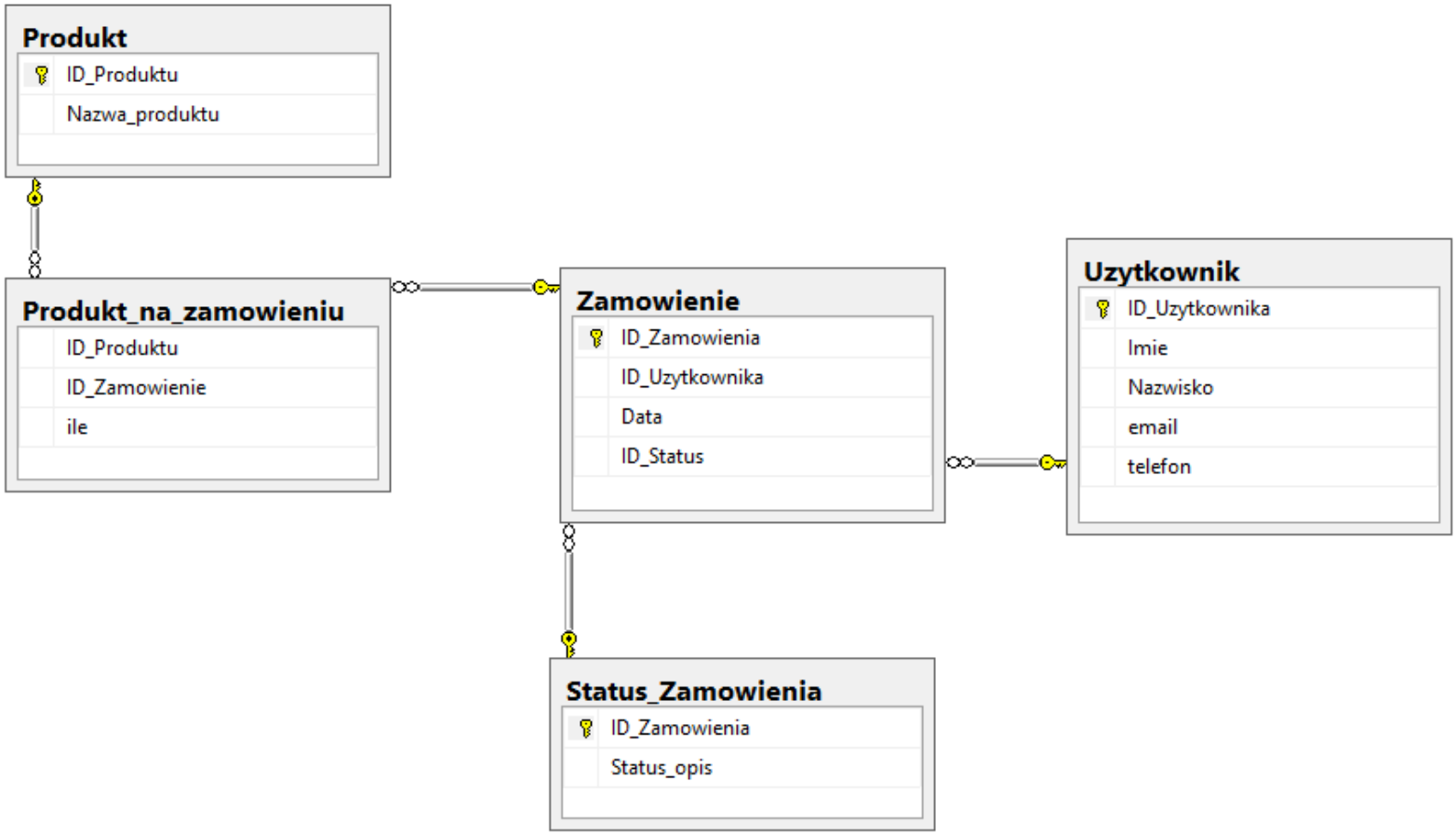
# Problem biznesowy

## Przechowywanie danych o zamówieniach


# Przykładowe zapytania

1. Podaj wszystkie zamówienia Alicji Zawadzkiej.
2. Podaj ostatnie zamówienie Alicji Zawadzkiej.
3. Co zostało zamówione przez Alicję Zawadzką w ramach zamówienia nr 1234?
4. Podaj, czy jest (jeżeli tak to jaki) produkt, który został zamówiony w ostatnim i przedostatnim zamówieniu Alicji Zawadzkiej.

# Relacyjna baza danych



# Tworzenie bazy danych

## Tworzenie bazy danych i ładowanie danych

```
CREATE TABLE Uzytkownik(
	ID_Uzytkownika integer PRIMARY KEY,
	Imie varchar(10) NOT NULL,
	Nazwisko varchar(20) NOT NULL,
	email varchar(20) NOT NULL,
	telefon varchar(20),
)
GO

CREATE TABLE Status_Zamowienia(
	ID_Zamowienia integer IDENTITY(1,1) PRIMARY KEY,
	Status_opis varchar(20),
)
GO
…
```

# Wypełnienie bazy danymi

| | ID_Produktu | Nazwa_produktu |
|---|---|---|
| 1 | abcd | lody truskawkowe |
| 2 | cdef | kawa espresso |
| 3 | efab | brukselka |

| | ID_Uzytkownika | Imie | Nazwisko | email | telefon |
|---|---|---|---|---|---|
| 1 | 1 | Alicja | Zawadzka | alzaw@gmail.com | 678342567 |

| | ID_Produktu | ID_Zamowienie | ile |
|---|---|---|---|
| 1 | abcd | 1234 | 1 |
| 2 | efab | 1234 | 1 |
| 3 | cdef | 1234 | 1 |
| 4 | cdef | 5678 | 1 |

| | ID_Zamowienia | ID_Uzytkownika | Data | ID_Status |
|---|---|---|---|---|
| 1 | 1234 | 1 | 2012-08-08 | 3 |
| 2 | 5678 | 1 | 2012-08-16 | 2 |

| | ID_Statusu | Status_opis |
|---|---|---|
| 1 | 1 | nie zaplacone |
| 2 | 2 | w realizacji |
| 3 | 3 | dostarczone |

# Zapytanie 1

Podaj wszystkie zamówienia Alicji Zawadzkiej

```
select
z.ID_Zamowienia, z.Data, s.Status_opis
from
dbo.Zamowienie z, dbo.Status_Zamowienia s,
dbo.Uzytkownik u
where
z.ID_Uzytkownika = u.ID_Uzytkownika and
s.ID_Statusu = z.ID_Status and Imie='Alicja'
and Nazwisko='Zawadzka'
```

# Zapytania 2 – wersja 1

## Podaj ostatnie zamówienie Alicji Zawadzkiej

```
select TOP 1
z.ID_Zamowienia, z.Data, s.Status_opis
from
dbo.Zamowienie z, dbo.Status_Zamowienia s,
dbo.Uzytkownik u
where
z.ID_Uzytkownika = u.ID_Uzytkownika and
s.ID_Statusu = z.ID_Status and u.Imie='Alicja'
and u.Nazwisko='Zawadzka'
order by z.Data DESC
```

# Zapytania 2 – wersja 2

## Podaj ostatnie zamówienie Alicji Zawadzkiej

```
select
ID_Zamowienia, Data from Zamowienie z,
Uzytkownik u
where
z.ID_Uzytkownika = u.ID_Uzytkownika and
u.Imie='Alicja' and u.Nazwisko='Zawadzka' and
z.Data = (SELECT MAX(z1.Data) from Zamowienie
          z1, Uzytkownik u1 where
          z.ID_Uzytkownika=u.ID_Uzytkownika and
          u.Imie='Alicja' and
          u.Nazwisko='Zawadzka')
```

# Zapytania 3

Co zostało zamówione przez Alicję Zawadzką w ramach zamówienia nr 1234?

```
select
p.Nazwa_produktu
from
dbo.Produkt p, dbo.Produkt_na_zamowieniu pz
where
pz.ID_Zamowienie = '1234' and
pz.ID_Produktu = p.ID_Produktu
```

# Zapytanie 4 – wersja 1

Podaj, czy jest (jeżeli tak to jaki) produkt, który został zamówiony w ostatnim i przedostatnim zamówieniu AZ

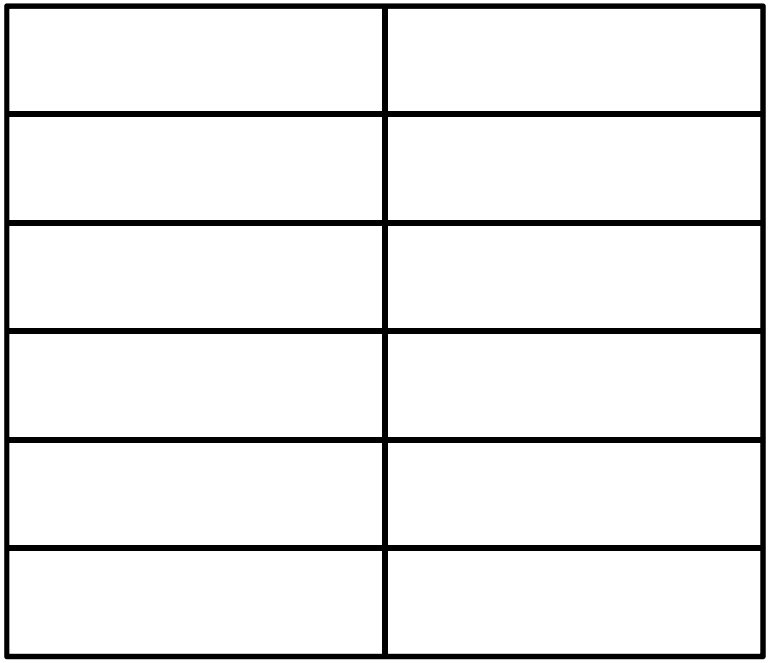
```
SELECT * FROM
( select TOP 1 z.ID_Zamowienia AS Ost_id from dbo.Zamowienie z,
    dbo.Status_Zamowienia s, dbo.Uzytkownik u where
    z.ID_Uzytkownika = u.ID_Uzytkownika and s.ID_Statusu =
    z.ID_Status order by z.Data DESC
  ) A,
(  select z.ID_Zamowienia AS PrzedOst_ID from dbo.Zamowienie z,
    dbo.Status_Zamowienia s, dbo.Uzytkownik u where
    z.ID_Uzytkownika = u.ID_Uzytkownika and s.ID_Statusu =
    z.ID_Status order by z.Data DESC
   OFFSET 1 ROWS
   FETCH NEXT 1 ROWS ONLY ) B,
   dbo.Produkt_na_zamowieniu pz1,
   dbo.Produkt_na_zamowieniu pz2
WHERE
    A.Ost_id = pz1.ID_Zamowienie and B.PrzedOst_ID =
    pz2.ID_Zamowienie AND pz1.ID_Produktu = pz2.ID_Produktu
```

# Zapytanie 4 – wersja 2

Podaj, czy jest (jeżeli tak to jaki) produkt, który został zamówiony w ostatnim i przedostatnim zamówieniu AZ

```
SELECT pz1.ID_Produktu FROM
( select TOP 1 z.ID_Zamowienia from dbo.Zamowienie z,
     dbo.Status_Zamowienia s, dbo.Uzytkownik u where z.ID_Uzytkownika =
     u.ID_Uzytkownika and s.ID_Statusu = z.ID_Status order by z.Data DESC
  ) A, dbo.Produkt_na_zamowieniu pz1
WHERE
     A.ID_Zamowienia = pz1.ID_Zamowienie
INTERSECT (
SELECT pz2.ID_Produktu FROM
(   select z.ID_Zamowienia from dbo.Zamowienie z, dbo.Status_Zamowienia s,
     dbo.Uzytkownik u where z.ID_Uzytkownika = u.ID_Uzytkownika and
     s.ID_Statusu = z.ID_Status order by z.Data DESC
   OFFSET 1 ROWS
   FETCH NEXT 1 ROWS ONLY ) B, dbo.Produkt_na_zamowieniu pz2
WHERE
     B.ID_Zamowienia = pz2.ID_Zamowienie
)
```

# Baza danych typu klucz-wartość (ang. *key-value*)



# Bazy danych typu klucz-wartość

- Rozproszona tablica mieszająca (ang. *hash table*).
- Przechowywane pary klucz: wartość.
- Baza nie rozumie wartości (czarna skrzynka).
- Trwałość oraz duża wydajność i skalowalność.
- Wydajność i skalowalność osiągnięta przez rozproszenie.

# Reprezentacja typu key-value

- Dodanie danych do bazy

  PUT(klucz, wartość)
- Pobranie danych z bazy

  GET(klucz)
- Usunięcie danych z bazy

  DELETE(klucz)

# Oracle NoSQL Database

- Cechą szczególną Oracle NoSQL database jest modułowa budowa klucza
- Klucz złożony z części głównych i podrzędnych
- Wartości o takiej samej głównej części klucza przechowywane w tym samym miejscu

/Ala/Kowalska/-/adres/ulica

/Ala/Kowalska/-/telefon

/Ala/Kowalska/email/-/służbowy

# Problem Alicji Zawadzkiej (1)

- Zapis danych

OsobaIJejZakupy alicja; //obiekt zawierający wszystkiedane o Alicji Zawadzkiej

baza.put(„AlicjaZawadzka”,alicja.serialize());

- Odczyt danych

alicja = (baza.get(„AlicjaZawadzka”)).deserialize();

//tu sprawdzenie konkretnych danych

# Problem Alicji Zawadzkiej (2)

- W Oracle NoSQL database można zrobić strukturę:

/Zawadzka/Alicja/-/5678/data: 2012-08-16

/Zawadzka/Alicja/-/5678/szczegóły/cdef: kawa espresso

/Zawadzka/Alicja/-/5678/status: w realizacji

# Problem Alicji Zawadzkiej (2)

- W Oracle NoSQL database można zrobić strukturę:

/Zawadzka/Alicja/-/1234/data: 2012-08-08

/Zawadzka/Alicja/-/1234/szczegóły/abcd: lody truskawkowe

/Zawadzka/Alicja/-/1234/szczegóły/efab: brukselka

/Zawadzka/Alicja/-/1234/szczegóły/cdef: kawa espresso

/Zawadzka/Alicja/-/1234/status: dostarczone

# Problem Alicji Zawadzkiej (3)

1. Podaj wszystkie zamówienia Alicji Zawadzkiej.

   multiget(„/Zawadzka/Alicja”);

2. Podaj ostatnie zamówienie Alicji Zawadzkiej.
   – Trzeba pobrać wszystkie zamówienia i przejrzeć daty

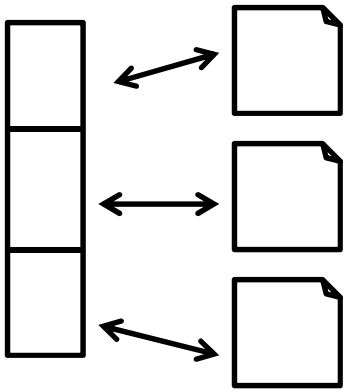
# Problem Alicji Zawadzkiej (3)

3. Co zostało zamówione przez Alicję Zawadzką w ramach zamówienia nr 1234?

   multiget(„/Zawadzka/Alicja/-/1234/szczegóły”)

4. Podaj, czy jest (jeżeli tak to jaki) produkt, który został zamówiony w ostatnim i przedostatnim zamówieniu Alicji Zawadzkiej.
   – Znów trzeba pobrać wszystkie zamówienia i je przejrzeć

# Dokumentowe bazy danych



# Dokumentowe bazy danych

- Podstawowym elementem jest *dokument* (stąd też nazwa),
- Dokument to mapa klucz-wartość, gdzie wartość może być typu prostego (np. liczba, wartość logiczna, ciąg znaków, null) lub złożonego (np. zagnieżdżony obiekt, tablica),
- Zbiory dokumentów grupowane w *kolekcje,*
- Możliwe wyszukiwanie względem wartości

# Przykładowe struktury

## Zamówienia

```
"ID_Zamowienia": 1234
"Uzytkownik": "Alicja Zawadzka"
"Data": "2012-08-08”
"Status": "dostarczone”
"ID_Produktu": "abcd"
"Nazwa_produktu": "lody truskawkowe"
"ID_Produktu": "efab"
"Nazwa_produktu": "brukselka"
"ID_Produktu": "cdef"
"Nazwa_produktu": "brukselka"
```

```
"ID_Zamowienia": 5678
"Uzytkownik": "Alicja Zawadzka"
"Data": "2012-08-16"
"Status": "w realizacji”
"ID_Produktu": "cdef"
"Nazwa_produktu": "brukselka"
```

## Użytkownicy

```
"Uzytkownik": "Alicja Zawadzka"
"ID_Zamowienia": 1234
"Data": "2012-08-08”
"Status": "dostarczone”
"ID_Produktu": "abcd"
"Nazwa_produktu": "lody truskawkowe"
"ID_Produktu": "efab"
"Nazwa_produktu": "brukselka"
"ID_Produktu": "cdef"
"Nazwa_produktu": "brukselka"
"ID_Zamowienia": 5678
"Data": "2012-08-16"
"Status": "w realizacji”
"ID_Produktu": "cdef"
"Description": "brukselka"
```

# Baza dokumentowa - MongoDB

1. Wszystkie zamówienia Alicji Zawadzkiej:

```
db.orders.find({"Uzytkownik" : "Alicja Zawadzka"})
```

2. Ostatnie zamówienie Alicji Zawadzkiej:

```
db.orders.find({"Uzytkownik" : "Alicja Zawadzka"}).
sort({"Data" : -1}).limit(1)
```

3. Zawartość zamówienia 1234 Alicji Zawadzkiej:

```
db.orders.find({"Uzytkownik" : "Alicja Zawadzka",
"ID_Zamowienia" : 1234},
{"Produkty_na_zamowieniu" : 1})
```

```
{
  "ID_Zamowienia": 1234,
  "Uzytkownik": "Alicja Zawadzka",
  "Data": "2012-08-08",
  "Status": "dostarczone",
  "Produkty_na_zamowieniu ": [
    {
      "ID_Produktu": "abcd",
      "Nazwa_produktu": "lody truskawkowe"
    },
    {
      "ID_Produktu": "efab",
      "Nazwa_produktu": "brukselka"
    },
    {
      "ID_Produktu": "cdef",
      "Nazwa_produktu": "kawa espresso"
    }
  ]
}

{
  "ID_Zamowienia": 5678,
  "Uzytkownik": "Alicja Zawadzka",
  "Data": "2012-08-16",
  "Status": "w realizacji",
  "Produkty_na_zamowieniu": [
    {
      "ID_Produktu": "cdef",
      "Nazwa_produktu": "kawa espresso"
    }
  ]
}
```

# Baza dokumentowa - MongoDB

4. Produkt kupowany w ramach ostatnich dwóch zamówień Alicji Zawadzkiej:

```
db.orders.aggregate([
{$match : {"Uzytkownik" : "Alicja Zawadzka"}},
{$sort : {"Data" : -1}},
{$limit : 2},
{$unwind: "$Produkty_na_zamowieniu"},
{$group : {"_id” : "$Produkty_na_zamowieniu.Nazwa_produktu",
"Ilosc" : {$sum : 1}}},
{$match : {"Ilosc": {$gt : 1}}}])
```

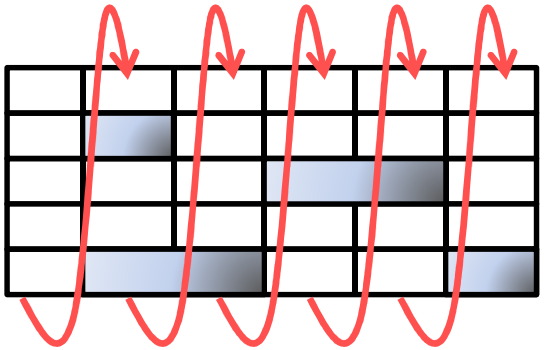
```
{
  "ID_Zamowienia": 1234,
  "Uzytkownik": "Alicja Zawadzka",
  "Data": "2012-08-08",
  "Status": "dostarczone",
  "Produkty_na_zamowieniu ": [
    {
      "ID_Produktu": "abcd",
      "Nazwa_produktu": "lody truskawkowe"
    },
    {
      "ID_Produktu": "efab",
      "Nazwa_produktu": "brukselka"
    },
    {
      "ID_Produktu": "cdef",
      "Nazwa_produktu": "kawa espresso"
    }
  ]
}

{
  "ID_Zamowienia": 5678,
  "Uzytkownik": "Alicja Zawadzka",
  "Data": "2012-08-16",
  "Status": "w realizacji",
  "Produkty_na_zamowieniu": [
    {
      "ID_Produktu": "cdef",
      "Nazwa_produktu": "kawa espresso"
    }
  ]
}
```

# Dokumentowe bazy danych – cechy

- Stosunkowo proste przejście z modelu relacyjnego
- Wyszukiwanie także względem wartości – możliwość tworzenia złożonych zapytań
- Elastyczność w strukturze dokumentu – brak sztywnego schematu
- Struktura ukierunkowana na procesy w aplikacji a nie na „uniwersalizm”

# Kolumnowe bazy danych



# Kolumnowe bazy danych

- Kolumnowe bazy danych różnią się sposobem przechowywania danych: przechowują dane w sposób ***kolumnowy***.
- Tradycyjny sposób przechowywania danych to wiersz-po-wierszu, np. Produkt_na_zamowieniu ((1234, abcd, 1), (1234, efab, 1), (1234, cdef, 1), (5678, cdef, 1))
- W bazach kolumnowych dane przechowywane są kolumna-po-kolumnie: ((1234, 1234, 1234, 5678), (abcd, efab, cdef, cdef), (1, 1, 1, 1))

# Kolumnowe bazy danych – po co?

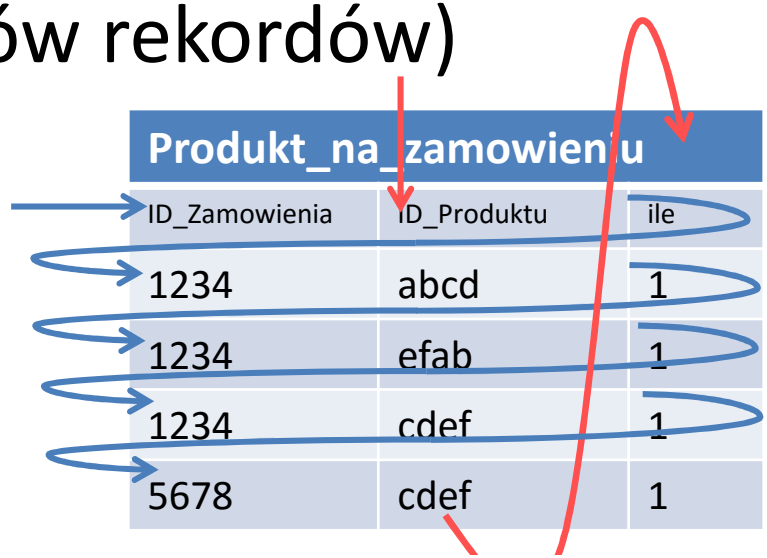
- W relacyjnej bazie (nieco upraszczając) istnieją dwa sposoby dostępu do danych:
  - przez indeks,
  - przez skanowanie.
- Większość zapytań analitycznych musi skanować całą zawartość tabeli, np.:

```
SELECT ID_Produktu, SUM(ile) FROM
Produkt_na_zamowieniu GROUP BY
ID_Produktu
```

# Kolumnowe = szybkość

```
SELECT ID_Produktu, SUM(ile) FROM
Produkt_na_zamowieniu GROUP BY
ID_Produktu
```

- Baza wierszowa musi przejrzeć całość tabeli (ze względu na specyfikę odczytu z dysku nie ma sensu „omijanie” fragmentów rekordów)
- Baza kolumnowa przejrzy tylko potrzebne kolumny



# Kolumnowe = szybkość (2)

- Zysk może wydawać się niewielki ale:
  - jeśli danych jest dużo, np. 1 mld rekordów (założenie: 10B na atrybut):
    - 30B x 1 mld / (100 MB/s) = 300 s
    - 10B x 2 x 1 mld / (100 MB/s) = 200 s
  - przeważnie kolumn jest znacznie więcej (np. w LineItem będzie też zapewne cena, wartość, uwagi itp.), co sprawia że zysk jest znacznie większy

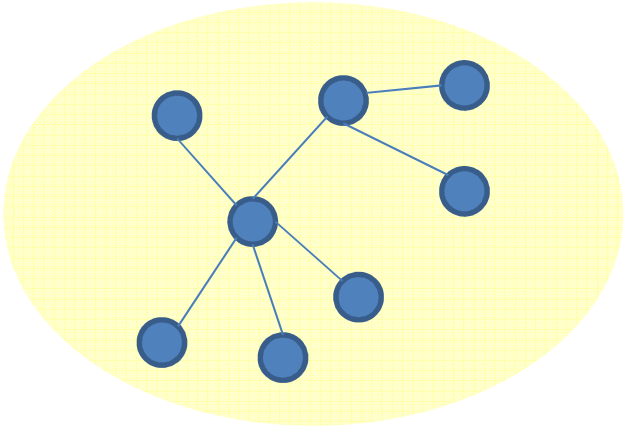
# Kolumnowe = kompresja

- Bazy kolumnowe mają znacznie większy potencjał kompresji danych, bo zazwyczaj wartości często się powtarzają (np. (1234, 1234, …, 5678))
- Dla dużych danych, gdzie wąskim gardłem staje się transfer dyskowy, kompresja pozwala osiągnąć jeszcze większe przyspieszenie

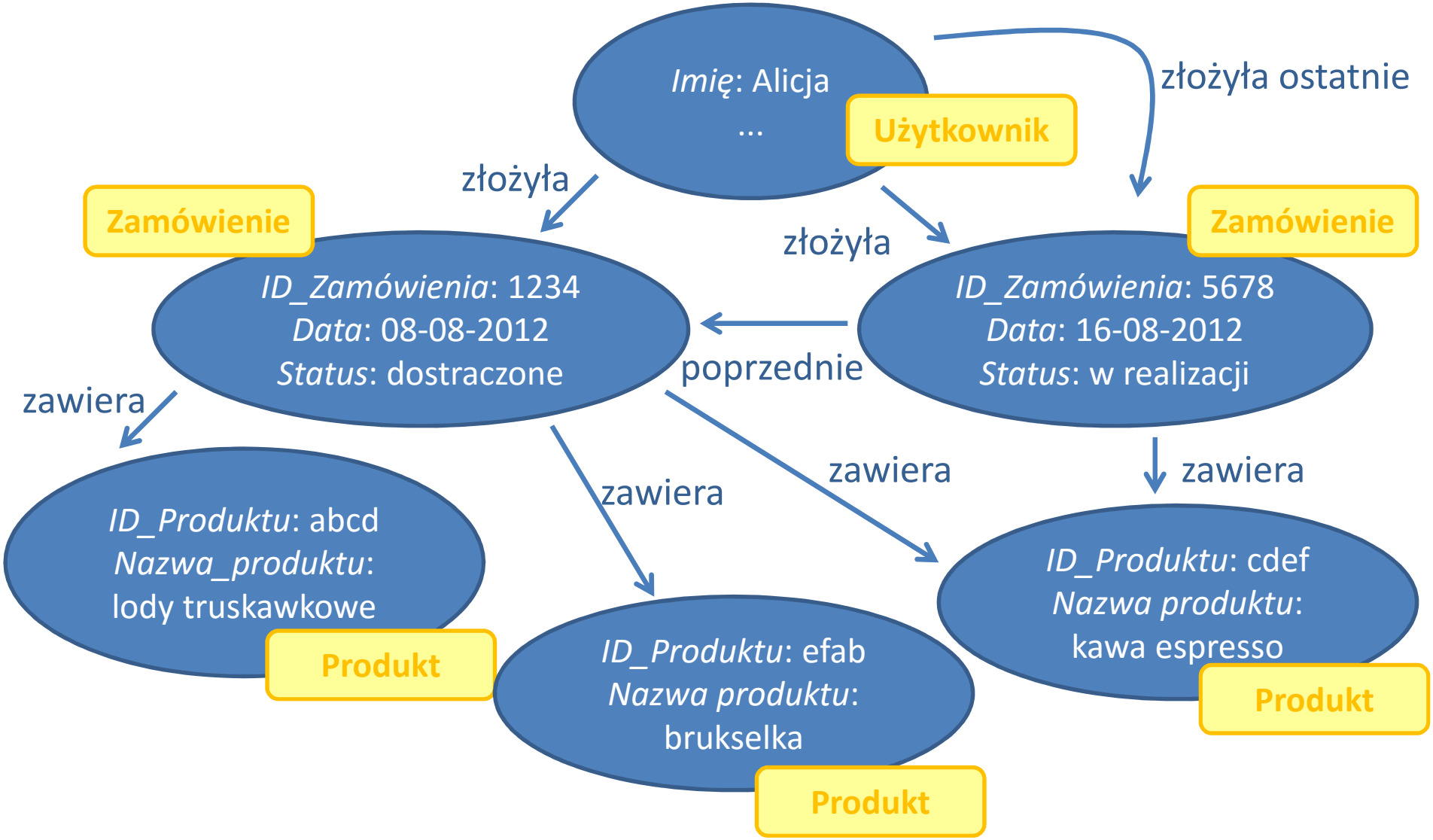
# HBase

- Baza przeznaczona do przechowywania dużych danych pozyskanych z sieci.
- Organizacja przechowywania w HBase jest kolumnowa (konkretniej HBase operuje pojęciem rodziny kolumn).
- HBase jest dobrze przystosowany do przechowywania danych o wielkiej liczbie atrybutów, ale rzadkich (tj. duża część atrybutów może być niewypełniona).
- HBase pozwala na swobodne dodanie kolumny.

# Grafowe bazy danych



# Grafowa baza danych



# Definiowanie bazy grafowej

```
CREATE (alicja:Uzytkownik {imie:’Alicja’,
       nazwisko:’Zawadzka’, email:’alzaw@gmail.com’,
       telefon:’678342267’}),
       (5678:Zamowienie {nr:’5678’,data:20120808, status:’w
       realizacji’}),
       (1234:Zamowienie {nr:’1234’data:20120816,
       status:’dostarczone’}),
       (alicja)-[:ZLOZYLA]->(5678),
       (alicja)-[:ZLOZYLA]->(1234),
       (abcd:Produkt {nazwa:’lody truskawkowe’}),
       (1234)-[:ZAWIERA {ile:1}]->(abcd),
       …
```

# Zapytania 1 i 2

**1. Podaj wszystkie zamówienia Alicji Zawadzkiej**

```
MATCH (a:Uzytkownik {name:'Alicja', nazwisko:'Zawadzka'})-
      [:ZLOZYLA]->(b)
RETURN b.nr AS zamowienie
```

**2. Podaj ostatnie zamówienie Alicji Zawadzkiej**

```
MATCH (a:Uzytkownik {name:'Alicja', nazwisko:'Zawadzka'})-
      [:ZLOZYLA_OSTATNIE]->(b)
RETURN b.nr AS zamowienie
```

# Zapytania 3 i 4

**3. Co zostało zamówione przez Alicję Zawadzką w ramach zamówienia nr 1234?**

```
MATCH (a:Uzytkownik {name:'Alicja', nazwisko:'Zawadzka'})-
      [:ZLOZYLA]->(b:Zamowienie {nr:'1234'}),
      (b)-[:ZAWIERA]->(c)
RETURN c.nazwa_produktu AS produkt
```

**4. Podaj, czy jest (jeżeli tak to jaki) produkt, który został zamówiony w ostatnim i przedostatnim zamówieniu Alicji Zawadzkiej**

```
MATCH (a:Uzytkownik {name:'Alicja', nazwisko:'Zawadzka'})-
      [:ZLOZYLA_OSTATNIE]->(ostatnieZam),
      (ostatnieZam)-[:ZAWIERA]->(c),
      (ostatnieZam)-[:POPRZEDNIE]->(przedostatnieZam),
      (przedostatnieZam)-[ZAWIERA]->(c)
RETURN c.nazwa_produktu AS produkt
```

# Grafowe bazy danych - cechy

- Prawie wszystko może być zamodelowane jako graf.
- Elastyczny model danych, grafy są addytywne z natury.
- Duża skalowalność – zapytania dotyczą fragmentu grafu, a nie jego całości.
- Proste zapytania

# Zagadnienia na egzamin

1. Cechy baz danych typu NoSQL
2. Teoria CAP
3. Skalowanie poziome i pionowe
4. Metody rozpraszania baz danych
5. BASE vs. ACID
6. Typy baz danych typu NoSQL i ich charakterystyki.