

# SQL - zapytania złożone

(na podstawie slajdów Przemysławy Kanarek)

# Podzapytania

# Podzapytania

SELECT A,B FROM R, S WHERE .....

# Podzapytania

SELECT A,B FROM R, S WHERE .....

SELECT A,B FROM (SELECT...) X, S WHERE .....

# Podzapytania w klauzuli WHERE

SELECT A,B FROM R, S WHERE .....

W klauzuli WHERE możemy stosować operatory, których argumentem jest zbiór - może to być zbiór stały (jawnie wymienione wartości) lub wynik (pod)zapytania SELECT.

# Podzapytania w klauzuli WHERE

SELECT A,B FROM R, S WHERE .....

W klauzuli WHERE możemy stosować operatory, których argumentem jest zbiór - może to być zbiór stały (jawnie wymienione wartości) lub wynik (pod)zapytania SELECT.

- wartość > ANY (SELECT... FROM ... WHERE ...)
- wartość > SOME (SELECT... FROM ... WHERE ...)
- wartość > ALL (SELECT... FROM ... WHERE ...)
- wartość IN (SELECT... FROM ... WHERE ...)
- EXISTS (SELECT \* FROM ... WHERE ...)

# Podzapytania w klauzuli WHERE

SELECT A,B FROM R, S WHERE .....

W klauzuli WHERE możemy stosować **operatory**, których argumentem jest zbiór - może to być zbiór stały (jawnie wymienione wartości) lub wynik (pod)zapytania SELECT.

- wartość > **ANY** (SELECT... FROM ... WHERE ...)
- wartość > **SOME** (SELECT... FROM ... WHERE ...)
- wartość > **ALL** (SELECT... FROM ... WHERE ...)
- wartość **IN** (SELECT... FROM ... WHERE ...)
- **EXISTS** (SELECT \* FROM ... WHERE ...)

# Podzapytania w klauzuli WHERE

SELECT A,B FROM R, S WHERE .....

W klauzuli WHERE możemy stosować **operatory**, których argumentem jest zbiór - może to być zbiór stały (jawnie wymienione wartości) lub wynik (pod)zapytania SELECT.

- wartość > **ANY** (SELECT... FROM ... WHERE ...)
- wartość > **SOME** (SELECT... FROM ... WHERE ...)
- wartość > **ALL** (SELECT... FROM ... WHERE ...)
- wartość **IN** (SELECT... FROM ... WHERE ...)
- **EXISTS** (SELECT \* FROM ... WHERE ...)



# Podzapytania w klauzuli WHERE

SELECT A,B FROM R, S WHERE .....

W klauzuli WHERE możemy stosować **operatory**, których argumentem jest **zbiór** - może to być zbiór stały (jawnie wymienione wartości) lub wynik (pod)zapytania SELECT.

- wartość > ANY (SELECT... FROM ... WHERE ...)
- wartość > SOME (SELECT... FROM ... WHERE ...)
- wartość > ALL (SELECT... FROM ... WHERE ...)
- wartość IN (SELECT... FROM ... WHERE ...)
- EXISTS (SELECT \* FROM ... WHERE ...)

# Podzapytania w klauzuli WHERE

SELECT A,B FROM R, S WHERE .....

W klauzuli WHERE możemy stosować **operatory**, których argumentem jest **zbiór** - może to być zbiór stały (jawnie wymienione wartości) lub wynik (pod)zapytania SELECT.

- wartość < ANY (SELECT... FROM ... WHERE ...)
- wartość >= SOME (SELECT... FROM ... WHERE ...)
- wartość != ALL (SELECT... FROM ... WHERE ...)
- wartość NOT IN (SELECT... FROM ... WHERE ...)
- NOT EXISTS (SELECT \* FROM ... WHERE ...)

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż jakiś wynik Anny Abackiej

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż jakiś wynik Anny Abackiej

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ... ;
```

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ... ;
```

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (wyniki Anny Abackiej);
```

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś** **wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (wyniki Anny Abackiej);
```

```
SELECT wynik FROM Wynik JOIN Osoba ON id=osoba  
WHERE nazwisko='Abacka' AND imie='Anna';
```

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś** **wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (
```

```
SELECT wynik FROM Wynik JOIN Osoba ON id=osoba  
WHERE nazwisko='Abacka' AND imie='Anna';
```



# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś** **wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba
WHERE wynik > ANY (
  SELECT wynik FROM Wynik JOIN Osoba ON id=osoba
  WHERE nazwisko='Abacka' AND imie='Anna';
```

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś** wynik Anny Abackiej

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (SELECT wynik  
FROM Wynik JOIN Osoba ON id=osoba  
WHERE nazwisko='Abacka' AND imie='Anna');
```

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś** **wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (SELECT wynik  
FROM Wynik JOIN Osoba ON id=osoba  
WHERE nazwisko='Abacka' AND imie='Anna');
```

*Podzapytanie jest wyliczane "w kontekście" zapytania, ale użyte w nim nazwy (relacje, zmienne, atrybuty) przysłaniają nazwy z kontekstu zapytania.*

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (SELECT wynik  
FROM Wynik JOIN Osoba ON id=osoba  
WHERE nazwisko='Abacka' AND imie='Anna');
```

*Podzapytanie jest wyliczane "w kontekście" zapytania, ale użyte w nim nazwy (relacje, zmienne, atrybuty) przysłaniają nazwy z kontekstu zapytania.*

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (SELECT wynik  
FROM Wynik JOIN Osoba ON id=osoba  
WHERE nazwisko='Abacka' AND imie='Anna');
```

*Podzapytanie jest wyliczane "w kontekście" zapytania, ale użyte w nim nazwy (relacje, zmienne, atrybuty) przysłaniają nazwy z kontekstu zapytania.*

*To wszystko?*

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (SELECT wynik  
FROM Wynik JOIN Osoba ON id=osoba  
WHERE nazwisko='Abacka' AND imie='Anna');
```

*Podzapytanie jest wyliczane "w kontekście" zapytania, ale użyte w nim nazwy (relacje, zmienne, atrybuty) przysłaniają nazwy z kontekstu zapytania.*

*To wszystko?*

*Pytalismy o osoby, a nie o krotki złączenia Osoba i Wynik!*

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś wynik Anny Abackiej**

```
SELECT * FROM Osoba JOIN Wynik ON id=osoba
WHERE wynik > ANY (SELECT wynik
FROM Wynik JOIN Osoba ON id=osoba
WHERE nazwisko='Abacka' AND imie='Anna');
```

*Podzapytanie jest wyliczane "w kontekście" zapytania, ale użyte w nim nazwy (relacje, zmienne, atrybuty) przysłaniają nazwy z kontekstu zapytania.*

*To wszystko?*

*Pytalismy o osoby, a nie o krotki złączenia Osoba i Wynik!*

# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś wynik Anny Abackiej**

```
SELECT Osoby.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (SELECT wynik  
FROM Wynik JOIN Osoba ON id=osoba  
WHERE nazwisko='Abacka' AND imie='Anna');
```

*Podzapytanie jest wyliczane "w kontekście" zapytania, ale użyte w nim nazwy (relacje, zmienne, atrybuty) przysłaniają nazwy z kontekstu zapytania.*

*To wszystko?*

*Pytalismy o osoby, a nie o krotki złączenia Osoba i Wynik!*



# Podzapytania w klauzuli WHERE - przykład 1

Osoby mające jakiś wynik lepszy niż **jakiś wynik Anny Abackiej**

```
SELECT Osoby.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ANY (SELECT wynik  
FROM Wynik JOIN Osoba ON id=osoba  
WHERE nazwisko='Abacka' AND imie='Anna');
```

*Podzapytanie jest wyliczane "w kontekście" zapytania, ale użyte w nim nazwy (relacje, zmienne, atrybuty) przysłaniają nazwy z kontekstu zapytania.*

*To wszystko?*

*Pytalismy o osoby, a nie o krotki złączenia Osoba i Wynik!*

*Teraz to wszystko*

## Podzapytania w klauzuli WHERE - przykład 2

Osoby mające jakiś wynik lepszy niż wszystkie wyniki Anny Abackiej

## Podzapytania w klauzuli WHERE - przykład 2

Osoby mające jakiś wynik lepszy niż **wszystkie** wyniki Anny Abackiej

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ALL (...);
```

## Podzapytania w klauzuli WHERE - przykład 2

Osoby mające jakiś wynik lepszy niż **wszystkie** wyniki Anny Abackiej

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE wynik > ALL (SELECT wynik  
                FROM Wynik JOIN Osoba ON id=osoba  
                WHERE nazwisko='Abacka' AND imie='Anna');
```

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik  
*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy, niż ich wszystkie wyniki za to zadanie w ubiegłych latach.

## Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik

*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy, niż ich wszystkie wyniki za to zadanie w ubiegłych latach.

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>
```

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik

*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy, niż

ich wszystkie wyniki za to zadanie w ubiegłych latach.

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>
```



## Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik

*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy, niż

ich wszystkie wyniki za to zadanie w ubiegłych latach.

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>
  ALL(wyniki tej osoby dla tego zadania w ubiegłych latach);
```

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik  
*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy, niż  
ich wszystkie wyniki za to zadanie w ubiegłych latach.

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>  
    ALL(wyniki tej osoby dla tego zadania w ubiegłych latach);
```

```
SELECT wynik FROM Wynik  
WHERE osoba=? AND zad=?  
    AND EXTRACT(year FROM czasWyk)<2020;
```

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik  
*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy,  
niż **ich wszystkie wyniki za to zadanie w ubiegłych latach.**

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>
  ALL(wyniki tej osoby dla tego zadania w ubiegłych latach);
```

```
SELECT wynik FROM Wynik
WHERE osoba=? AND zad=?
  AND EXTRACT(year FROM czasWyk)<2020;
```

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik  
*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy,  
niż **ich wszystkie wyniki za to zadanie w ubiegłych latach.**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>  
    ALL(wyniki tej osoby dla tego zadania w ubiegłych latach);
```

```
SELECT wynik FROM Wynik  
WHERE osoba=w1.osoba AND zad=w1.zad  
    AND EXTRACT(year FROM czasWyk)<2020;
```

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik  
*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy,  
niż **ich wszystkie wyniki za to zadanie w ubiegłych latach.**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>  
ALL(
```

```
SELECT wynik FROM Wynik  
WHERE osoba=w1.osoba AND zad=w1.zad  
AND EXTRACT(year FROM czasWyk)<2020;
```

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik  
*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy,  
niż **ich wszystkie wyniki za to zadanie w ubiegłych latach.**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>
  ALL(
SELECT wynik FROM Wynik
WHERE osoba=w1.osoba AND zad=w1.zad
AND EXTRACT(year FROM czasWyk)<2020);
```

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik  
*czyli*

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy,  
niż **ich wszystkie wyniki za to zadanie w ubiegłych latach.**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>
  ALL(SELECT wynik FROM Wynik
       WHERE osoba=w1.osoba AND zad=w1.zad
       AND EXTRACT(year FROM czasWyk)<2020);
```

# Podzapytania w klauzuli WHERE - przykład 3

Osoby, które od zeszłego roku poprawiły jakiś swój wynik  
czyli

Osoby, które mają jakiś wynik za zadanie zrobiony w 2020 roku lepszy, niż  
ich wszystkie wyniki za to zadanie w ubiegłych latach.

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba
WHERE EXTRACT(year FROM czasWyk)=2020 AND wynik>
  ALL(SELECT wynik FROM Wynik
       WHERE osoba=w1.osoba AND zad=w1.zad
       AND EXTRACT(year FROM czasWyk)<2020);
```

*W tym przykładzie mamy przykład podzapytania zależnego - w jego treści pojawiają się odwołania do krotki, dla której jest obliczane i musi być obliczane wielokrotnie.*



## Podzapytania w klauzuli WHERE - przykład 4

# Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD  
występujące również w tabeli Klas

## Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD  
występujące również w tabeli Klas

```
SELECT * FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (...)
```

## Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś **zadanie z kategorii BD**  
**występujące również w tabeli Klas**

```
SELECT * FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (...)
```

## Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś **zadanie z kategorii BD**  
**występujące również w tabeli Klas**

```
SELECT * FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (...)
```

```
SELECT Zad.id  
FROM Zad JOIN Klas ON id=zad JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD'
```

# Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT * FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (
```

```
SELECT Zad.id  
FROM Zad JOIN Klas ON id=zad JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD');
```

## Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT * FROM Osoba JOIN Wynik ON osoba=id
WHERE wynik=100 AND zad IN (
SELECT Zad.id
FROM Zad JOIN Klas ON id=zad JOIN Kat ON Kat.id=Klas.kat
WHERE Kat.nazwa='BD');
```

## Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT * FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (SELECT Zad.id  
FROM Zad JOIN Klas ON id=zad JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD');
```



## Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT * FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (SELECT Zad.id  
FROM Zad JOIN Klas ON id=zad JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD');
```

*To wszystko?*

## Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (SELECT Zad.id  
FROM Zad JOIN Klas ON id=zad JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD');
```

*To wszystko?*

# Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (SELECT Zad.id  
FROM Zad JOIN Klas ON id=zad JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD');
```

*To wszystko?*

*To wszystko?*

# Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (SELECT Zad.id  
FROM Zad JOIN Klas ON id=zad JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD');
```

*To wszystko?*

*To wszystko?*

*Łączenie z tabelą Zad w podzapytaniu jest zbędne!*

# Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (SELECT Zad.id  
FROM Zad JOIN Klas ON id=zad JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD');
```

*To wszystko?*

*To wszystko?*

*Łączenie z tabelą Zad w podzapytaniu jest zbędne!*

# Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (SELECT Zad.id  
FROM Klas JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD');
```

*To wszystko?*

*To wszystko?*

*Łączenie z tabelą Zad w podzapytaniu jest zbędne!*

# Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON osoba=id
WHERE wynik=100 AND zad IN (SELECT Zad.id
FROM Klas JOIN Kat ON Kat.id=Klas.kat
WHERE Kat.nazwa='BD');
```

*To wszystko?*

*To wszystko?*

*Łączenie z tabelą Zad w podzapytaniu jest zbędne!*

# Podzapytania w klauzuli WHERE - przykład 4

Osoby, które uzyskały wynik 100 za jakieś zadanie z kategorii BD występujące również w tabeli Klas

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON osoba=id  
WHERE wynik=100 AND zad IN (SELECT Klas.zad  
FROM Klas JOIN Kat ON Kat.id=Klas.kat  
WHERE Kat.nazwa='BD');
```

*To wszystko?*

*To wszystko?*

*Łączenie z tabelą Zad w podzapytaniu jest zbędne!*



# Podzapytania w klauzuli WHERE - przykład 5

# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie.

# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
*czyli*  
istnieje ich gorszy wynik za to samo zadanie z ubiegłych lat

## Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
*czyli*

istnieje ich gorszy wynik za to samo zadanie z ubiegłych lat

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(...)
```

## Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
*czyli*

istnieje **ich gorszy wynik za to samo zadanie z ubiegłych lat**

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(...)
```

# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
*czyli*

istnieje **ich gorszy wynik za to samo zadanie z ubiegłych lat**

```
SELECT Osoba.* FROM Osoba JOIN Wynik ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(...)
```

```
SELECT * FROM Wynik WHERE  
EXTRACT(year FROM czasWyk)<2020 AND  
osoba=osoba AND zad=zad AND wynik<wynik
```

# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
*czyli*

istnieje **ich gorszy wynik za to samo zadanie z ubiegłych lat**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(...)
```

```
SELECT * FROM Wynik WHERE  
EXTRACT(year FROM czasWyk)<2020 AND  
osoba=w1.osoba AND zad=w1.zad AND wynik<w1.wynik
```

# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
*czyli*

istnieje **ich gorszy wynik za to samo zadanie z ubiegłych lat**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(...)
```

```
SELECT * FROM Wynik WHERE  
EXTRACT(year FROM czasWyk)<2020 AND  
osoba=w1.osoba AND zad=w1.zad AND wynik<w1.wynik
```



# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie.

*czyli*

istnieje **ich gorszy wynik za to samo zadanie z ubiegłych lat**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(...)
```

```
SELECT * FROM Wynik WHERE  
EXTRACT(year FROM czasWyk)<2020 AND  
osoba=w1.osoba AND zad=w1.zad AND wynik<w1.wynik
```

# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
*czyli*

istnieje **ich gorszy wynik za to samo zadanie z ubiegłych lat**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(...)
```

```
SELECT * FROM Wynik w2 WHERE  
EXTRACT(year FROM czasWyk)<2020 AND  
w2.osoba=w1.osoba AND w2.zad=w1.zad AND w2.wynik<w1.wynik
```

# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
*czyli*

istnieje **ich gorszy wynik za to samo zadanie z ubiegłych lat**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba  
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(
```

```
SELECT * FROM Wynik w2 WHERE  
EXTRACT(year FROM czasWyk)<2020 AND  
w2.osoba=w1.osoba AND w2.zad=w1.zad AND w2.wynik<w1.wynik);
```

# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
*czyli*

istnieje **ich gorszy wynik za to samo zadanie z ubiegłych lat**

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(
SELECT * FROM Wynik w2 WHERE
EXTRACT(year FROM czasWyk)<2020 AND
w2.osoba=w1.osoba AND w2.zad=w1.zad AND w2.wynik<w1.wynik);
```

# Podzapytania w klauzuli WHERE - przykład 5

Osoby, które od zeszłego roku poprawiły swój wynik za jakieś zadanie  
czyli

istnieje ich gorszy wynik za to samo zadanie z ubiegłych lat

```
SELECT Osoba.* FROM Osoba JOIN Wynik w1 ON id=osoba
WHERE EXTRACT(year FROM czasWyk)=2020 AND EXISTS(
SELECT * FROM Wynik w2 WHERE
EXTRACT(year FROM czasWyk)<2020 AND
w2.osoba=w1.osoba AND w2.zad=w1.zad AND w2.wynik<w1.wynik);
```

*W podzapytaniu nie ma sensu bawić się w sortowanie, usuwanie duplikatów (ew. w rzutowanie przy EXISTS). To zbędna praca.*

# Alternatywne postaci zapytań

# Alternatywne postaci zapytań

To samo pytanie można zadać na kilka istotnie różnych sposobów. Posłużymy się teraz przykładami z bazy zapisów. Osoby, które chodziły na wykład z Baz Danych można znaleźć:

- wybierając tych, którzy łączą się z wpisem do jakiejś grupy wykładowej z Baz danych (JOIN);
- wybierając osoby, których kody są w zbiorze tych, które dotyczą wyboru grupy wykładowej... (IN);
- wybierając osoby, dla których istnieje wybór, dla którego istnieje grupa, dla której istnieje.... (EXISTS)

# Alternatywne postaci zapytań - przykład

Wyberzmy nazwiska osób, które zapisały się (kiedykolwiek) na wykład z Bazy danych - zastosujemy JOIN

```
SELECT DISTINCT nazwisko FROM
  uzytkownik
  JOIN wybor USING(kod_uz)
  JOIN grupa USING(kod_grupy)
  JOIN przedmiot_semestr USING(kod_przed_sem)
  JOIN przedmiot using(kod_przed)
WHERE rodzaj_zajec='w' AND nazwa='Bazy danych';
```



# Alternatywne postaci zapytań - przykład

Wyberzmy nazwiska osób, które zapisały się (kiedykolwiek) na wykład z Baz danych - zastosujmy IN

```
SELECT DISTINCT nazwisko FROM uzytkownik
WHERE kod_uz IN
  (SELECT kod_uz FROM wybor
   WHERE kod_grupy IN
    (SELECT kod_grupy FROM grupa
     WHERE rodzaj_zajec='w' AND kod_przed_sem IN
      (SELECT kod_przed_sem FROM przedmiot_semestr
       WHERE kod_przed IN
        (SELECT kod_przed FROM przedmiot
         WHERE nazwa='Bazy danych')))));
```

# Alternatywne postaci zapytań - przykład

Wyberzmy nazwiska osób, które zapisały się (kiedykolwiek) na wykład z Baz danych - zastosujmy EXISTS

```
SELECT DISTINCT nazwisko FROM uzytkownik u WHERE
EXISTS(SELECT * FROM wybor w WHERE u.kod_uz=w.kod_uz AND
EXISTS (SELECT * FROM grupa g WHERE rodzaj_zajec='w'
AND g.kod_grupy=w.kod_grupy AND
EXISTS (SELECT * FROM przedmiot_semestr ps
WHERE ps.kod_przed_sem=g.kod_przed_sem AND
EXISTS (SELECT * FROM przedmiot p WHERE
p.nazwa='Bazy danych' AND
p.kod_przed=ps.kod_przed))));
```

# Alternatywne postaci zapytań - przykład

Aby porównać działanie tych wersji możemy je wykonać poprzedzając poleceniem EXPLAIN lub EXPLAIN ANALYZE:

```
explain analyze select nazwisko from uzytkownik join wybor using(kod_uz)  
join ....
```

# Alternatywne postaci zapytań - przykład

Aby porównać działanie tych wersji możemy je wykonać poprzedzając poleceniem EXPLAIN lub EXPLAIN ANALYZE:

## QUERY PLAN

```
-----
Unique (cost=53.41..53.46 rows=9 width=11) (actual time=2.945..3.114 rows=254 loops=1)
-> Sort (cost=53.41..53.44 rows=9 width=11) (actual time=2.944..2.984 rows=306 loops=1)
    Sort Key: uzytkownik.nazwisko
    Sort Method: quicksort  Memory: 39kB
-> Nested Loop (cost=22.41..53.27 rows=9 width=11) (actual time=0.375..1.925 rows=306 loops=1)
-> Nested Loop (cost=22.13..50.59 rows=9 width=4) (actual time=0.368..1.012 rows=306 loops=1)
    -> Hash Join (cost=21.84..47.24 rows=1 width=4) (actual time=0.353..0.650 rows=3 loops=1)
        Hash Cond: (grupa.kod_przed_sem = przedmiot_semestr.kod_przed_sem)
    -> Seq Scan on grupa (cost=0.00..24.30 rows=289 width=8) (actual time=0.017..0.298 rows=289 loops=1)
        Filter: (rodzaj_zajec = 'w'::bpchar)
        Rows Removed by Filter: 855
-> Hash (cost=21.83..21.83 rows=1 width=4) (actual time=0.291..0.291 rows=3 loops=1)
    Buckets: 1024  Batches: 1  Memory Usage: 9kB
    -> Hash Join (cost=13.18..21.83 rows=1 width=4) (actual time=0.141..0.289 rows=3 loops=1)
        Hash Cond: (przedmiot_semestr.kod_przed = przedmiot.kod_przed)
    -> Seq Scan on przedmiot_semestr (cost=0.00..7.47 rows=447 width=8) (actual time=0.010..0.078 rows=447
loops=1)
    -> Hash (cost=13.16..13.16 rows=1 width=4) (actual time=0.105..0.106 rows=1 loops=1)
        Buckets: 1024  Batches: 1  Memory Usage: 9kB
```

# Alternatywne postaci zapytań - przykład

Aby porównać działanie tych wersji możemy je wykonać poprzedzając poleceniem EXPLAIN lub EXPLAIN ANALYZE:

## QUERY PLAN

```
-----  
-> Sort (cost=53.41..53.44 rows=9 width=11) (actual time=2.944..2.984 rows=306 loops=1)  
    Sort Key: uzytkownik.nazwisko  
    Sort Method: quicksort  Memory: 39kB  
-> Nested Loop (cost=22.41..53.27 rows=9 width=11) (actual time=0.375..1.925 rows=306 loops=1)  
-> Nested Loop (cost=22.13..50.59 rows=9 width=4) (actual time=0.368..1.012 rows=306 loops=1)  
    -> Hash Join (cost=21.84..47.24 rows=1 width=4) (actual time=0.353..0.650 rows=3 loops=1)  
        Hash Cond: (grupa.kod_przed_sem = przedmiot_semestr.kod_przed_sem)  
-> Seq Scan on grupa (cost=0.00..24.30 rows=289 width=8) (actual time=0.017..0.298 rows=289 loops=1)  
    Filter: (rodzaj_zajec = 'w':bpchar)  
    Rows Removed by Filter: 855  
-> Hash (cost=21.83..21.83 rows=1 width=4) (actual time=0.291..0.291 rows=3 loops=1)  
    Buckets: 1024  Batches: 1  Memory Usage: 9kB  
-> Hash Join (cost=13.18..21.83 rows=1 width=4) (actual time=0.141..0.289 rows=3 loops=1)  
    Hash Cond: (przedmiot_semestr.kod_przed = przedmiot.kod_przed)  
-> Seq Scan on przedmiot_semestr (cost=0.00..7.47 rows=447 width=8) (actual time=0.010..0.078 rows=447  
loops=1)  
-> Hash (cost=13.16..13.16 rows=1 width=4) (actual time=0.105..0.106 rows=1 loops=1)  
    Buckets: 1024  Batches: 1  Memory Usage: 9kB  
-> Seq Scan on przedmiot (cost=0.00..13.16 rows=1 width=4) (actual time=0.014..0.102 rows=1 loops=1)
```

# Alternatywne postaci zapytań - przykład

Aby porównać działanie tych wersji możemy je wykonać poprzedzając poleceniem EXPLAIN lub EXPLAIN ANALYZE:

## QUERY PLAN

```
-----
-> Nested Loop (cost=22.41..53.27 rows=9 width=11) (actual time=0.375..1.925 rows=306 loops=1)
-> Nested Loop (cost=22.13..50.59 rows=9 width=4) (actual time=0.368..1.012 rows=306 loops=1)
    -> Hash Join (cost=21.84..47.24 rows=1 width=4) (actual time=0.353..0.650 rows=3 loops=1)
        Hash Cond: (grupa.kod_przed_sem = przedmiot_semestr.kod_przed_sem)
    -> Seq Scan on grupa (cost=0.00..24.30 rows=289 width=8) (actual time=0.017..0.298 rows=289 loops=1)
        Filter: (rodzaj_zajec = 'w'::bpchar)
        Rows Removed by Filter: 855
    -> Hash (cost=21.83..21.83 rows=1 width=4) (actual time=0.291..0.291 rows=3 loops=1)
        Buckets: 1024 Batches: 1 Memory Usage: 9kB
    -> Hash Join (cost=13.18..21.83 rows=1 width=4) (actual time=0.141..0.289 rows=3 loops=1)
        Hash Cond: (przedmiot_semestr.kod_przed = przedmiot.kod_przed)
    -> Seq Scan on przedmiot_semestr (cost=0.00..7.47 rows=447 width=8) (actual time=0.010..0.078 rows=447
loops=1)

    -> Hash (cost=13.16..13.16 rows=1 width=4) (actual time=0.105..0.106 rows=1 loops=1)
        Buckets: 1024 Batches: 1 Memory Usage: 9kB
    -> Seq Scan on przedmiot (cost=0.00..13.16 rows=1 width=4) (actual time=0.014..0.102 rows=1 loops=1)
        Filter: (nazwa = 'Bazy danych'::text)
        Rows Removed by Filter: 572
-> Index Only Scan using wybor_key on wybor (cost=0.29..3.12 rows=24 width=8) (actual time=0.012..0.097 rows=102
```

# Alternatywne postaci zapytań - przykład

Aby porównać działanie tych wersji możemy je wykonać poprzedzając poleceniem EXPLAIN lub EXPLAIN ANALYZE:

## QUERY PLAN

```
-----
-> Hash Join (cost=21.84..47.24 rows=1 width=4) (actual time=0.353..0.650 rows=3 loops=1)
   Hash Cond: (grupa.kod_przed_sem = przedmiot_semestr.kod_przed_sem)
-> Seq Scan on grupa (cost=0.00..24.30 rows=289 width=8) (actual time=0.017..0.298 rows=289 loops=1)
   Filter: (rodzaj_zajec = 'w'::bpchar)
   Rows Removed by Filter: 855
-> Hash (cost=21.83..21.83 rows=1 width=4) (actual time=0.291..0.291 rows=3 loops=1)
   Buckets: 1024 Batches: 1 Memory Usage: 9kB
   -> Hash Join (cost=13.18..21.83 rows=1 width=4) (actual time=0.141..0.289 rows=3 loops=1)
      Hash Cond: (przedmiot_semestr.kod_przed = przedmiot.kod_przed)
      -> Seq Scan on przedmiot_semestr (cost=0.00..7.47 rows=447 width=8) (actual time=0.010..0.078 rows=447
loops=1)
         -> Hash (cost=13.16..13.16 rows=1 width=4) (actual time=0.105..0.106 rows=1 loops=1)
            Buckets: 1024 Batches: 1 Memory Usage: 9kB
            -> Seq Scan on przedmiot (cost=0.00..13.16 rows=1 width=4) (actual time=0.014..0.102 rows=1 loops=1)
               Filter: (nazwa = 'Bazy danych'::text)
               Rows Removed by Filter: 572
-> Index Only Scan using wybor_key on wybor (cost=0.29..3.12 rows=24 width=8) (actual time=0.012..0.097 rows=102
loops=3)
   Index Cond: (kod_grupy = grupa.kod_grupy)
```

# Alternatywne postaci zapytań - przykład

Aby porównać działanie tych wersji możemy je wykonać poprzedzając poleceniem EXPLAIN lub EXPLAIN ANALYZE:

## QUERY PLAN

```
-----
-> Seq Scan on grupa (cost=0.00..24.30 rows=289 width=8) (actual time=0.017..0.298 rows=289 loops=1)
    Filter: (rodzaj_zajec = 'w'::bpchar)
    Rows Removed by Filter: 855
-> Hash (cost=21.83..21.83 rows=1 width=4) (actual time=0.291..0.291 rows=3 loops=1)
    Buckets: 1024 Batches: 1 Memory Usage: 9kB
    -> Hash Join (cost=13.18..21.83 rows=1 width=4) (actual time=0.141..0.289 rows=3 loops=1)
        Hash Cond: (przedmiot_semestr.kod_przed = przedmiot.kod_przed)
        -> Seq Scan on przedmiot_semestr (cost=0.00..7.47 rows=447 width=8) (actual time=0.010..0.078 rows=447
loops=1)
            -> Hash (cost=13.16..13.16 rows=1 width=4) (actual time=0.105..0.106 rows=1 loops=1)
                Buckets: 1024 Batches: 1 Memory Usage: 9kB
                -> Seq Scan on przedmiot (cost=0.00..13.16 rows=1 width=4) (actual time=0.014..0.102 rows=1 loops=1)
                    Filter: (nazwa = 'Bazy danych'::text)
                    Rows Removed by Filter: 572
            -> Index Only Scan using wybor_key on wybor (cost=0.29..3.12 rows=24 width=8) (actual time=0.012..0.097 rows=102
loops=3)
                Index Cond: (kod_grupy = grupa.kod_grupy)
                Heap Fetches: 306
-> Index Scan using uzytkownik_key on uzytkownik (cost=0.28..0.30 rows=1 width=15) (actual time=0.002..0.002 rows=1
```



# Alternatywne postaci zapytań - przykład

Aby porównać działanie tych wersji możemy je wykonać poprzedzając poleceniem EXPLAIN lub EXPLAIN ANALYZE:

QUERY PLAN

-----  
-> Index Only Scan using wybor\_key on wybor (cost=0.29..3.12 rows=24 width=8) (actual time=0.012..0.097 rows=102 loops=3)

Index Cond: (kod\_grupy = grupa.kod\_grupy)

Heap Fetches: 306

-> Index Scan using uzytkownik\_key on uzytkownik (cost=0.28..0.30 rows=1 width=15) (actual time=0.002..0.002 rows=1 loops=306)

Index Cond: (kod\_uz = wybor.kod\_uz)

Planning Time: 1.020 ms

Execution Time: 3.205 ms

(28 rows)

# Alternatywne postaci zapytań - przykład

Aby porównać działanie tych wersji możemy je wykonać poprzedzając poleceniem EXPLAIN lub EXPLAIN ANALYZE:

QUERY PLAN

-----  
-> Index Only Scan using wybor\_key on wybor (cost=0.29..3.12 rows=24 width=8) (actual time=0.012..0.097 rows=102 loops=3)

Index Cond: (kod\_grupy = grupa.kod\_grupy)

Heap Fetches: 306

-> Index Scan using uzytkownik\_key on uzytkownik (cost=0.28..0.30 rows=1 width=15) (actual time=0.002..0.002 rows=1 loops=306)

Index Cond: (kod\_uz = wybor.kod\_uz)

Planning Time: 1.020 ms

**Execution Time: 3.205ms**

(28 rows)

# Alternatywne postaci zapytań - przykład

- JOIN -
- IN -
- EXISTS -

# Alternatywne postaci zapytań - przykład

- JOIN - Execution Time: 3.205 ms
- IN -
- EXISTS -

# Alternatywne postaci zapytań - przykład

- JOIN - Execution Time: 3.205 ms
- IN - Execution Time: 5.233 ms
- EXISTS -

# Alternatywne postaci zapytań - przykład

- JOIN - Execution Time: 3.205 ms
- IN - Execution Time: 5.233 ms
- EXISTS - Execution Time: 1.659 ms

## Alternatywne postaci zapytań - przykład 2

Zapytanie z negacją ma inne możliwe postaci. Pytając o osoby, które nigdy na wykład z BD nie chodziły, możemy:

## Alternatywne postaci zapytań - przykład 2

Zapytanie z negacją ma inne możliwe postaci. Pytając o osoby, które nigdy na wykład z BD nie chodziły, możemy:

- wybrać osoby, które nie należą (NOT IN) do grupy uczestników wykładów z Baz danych;



## Alternatywne postaci zapytań - przykład 2

Zapytanie z negacją ma inne możliwe postaci. Pytając o osoby, które nigdy na wykład z BD nie chodziły, możemy:

- wybrać osoby, które nie należą (NOT IN) do grupy uczestników wykładów z Baz danych;
- wybrać osoby, dla których nie istnieje (NOT EXISTS) wpis na wykład z Baz danych;

## Alternatywne postaci zapytań - przykład 2

Zapytanie z negacją ma inne możliwe postaci. Pytając o osoby, które nigdy na wykład z BD nie chodziły, możemy:

- wybrać osoby, które nie należą (NOT IN) do grupy uczestników wykładów z Baz danych;
- wybrać osoby, dla których nie istnieje (NOT EXISTS) wpis na wykład z Baz danych;
- od wszystkich osób możemy odjąć (EXCEPT) te, które chodziły na wykład z Baz danych;

## Alternatywne postaci zapytań - przykład 2

Zapytanie z negacją ma inne możliwe postaci. Pytając o osoby, które nigdy na wykład z BD nie chodziły, możemy:

- wybrać osoby, które nie należą (NOT IN) do grupy uczestników wykładów z Baz danych;
- wybrać osoby, dla których nie istnieje (NOT EXISTS) wpis na wykład z Baz danych;
- od wszystkich osób możemy odjąć (EXCEPT) te, które chodziły na wykład z Baz danych;
- złączyć lewostronnie (LEFT JOIN) wszystkie osoby z wpisami na wykład z Baz danych i znaleźć te, które z niczym się nie połączyły (np. Wybor.kod\_uz IS NULL).

## Alternatywne postaci zapytań - przykład 2

Wybieramy osoby, które nie należą (NOT IN) do grupy uczestników wykładów z Baz danych.

## Alternatywne postaci zapytań - przykład 2

Wybieramy osoby, które nie należą (NOT IN) do grupy uczestników wykładów z Baz danych.

```
select nazwisko from uzytkownik  
where kod_uz not in  
(select wybor.kod_uz from wybor join grupa using(kod_grupy) join  
przedmiot_semestr using(kod_przed_sem) join przedmiot  
using(kod_przed) where nazwa='Bazy danych' and  
rodzaj_zajec='w');
```

## Alternatywne postaci zapytań - przykład 2

Wybieramy osoby, które nie należą (**NOT IN**) do grupy uczestników wykładów z Baz danych.

```
select nazwisko from uzytkownik
where kod_uz not in
(select wybor.kod_uz from wybor join grupa using(kod_grupy) join
przedmiot_semestr using(kod_przed_sem) join przedmiot
using(kod_przed) where nazwa='Bazy danych' and
rodzaj_zajec='w');
```

## Alternatywne postaci zapytań - przykład 2

Wybieramy osoby, które nie należą (**NOT IN**) do grupy uczestników wykładów z Baz danych.

```
select nazwisko from uzytkownik  
where kod_uz not in  
(select wybor.kod_uz from wybor join grupa using(kod_grupy) join  
przedmiot_semestr using(kod_przed_sem) join przedmiot  
using(kod_przed) where nazwa='Bazy danych' and  
rodzaj_zajec='w');
```

Execution Time: 2.587 ms

## Alternatywne postaci zapytań - przykład 2

Wybieramy osoby, dla których nie istnieje (NOT EXISTS) wpis na wykład z Baz danych.



## Alternatywne postaci zapytań - przykład 2

Od wszystkich osób możemy odjąć (EXCEPT) te, które chodziły na wykład z Baz danych.

## Alternatywne postaci zapytań - przykład 2

Wybieramy osoby, dla których nie istnieje (NOT EXISTS) wpis na wykład z Baz danych.

```
select nazwisko from uzytkownik u
where not exists(select * from wybor join grupa using(kod_grupy) join
przedmiot_semestr using(kod_przed_sem) join przedmiot
using(kod_przed) where nazwa='Bazy danych' and rodzaj_zajec='w'
and wybor.kod_uz=u.kod_uz);
```

## Alternatywne postaci zapytań - przykład 2

Wybieramy osoby, dla których nie istnieje (**NOT EXISTS**) wpis na wykład z Baz danych.

```
select nazwisko from uzytkownik u
where not exists(select * from wybor join grupa using(kod_grupy) join
przedmiot_semestr using(kod_przed_sem) join przedmiot
using(kod_przed) where nazwa='Bazy danych' and rodzaj_zajec='w'
and wybor.kod_uz=u.kod_uz);
```

## Alternatywne postaci zapytań - przykład 2

Wybieramy osoby, dla których nie istnieje (**NOT EXISTS**) wpis na wykład z Baz danych.

```
select nazwisko from uzytkownik u
where not exists(select * from wybor join grupa using(kod_grupy) join
przedmiot_semestr using(kod_przed_sem) join przedmiot
using(kod_przed) where nazwa='Bazy danych' and rodzaj_zajec='w'
and wybor.kod_uz=u.kod_uz);
```

Execution Time: 1.810 ms

## Alternatywne postaci zapytań - przykład 2

Od wszystkich osób możemy odjąć (EXCEPT) te, które chodziły na wykład z Baz danych.

```
(select nazwisko,kod_uz from uzytkownik)  
except  
(select nazwisko,kod_uz from uzytkownik join wybor using(kod_uz)  
join grupa using(kod_grupy) join przedmiot_semestr  
using(kod_przed_sem) join przedmiot using(kod_przed) where  
nazwa='Bazy danych' and rodzaj_zajec='w');
```

## Alternatywne postaci zapytań - przykład 2

Od wszystkich osób możemy odjąć (**EXCEPT**) te, które **chodziły na wykład z Baz danych**.

```
(select nazwisko,kod_uz from uzytkownik)  
except  
(select nazwisko,kod_uz from uzytkownik join wybor using(kod_uz)  
join grupa using(kod_grupy) join przedmiot_semestr  
using(kod_przed_sem) join przedmiot using(kod_przed) where  
nazwa='Bazy danych' and rodzaj_zajec='w');
```

## Alternatywne postaci zapytań - przykład 2

Od wszystkich osób możemy odjąć (**EXCEPT**) te, które **chodziły na wykład z Baz danych**.

```
(select nazwisko,kod_uz from uzytkownik)  
except  
(select nazwisko,kod_uz from uzytkownik join wybor using(kod_uz)  
join grupa using(kod_grupy) join przedmiot_semestr  
using(kod_przed_sem) join przedmiot using(kod_przed) where  
nazwa='Bazy danych' and rodzaj_zajec='w');
```

Execution Time: 4.350 ms

## Alternatywne postaci zapytań - przykład 2

Łączymy lewostronnie (LEFT JOIN) wszystkie osoby z wpisami na wykład z Baz danych i znajdujemy te osoby, które z niczym się nie połączyły.



## Alternatywne postaci zapytań - przykład 2

Łączymy lewostronnie (LEFT JOIN) wszystkie osoby z wpisami na wykład z Baz danych i znajdujemy te osoby, które z niczym się nie połączyły.

```
select nazwisko from uzytkownik left join  
((select wybor.kod_uz from wybor join grupa using(kod_grupy) join  
przedmiot_semestr using(kod_przed_sem) join przedmiot  
using(kod_przed) where nazwa='Bazy danych' and  
rodzaj_zajec='w')) AA using(kod_uz) where AA.kod_uz IS NULL;
```

## Alternatywne postaci zapytań - przykład 2

Łączymy lewostronnie (LEFT JOIN) wszystkie osoby z **wpisami na wykład z Baz danych** i znajdujemy te osoby, które z niczym się nie połączyły.

```
select nazwisko from uzytkownik left join  
((select wybor.kod_uz from wybor join grupa using(kod_grupy) join  
przedmiot_semestr using(kod_przed_sem) join przedmiot  
using(kod_przed) where nazwa='Bazy danych' and  
rodzaj_zajec='w')) AA using(kod_uz) where AA.kod_uz IS NULL;
```

## Alternatywne postaci zapytań - przykład 2

Łączymy lewostronnie (**LEFT JOIN**) wszystkie osoby z **wpisami na wykład z Baz danych** i znajdujemy te osoby, które z niczym się nie połączyły.

```
select nazwisko from uzytkownik left join  
((select wybor.kod_uz from wybor join grupa using(kod_grupy) join  
przedmiot_semestr using(kod_przed_sem) join przedmiot  
using(kod_przed) where nazwa='Bazy danych' and  
rodzaj_zajec='w')) AA using(kod_uz) where AA.kod_uz IS NULL;
```

## Alternatywne postaci zapytań - przykład 2

Łączymy lewostronnie (**LEFT JOIN**) wszystkie osoby z **wpisami na wykład z Baz danych** i znajdujemy te osoby, które **z niczym się nie połączyły**.

```
select nazwisko from uzytkownik left join  
((select wybor.kod_uz from wybor join grupa using(kod_grupy) join  
przedmiot_semestr using(kod_przed_sem) join przedmiot  
using(kod_przed) where nazwa='Bazy danych' and  
rodzaj_zajec='w')) AA using(kod_uz) where AA.kod_uz IS NULL;
```

## Alternatywne postaci zapytań - przykład 2

Łączymy lewostronnie (**LEFT JOIN**) wszystkie osoby z **wpisami na wykład z Baz danych** i znajdujemy te osoby, które **z niczym się nie połączyły**.

```
select nazwisko from uzytkownik left join  
((select wybor.kod_uz from wybor join grupa using(kod_grupy) join  
przedmiot_semestr using(kod_przed_sem) join przedmiot  
using(kod_przed) where nazwa='Bazy danych' and  
rodzaj_zajec='w')) AA using(kod_uz) where AA.kod_uz IS NULL;
```

Execution Time: 1.736 ms

# Podzapytania w klauzuli FROM

# Podzapytania w klauzuli FROM

O tym było właśnie przed chwilą!

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

# Podzapytania w klauzuli FROM

O tym było właśnie przed chwilą!

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...



# Podzapytania w klauzuli FROM

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...

# Podzapytania w klauzuli FROM

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...
- Wyrażenie może odwoływać się do innych tabel, ale nie do innych aliasów, czyli zmiennych krotkowych;

# Podzapytania w klauzuli FROM

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...
- Wyrażenie może odwoływać się do innych tabel, ale nie do innych aliasów, czyli zmiennych krotkowych - **LATERAL**

# Podzapytania w klauzuli FROM

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...
- Wyrażenie może odwoływać się do innych tabel, ale nie do innych aliasów, czyli zmiennych krotkowych - **LATERAL**

```
SELECT m.name FROM manufacturers m LEFT JOIN  
LATERAL get_product_names(m.id) pname ON true  
WHERE pname IS NULL;
```

# Podzapytania w klauzuli FROM

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...
- Wyrażenie może odwoływać się do innych tabel, ale nie do innych aliasów, czyli zmiennych krotkowych;
- Wyrażenia można łączyć przez JOIN, UNION,

# Podzapytania w klauzuli FROM

SELECT nazwisko FROM Uzytkownik

**JOIN** (wpisy na wykład z BD) AA **USING** (kod\_uz)

**JOIN** (grupy prowadzone przez PKA) BB **ON** (...)

WHERE semestr>5;

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...
- Wyrażenie może odwoływać się do innych tabel, ale nie do innych aliasów, czyli zmiennych krotkowych;
- Wyrażenia można łączyć przez **JOIN**, UNION,

# Podzapytania w klauzuli FROM

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...
- Wyrażenie może odwoływać się do innych tabel, ale nie do innych aliasów, czyli zmiennych krotkowych;
- Wyrażenia można łączyć przez JOIN, UNION,
- Wyrażenia muszą mieć aliasy!

# Podzapytania w klauzuli FROM

SELECT nazwisko FROM Uzytkownik

JOIN (wpisy na wykład z BD) **AA** USING (kod\_uz)

JOIN (grupy prowadzone przez PKA) **BB** ON (...)

WHERE semestr>5;

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...
- Wyrażenie może odwoływać się do innych tabel, ale nie do innych aliasów, czyli zmiennych krotkowych;
- Wyrażenia można łączyć przez JOIN, UNION,
- Wyrażenia muszą mieć **aliasy**!



# Podzapytania w klauzuli FROM

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

- Wyrażenia są obliczane przed rozpoczęciem obliczania klauzul WHERE i SELECT...
- Wyrażenie może odwoływać się do innych tabel, ale nie do innych aliasów, czyli zmiennych krotkowych;
- Wyrażenia można łączyć przez JOIN, UNION,
- Wyrażenia muszą mieć aliasy!
- Wyrażenia pozwalają nam zapisać w SQL praktycznie dowolnie skomplikowane zapytanie algebry relacji lub relacyjnego rachunku krotek.

# Podzapytania w klauzuli FROM

```
SELECT nazwisko FROM Uzytkownik  
      JOIN (wpisy na wykład z BD) AA USING (kod_uz)  
      JOIN (grupy prowadzone przez PKA) BB ON (...)  
WHERE semestr>5;
```

**Nie należy przesadzać z wyrażeniami tabelowymi!**

- Wyrażenia muszą mieć aliasy!
- Wyrażenia pozwalają nam zapisać w SQL praktycznie dowolnie skomplikowane zapytanie algebry relacji lub relacyjnego rachunku krotek.

# Upraszczenie zapytań

# Upraszczenie zapytań

Podzapytania w klauzuli FROM

```
SELECT ... FROM (...) AA, R, S WHERE F(R,S,AA)
```

*-- takie sobie uproszczenie*

Tabele tymczasowe

```
CREATE TEMP TABLE AA (...); INSERT INTO AA SELECT...  
SELECT ... FROM AA, R, S WHERE F(R,S,AA)
```

*-- tabel można używać wielokrotnie; są kasowane na koniec sesji*

Perspektywy (widoki)

```
CREATE VIEW AA AS SELECT ...  
SELECT ... FROM AA, R, S WHERE F(R,S,AA)
```

*-- perspektywy są obiektem trwałym, ale wirtualnym*

Instrukcja WITH

```
WITH AA AS (...)  
SELECT ... FROM AA, BB, R, S WHERE F(R,S,AA)
```

*-- na pierwszy rzut oka wygląda jak podzapytanie, ale pozwala na rekursję!*

# Upraszczenie zapytań

Podzapytania w klauzuli FROM

```
SELECT ... FROM (...) AA, R, S WHERE F(R,S,AA)
```

*-- takie sobie uproszczenie*

Tabele tymczasowe

```
CREATE TEMP TABLE AA (...); INSERT INTO AA SELECT...  
SELECT ... FROM AA, R, S WHERE F(R,S,AA)
```

*-- tabel można używać wielokrotnie; są kasowane na koniec sesji*

Perspektywy (widoki)

```
CREATE VIEW AA AS SELECT ...  
SELECT ... FROM AA, R, S WHERE F(R,S,AA)
```

*-- perspektywy są obiektem trwałym, ale wirtualnym*

Instrukcja WITH ← **bardzo przydatne**

```
WITH AA AS (...)  
SELECT ... FROM AA, BB, R, S WHERE F(R,S,AA)
```

*-- na pierwszy rzut oka wygląda jak podzapytanie, ale pozwala na rekursję!*

# Upraszczenie zapytań - TEMP TABLE

```
CREATE TEMPORARY TABLE trudneBD LIKE(zad);  
INSERT INTO trudneBD  
    SELECT zad.*  
    FROM zad JOIN klas ON zad.id=klas.zad JOIN kat ON kat.id=klas.kat  
    WHERE klas.trudnosc=100 AND kat.nazwa='Bazy danych';
```

```
CREATE TEMPORARY TABLE masters2019  
    (id int, imie text, nazwisko text, wsp real);  
  
INSERT INTO masters2019  
    SELECT id, imie, nazwisko, avg(wynik)  
    FROM osoba JOIN wynik ON id=osoba  
    WHERE EXTRACT(year FROM dataWyn)=2019  
    GROUP BY id, imie, nazwisko  
    HAVING count(DISTINCT zad.id)>100 AND avg(wynik)>75;
```

# Upraszczenie zapytań - TEMP TABLE

Korzystając z `trudneBD` i `masters2019` możemy znaleźć "mistrzów 2019", którzy robili trudne zadania z baz danych:

```
SELECT masters2019.id, imie, nazwisko, avg(wynik)
FROM masters2019 JOIN wynik ON id=osoba JOIN
    trudneBD ON trudneBD.id=wynik.zad
GROUP BY masters2019.id, imie, nazwisko
ORDER BY 4 DESC;
```

# Upraszczenie zapytań - **VIEWS**

```
CREATE VIEW trudneBD AS  
  SELECT zad.*  
  FROM zad JOIN klas ON zad.id=klad.zad JOIN kat ON kat.id=klas.kat  
  WHERE trudnosc=100 AND kat.nazwa='Bazy danych';
```

```
CREATE VIEW masters2019(id, imie, nazwisko, wsp) AS  
  SELECT id, imie, nazwisko, avg(wynik)  
  FROM osoba JOIN wynik ON id=osoba  
  WHERE EXTRACT(year FROM dataWyn)=2019  
  GROUP BY id, imie, nazwisko  
  HAVING count(DIST zad.id)>100 AND avg(wynik)>75;
```



# Upraszczenie zapytań - **VIEWS**

Korzystając z **trudneBD** i **masters2019** możemy znaleźć "mistrzów 2019", którzy robili trudne zadania z baz danych:

```
SELECT masters2019.id, imie, nazwisko, avg(wynik)
FROM masters2019 JOIN wynik ON id=osoba JOIN
     trudneBD ON trudneBD.id=wynik.zad
GROUP BY masters2019.id, imie, nazwisko
ORDER BY 4 DESC;
```

# TEMP TABLE versus VIEWS

id	nazwisko	imie	wsp

id	nazwisko	imie	wsp

id	imie	...
1	Anna	...
2	Basia	...

osoba	zad	czasWyk	wynik
1	13	2019-...	80
2	15	2019-...	70

# TEMP TABLE versus VIEWS

id	nazwisko	imie	wsp
1	Abacka	Anna	80

id	nazwisko	imie	wsp
1	Abacka	Anna	80

id	imie	...
1	Anna	...
2	Basia	...

osoba	zad	czasWyk	wynik
1	13	2019-...	80
2	15	2019-...	70

# TEMP TABLE versus VIEWS

id	nazwisko	imie	wsp
1	Abacka	Anna	80

id	nazwisko	imie	wsp
1	Abacka	Anna	80

id	imie	...
1	Anna	...
2	Basia	...

osoba	zad	czasWyk	wynik
1	13	2019-...	80
2	15	2019-...	70
2	13	2019-...	100

# TEMP TABLE versus VIEWS

id	nazwisko	imie	wsp
1	Abacka	Anna	80

id	nazwisko	imie	wsp
1	Abacka	Anna	80
2	Babacka	Basia	85

id	imie	...
1	Anna	...
2	Basia	...

osoba	zad	czasWyk	wynik
1	13	2019-...	80
2	15	2019-...	70
2	13	2019-...	100

# TEMP TABLE versus VIEWS

id	nazwisko	imie	wsp
1	Abacka	Anna	80

id	nazwisko	imie	wsp
1	Abacka	Anna	80
2	Babacka	Basia	85

id	imie	...
1	Anna	...
2	Basia	...

osoba	zad	czasWyk	wynik
1	13	2019-...	80
2	15	2019-...	70
2	13	2019-...	100
1	15	2019-...	30

# TEMP TABLE versus VIEWS

id	nazwisko	imie	wsp
1	Abacka	Anna	80

id	nazwisko	imie	wsp
1	Abacka	Anna	80
2	Babacka	Basia	85

id	imie	...
1	Anna	...
2	Basia	...

osoba	zad	czasWyk	wynik
1	13	2019-...	80
2	15	2019-...	70
2	13	2019-...	100
1	15	2019-...	30

# Klauzula WITH

```
WITH trudneBD AS (SELECT zad.* FROM zad JOIN klas ON  
zad.id=klad.zad JOIN kat ON kat.id=klas.kat WHERE trudnosc=100 AND  
kat.nazwa='Bazy danych'),
```

```
masters2019 AS (SELECT id, imie, nazwisko, avg(wynik) AS "wsp"  
FROM osoba JOIN wynik ON id=osoba  
WHERE EXTRACT(year FROM dataWyn)=2019  
GROUP BY id, imie, nazwisko  
HAVING count(DIST zad.id)>100 AND avg(wynik)>75)
```

```
SELECT masters2019.id, imie, nazwisko, avg(wynik)  
FROM masters2019 JOIN wynik ON id=osoba JOIN  
trudneBD ON trudneBD.id=wynik.zad  
GROUP BY masters2019.id, imie, nazwisko  
ORDER BY 4 DESC;
```



# Klauzula WITH

- WITH może poprzedzać nie tylko SELECT ale również INSERT, UPDATE lub DELETE.
- WITH może też używać poleceń modyfikujących bazę  
WITH **moved\_rows** AS (  
    **DELETE** FROM products  
    WHERE  
        "date" >= '2010-10-01' AND  
        "date" < '2010-11-01'  
    **RETURNING \***  
)  
INSERT INTO products\_log  
SELECT \* FROM **moved\_rows**;

## Klauzula WITH - rekursja

```
WITH RECURSIVE Path(a, b) AS (  
  SELECT a, b FROM Edge  
  UNION ALL  
  SELECT a, b FROM Edge e JOIN Path p ON (e.b = p.a))  
SELECT * FROM Path;
```

## Klauzula WITH - rekursja

```
WITH RECURSIVE Path(a, b) AS (  
  SELECT a, b FROM Edge  
  UNION  
  SELECT a, b FROM Edge e JOIN Path p ON (e.b = p.a))  
SELECT * FROM Path;
```

# Klauzula WITH - rekursja

```
WITH RECURSIVE Path(a, b) AS (  
  SELECT a, b FROM Edge  
  UNION  
  SELECT a, b FROM Edge e JOIN Path p ON (e.b = p.a))  
SELECT * FROM Path;
```

```
=> CREATE TABLE edge(a int, b int);  
=> INSERT INTO edge VALUES (1,2), (2,3);  
=> SELECT a, b FROM Edge;
```

a	b
1	2
2	3

(2 rows)

# Klauzula WITH - rekursja

```
WITH RECURSIVE Path(a, b) AS (  
  SELECT a, b FROM Edge  
  UNION  
  SELECT a, b FROM Edge e JOIN Path p ON (e.b = p.a))  
SELECT * FROM Path;
```

=> WITH (...) SELECT \* FROM Path;

a	b
1	2
2	3
1	3

(3 rows)

# Klauzula WITH - rekursja

```
WITH RECURSIVE Path(a, b) AS (  
  SELECT a, b FROM Edge  
  UNION  
  SELECT a, b FROM Edge e JOIN Path p ON (e.b = p.a))  
SELECT * FROM Path;
```

=> INSERT INTO edge VALUES (3,1);

=> WITH (...) SELECT \* FROM Path;

# Klausele WITH - rekursja

```
WITH RECURSIVE Path(a, b) AS (  
  SELECT a, b FROM Edge  
  UNION  
  SELECT a, b FROM Edge e JOIN Path p ON (e.b = p.a)  
  SELECT * FROM Path;
```

=> INSERT INTO edge VALUES (3,1);

=> WITH (...) SELECT \* FROM Path;

a	b
---	---

1	2
2	3
1	3

3	2
3	3
1	1

3	1
2	1
2	2

(9 rows)

## Klauzula WITH - rekursja

```
WITH RECURSIVE search_graph(id, link, data, depth) AS (  
    SELECT g.id, g.link, g.data, 1  
    FROM graph g  
    UNION  
    SELECT g.id, g.link, g.data, sg.depth + 1  
    FROM graph g, search_graph sg  
    WHERE g.id = sg.link  
)  
SELECT * FROM search_graph;
```



## Klauzula WITH - rekursja

```
WITH RECURSIVE search_graph(id, link, data, depth) AS (  
    SELECT g.id, g.link, g.data, 1  
    FROM graph g  
    UNION  
    SELECT g.id, g.link, g.data, sg.depth + 1  
    FROM graph g, search_graph sg  
    WHERE g.id = sg.link  
)  
SELECT * FROM search_graph;
```

Cykle?

## Klauzula WITH - rekursja

```
WITH RECURSIVE search_graph(id, link, data, depth) AS (  
    SELECT g.id, g.link, g.data, 1  
    FROM graph g  
    UNION  
    SELECT g.id, g.link, g.data, sg.depth + 1  
    FROM graph g, search_graph sg  
    WHERE g.id = sg.link  
)  
SELECT * FROM search_graph;
```

Cykle? usuwanie duplikatów nie pomoże bo depth!

## DELETE + join

```
DELETE FROM films USING producers
WHERE producer_id = producers.id AND producers.name = 'foo';
```

```
DELETE FROM films
WHERE producer_id IN (SELECT id FROM producers WHERE
name = 'foo');
```

## DELETE + join

```
DELETE FROM films USING producers
WHERE producer_id = producers.id AND producers.name = 'foo';
```

```
DELETE FROM films
WHERE producer_id IN (SELECT id FROM producers WHERE
name = 'foo');
```

**Dziękuję**