



# Bazy Danych

---

## SQL – Podstawy języka II: zapytania

Krzysztof Regulski

WIMiP, KISiM,

[regulski@metal.agh.edu.pl](mailto:regulski@metal.agh.edu.pl)

Konsultacje: wtorek, godz. 15:00 – 16:30

B5, pok. 409

---

## Konstrukcja *select-from-where*

– SQL oparty jest na algebrze relacji z pewnymi modyfikacjami i rozszerzeniami.

– Typowe zapytanie SQL ma postać:

```
SELECT  $A_1, A_2, \dots, A_k$   
FROM  $r_1, r_2, \dots, r_m$   
WHERE  $P$ 
```

$r_i$  oznaczają relacje w bazie danych.

$A_i$  oznaczają atrybuty tych relacji.

$P$  jest predykatem.

– Wynikiem zapytania SQL jest relacja.

## SELECT (1)

---

- Klauzula **SELECT** jest używana do wskazania tych atrybutów relacji określonych w klauzuli **FROM**, które są objęte zapytaniem.

» Przykład: znajdź nazwy wszystkich oddziałów z relacji `oddzialy`

```
SELECT nazwa_oddzialu  
FROM oddzialy;
```

- Gwiazdka w klauzuli **SELECT** oznacza „wszystkie atrybuty relacji”

```
SELECT *  
FROM oddzialy;
```

# Przykładowa tabela – dowody wydania Rw

## DowodyWydania

Symbol Towaru	NazwaTowaru	Magazyn	Od-biorca	Data	Ilosc	J.m.
S0001	rura fi 0,63 gr 0,2	MWG01	HS	16.04.04	12	mb
S0025	rura fi 1,26 gr 0,3	MWG02	PP	12.04.04	80	mb
S0001	rura fi 0,63 gr 0,2	MWG01	PP	12.04.04	15	mb
S1025	złączka	MOZ01	ZP	03.04.04	100	szt.
S0025	rura fi 1,26 gr 0,3	MOZ01	ZP	03.04.04	20	mb
S0152	rura kw. 2 gr 0,2	MOP02	TT	01.04.04	1	mb

## Przykład zapytania

```
SELECT *  
FROM DowodyWydania  
WHERE Magazyn = 'MWG01'
```

Symbol Towaru	NazwaTowaru	Magazyn	Od- biorc a	Data	Ilos c	J.m.
S0001	rura fi 0,63 gr 0,2	MWG01	HS	16.04.04	12	mb
S0001	rura fi 0,63 gr 0,2	MWG01	PP	12.04.04	15	mb

## SELECT (2)

---

- SQL dopuszcza duplikaty zarówno w relacjach jak i rezultatach zapytań.
- Dla wymuszenia eliminacji duplikatów wstawia się słowo kluczowe **DISTINCT** po **SELECT**.

Przykład: znajdź imiona wszystkich pracowników i usuń duplikaty

```
SELECT DISTINCT imie  
FROM pracownicy;
```

- Słowo kluczowe **ALL** oznacza, że duplikaty nie będą usuwane

```
SELECT ALL imie  
FROM pracownicy;
```

## SELECT (3)

---

- Klauzula **SELECT** może zawierać **wyrażenia arytmetyczne** z operatorami **+**, **-**, **\***, **/** operujące na stałych i atrybutach krotek
- Zapytanie:

```
SELECT nazwisko, imie, placa + 100  
FROM pracownicy;
```

zwróci relację, w której atrybut `placa` będzie zwiększony o 100.

## SELECT (4)

- Polecenia **SELECT** można używać również nie odwołując się do żadnej tabeli w celu obliczania wyrażeń lub pracy na ciągach znaków lub zmiennych:
- Przykłady:

```
SELECT 1+1;
```

```
SELECT `ten napis pojawi sie na ekranie`;
```

```
SELECT `słowa`, `w`, `osobnych`, `kolumnach`;
```

```
SELECT @liczba1:=8 AS A, @liczba2:=2 AS B,  
        @wynik:=@liczba1+@liczba2 AS 'WYNIK A+B';
```

```
+---+---+-----+  
| A | B | WYNIK A+B |  
+---+---+-----+  
| 8 | 2 |          10 |  
+---+---+-----+
```



## Klauzula WHERE (1)

- Klauzula **WHERE** składa się z warunków dotyczących atrybutów relacji z klauzuli **FROM**. Umożliwia wyświetlanie wierszy, których kolumny spełniają określony warunek. Pola objęte klauzurą **WHERE** nie muszą być na liście wyboru.

```
SELECT nazwisko, imie  
FROM pracownicy  
WHERE placa > 100;
```

- umożliwia łączenie tabel według różnych kryteriów.

zapytanie  
(1)

```
SELECT CONCAT (pracownicy.imie, ' ',  
pracownicy.nazwisko)  
AS PRACOWNIK  
FROM pracownicy, oddzialy  
WHERE pracownicy.id_oddzialu=oddzialy.id_oddzialu  
AND oddzialy.nazwa_oddzialu LIKE 'Betatrex';
```

# Klauzula WHERE (2)

```

+-----+
| pracownicy |
+-----+
| pesel      |
| imie       |
| nazwisko   |
| id_oddzialu |
+-----+

```

```

+-----+
| oddzialy   |
+-----+
| id_oddzialu |
| nazwa_oddzialu |
+-----+

```

zapytanie  
(1)

```

+-----+
| PRACOWNIK |
+-----+
| Okowita Ambrozjowa |
| Fableusz Kosonosy |
| Atanazy Angonilewicz |
| Kosmateusz Buler |
| Nikczemilian Wikrus |
| Krowimir Dojny |
| Inwertyk Dosiebski |
+-----+

```

# Operatory porównań

- SQL używa logicznych operatorów **AND**, **OR** i **NOT**, **=**, **<**, **>**, **>=**, **<=**, **<>**

- **(NOT) BETWEEN . . AND** kiedy specyfikuje się, że wartość ma zawierać się w określonym przedziale zamkniętym

np.: **WHERE** cena **BETWEEN** 'dolna\_granica' **AND** 'górna\_granica'

- **(NOT) IN (e<sub>1</sub>, . . . , e<sub>n</sub>)** kiedy porównujemy do jednego z elementów zbioru

np.: **WHERE** imie **NOT IN** ('Stefan', 'Bożena')

- **(NOT) LIKE** kiedy porównujemy ciągi znaków do wzorca

Wyrażenia regularne:

%	- dowolny ciąg znaków
<u>_</u> (podkreślenie)	- dowolny znak
[ck]	- znak 'c' lub 'k'
[c-k]	- znak z zakresu od 'c' do 'k'
[^c]	- nie 'c'

- **IS (NOT) NULL** służy do sprawdzania, czy wartość w polu to NULL

## Przykładowa tabela – towary

IdTowaru	SymbolTowaru	NazwaTowaru
1	RZ001	Rura zgrz. fi 6,3 gr 0,2
2	RZ002	Rura zgrz. fi 12,6 gr 0,2
3	RZ003	Rura zgrz. fi 6,3 gr 0,3
4	RZ004	Rura zgrz. fi 12,6 gr 0,3
5	RZ011	Rura zgrz. kw 4 gr 0,2
6	RZ012	Rura zgrz. kw 5 gr 0,3
7	ZL001	Złączka 1'
8	ZL002	Złączka 2'

## Przykład zapytania

```
SELECT *  
FROM `towar`  
WHERE SymbolTowaru LIKE 'R%'
```

IdTowaru	SymbolTowaru	NazwaTowaru
1	RZ001	Rura zgrz. fi 6,3 gr 0,2
2	RZ002	Rura zgrz. fi 12,6 gr 0,2
3	RZ003	Rura zgrz. fi 6,3 gr 0,3
4	RZ004	Rura zgrz. fi 12,6 gr 0,3
5	RZ011	Rura zgrz. kw 4 gr 0,2
6	RZ012	Rura zgrz. kw 5 gr 0,3

## Łączenie relacji

- Operacje połączenia (**JOIN**) bierze dwie relacje i zwraca jako wynik inną relację.
- Te dodatkowe operacje są zazwyczaj używane jako polecenie podzapytania w klauzuli **FROM**.
- Warunki połączenia definiują, które krotki z dwóch relacji pasują i które atrybuty będą obecne jako wynik połączenia.
- Typ połączenia - definiuje jak będą traktowane takie krotki z poszczególnych relacji, które nie pasują do krotek z drugiej relacji.

```
tabela1 [NATURAL | LEFT | RIGHT | INNER | OUTER] JOIN  
tabela2
```

- Warunki łączenia:  
**NATURAL**, **ON** <warunek\_złączenia>, **USING** <lista\_atrybutów>
- **JOIN** w większości przypadków może być zastąpione przez odpowiednie klauzule **FROM** i **WHERE**

## Rzutowanie w SQL

```
SELECT SymbolTowaru, NazwaTowaru  
FROM `towar`  
WHERE SymbolTowaru LIKE 'R%'
```

SymbolTowaru	NazwaTowaru
RZ001	Rura zgrz. fi 6,3 gr 0,2
RZ002	Rura zgrz. fi 12,6 gr 0,2
RZ003	Rura zgrz. fi 6,3 gr 0,3
RZ004	Rura zgrz. fi 12,6 gr 0,3
RZ011	Rura zgrz. kw 4 gr 0,2
RZ012	Rura zgrz. kw 5 gr 0,3

## Warunki selekcji i porządkowanie

```
SELECT NrZamowienia, DataZamowienia
FROM Zamowienie
WHERE NrZamowienia LIKE '____2004' AND
DataZamowienia > '2004-04-04'
ORDER BY DataZamowienia DESC
```

NrZamowienia	DataZamowienia
005/2004	2004-04-07 00:00:00
003/2004	2004-04-06 00:00:00
004/2004	2004-04-06 00:00:00
002/2004	2004-04-05 00:00:00



# Rodzaje złączeń (1)

## Złączenie naturalne (NATURAL)

- polega na połączeniu w pary tych krotek z relacji  $R$  i  $S$ , które mają identyczne wartości dla wszystkich wspólnych atrybutów i jest oznaczane  $R \bowtie S$
- w rezultacie powstaje relacja, której schemat zawiera atrybuty relacji  $R$  i relacji  $S$ , przy czym wspólna część uwzględniana jest tylko raz

## Złączenie teta (ON <warunek\_złączenia>)

- polega na złączeniu dwóch relacji  $R$  i  $S$  w iloczyn kartezyjski i wyborze z niego tych krotek, które spełniają wyrażenie warunkowe na parze lub zbiorze par atrybutów z  $R$  i  $S$  i jest oznaczane symbolem  $R \bowtie_{\Theta} R$  lub  $R \bowtie_C S$ , gdzie  $\Theta$  lub  $C$  to wyrażenia logiczne

## Równozłączenie (USING <lista\_atrybutów>)

- to szczególny przypadek złączenia teta, w którym warunek ma charakter równości wybranych atrybutów obu relacji
- powtarzające się kolumny opisujące atrybuty z warunku złączenia są pomijane

## Rodzaje złączeń (2)

- **złączenie wewnętrzne (INNER JOIN)** – w relacji wynikowej występują wyłącznie te krotki, które spełniają warunek złączenia
- **złączenie lewostronne zewnętrzne (LEFT OUTER JOIN)** – zawiera wszystkie krotki **R** uzupełnione krotkami **S** spełniającymi warunek
- **złączenie prawostronne zewnętrzne (RIGHT OUTER JOIN)** - zawiera wszystkie krotki **S** uzupełnione krotkami **R** spełniającymi warunek
- **złączenie zewnętrzne pełne (FULL OUTER JOIN)** – zawiera wszystkie krotki **R** oraz **S** uzupełnione wartościami typu **NULL** gdy do danej krotki nie pasuje żadna krotka z drugiej relacji
- **złączenie wewnętrzne typu CROSS** – w relacji wynikowej występują wszystkie krotki będące wynikiem iloczynu kartezyjskiego
- **złączenie zewnętrzne typu UNION** - zawiera wszystkie krotki **R** nie pasujące do żadnej krotki **S** uzupełnione krotkami **S** nie pasującymi do żadnej krotki **R**

## Przykład – dane:

– Relacja **oddzialy**:

id_oddzialu	nazwa_oddzialu
L140	Betatrex
A4	Alfatron
B340	Tetrix

– Relacja **pracownicy**:

pesel	imie	nazwisko	id_oddzialu
75102406713	Okowita	Ambrozjowa	L140
54032204567	Fableusz	Kosonosy	L140
56123099087	Atanazy	Angonilewicz	A4

## Przykład:

```
SELECT oddzialy.nazwa_oddzialu, pracownicy.nazwisko  
FROM (oddzialy INNER JOIN pracownicy  
ON oddzialy.id_oddzialu=pracownicy.id_oddzialu);
```

```
+-----+-----+  
| nazwa_oddzialu | nazwisko      |  
+-----+-----+  
| Betatrex       | Ambrozjowa    |  
| Betatrex       | Kosonosy      |  
| Alfatron       | Angonilewicz  |  
+-----+-----+
```

```
SELECT oddzialy.nazwa_oddzialu, pracownicy.nazwisko  
FROM (oddzialy LEFT OUTER JOIN pracownicy  
ON oddzialy.id_oddzialu=pracownicy.id_oddzialu);
```

```
+-----+-----+  
| nazwa_oddzialu | nazwisko      |  
+-----+-----+  
| Betatrex       | Ambrozjowa    |  
| Betatrex       | Kosonosy      |  
| Alfatron       | Angonilewicz  |  
| Tetrax         | NULL          |  
+-----+-----+
```

– Możliwe jest używanie aliasów nazw kolumn i nazw tabel.

Umożliwiają one:

- » zmianę nazwy kolumny wyświetlanej
- » nadanie nazwy kolumnie będącej wynikiem wyrażenia lub stałą

## **SELECT**

```
@liczba1:=8 AS A,
```

```
@liczba2:=2 AS B,
```

```
@wynik:=@liczba1+@liczba2 AS 'WYNIK A+B';
```

```
+---+---+-----+  
| A | B | WYNIK A+B |  
+---+---+-----+  
| 8 | 2 |           10 |  
+---+---+-----+
```

## Sortowanie wyników

---

- Sortowanie wyników osiąga się dzięki klauzuli **ORDER BY**. Sortowanie odbywa się kolejno według wartości atrybutów wymienionych w klauzuli.
- Dla każdego z atrybutów można podać specyfikator **DESC** dla porządku malejącego lub **ASC** dla porządku rosnącego. Porządek rosnący jest domyślny.
- Ponieważ sortowanie dużej ilości krotek jest kosztowne, wskazane jest wykonywanie sortowania tylko wtedy, gdy jest to niezbędne.

```
SELECT nazwisko, imie  
FROM pracownicy  
ORDER BY nazwisko DESC, imie DESC;
```

# Operacje teoriomnogościowe

- Operacje **UNION**, **INTERSECT** oraz **EXCEPT** odpowiadają kolejno następującym operatorom algebry relacyjnej:  $\cup$ ,  $\cap$  i  $-$  przy czym zachodzi:

$$r \cap s = r - (r - s)$$

- Każda z operacji automatycznie eliminuje duplikaty; dla zachowania duplikatów stosuje się wersje **UNION ALL**, **INTERSECT ALL** oraz **EXCEPT ALL**.

```
(SELECT nazwa_firmy FROM dostawcy)
UNION
(SELECT nazwa FROM klienci);
```

```
(SELECT nazwa_firmy FROM dostawcy)
INTERSECT
(SELECT nazwa FROM klienci);
```

```
(SELECT nazwa_firmy FROM dostawcy)
EXCEPT
(SELECT nazwa FROM klienci);
```

## Przykładowe dane

IdKlienta	NazwaKlienta	Telefon	KodPocztowy
1	FH Klin SA	48 12 1273210	30-121
2	Firma Krok Sp zoo	48 12 6374532	30-321
3	STALHANDEL	48 32 7865748	34-876
4	Rower Polska SA	48 12 2853364	32-082

IdBanku	IdKlienta	NrKonta
1	1	12345678901234567892022222
2	1	43527897963543645632726336
3	2	46748329374637843254632546
1	2	78789798979879879877878978
1	3	98087079643906432786443324
2	3	67876864376438209876473674
3	4	67686868768348364836483764



# Iloczyn kartezjański

```
SELECT NazwaKlienta, NrKonta
FROM Klient, Konto
ORDER BY NazwaKlienta
```

NazwaKlienta	NrKonta
FH Klin SA	78789798979879879877878978
FH Klin SA	43527897963543645632726336
FH Klin SA	67686868768348364836483764
FH Klin SA	46748329374637843254632546
FH Klin SA	67876864376438209876473674
FH Klin SA	12345678901234567892022222
FH Klin SA	98087079643906432786443324
Firma Krok Sp zoo	12345678901234567892022222
Firma Krok Sp zoo	43527897963543645632726336
Firma Krok Sp zoo	46748329374637843254632546
Firma Krok Sp zoo	78789798979879879877878978
Firma Krok Sp zoo	67876864376438209876473674
Firma Krok Sp zoo	67686868768348364836483764
Firma Krok Sp zoo	98087079643906432786443324
Rower Polska SA	12345678901234567892022222
Rower Polska SA	67876864376438209876473674
Rower Polska SA	43527897963543645632726336
Rower Polska SA	46748329374637843254632546
Rower Polska SA	98087079643906432786443324
Rower Polska SA	67686868768348364836483764
Rower Polska SA	78789798979879879877878978
STALHANDEL	46748329374637843254632546
STALHANDEL	78789798979879879877878978
STALHANDEL	43527897963543645632726336
STALHANDEL	98087079643906432786443324
STALHANDEL	12345678901234567892022222
STALHANDEL	67686868768348364836483764
STALHANDEL	67876864376438209876473674

## Złączenie naturalne

```

SELECT NazwaKlienta, NrKonta
FROM Klient ,Konto
WHERE Klient.IdKlienta = Konto.IdKlienta
ORDER BY NazwaKlienta
  
```

```

SELECT NazwaKlienta, NrKonta
FROM Klient JOIN Konto USING (IdKlienta)
ORDER BY NazwaKlienta
  
```

NazwaKlienta	NrKonta
FH Klin SA	12345678901234567892022222
FH Klin SA	43527897963543645632726336
Firma Krok Sp zoo	46748329374637843254632546
Firma Krok Sp zoo	78789798979879879877878978
STALHANDEL	98087079643906432786443324
STALHANDEL	67876864376438209876473674
Rower Polska SA	67686868768348364836483764

## Złączenie naturalne – trzy tabele

---

```
SELECT NazwaKlienta, NrKonta, NazwaBanku  
FROM Klient JOIN Konto USING  
(IdKlienta) JOIN Bank USING (IdBanku)  
ORDER BY NazwaKlienta, NrKonta
```

```
SELECT NazwaKlienta, NrKonta, NazwaBanku  
FROM Klient, Konto, Bank  
WHERE Klient.IdKlienta = Konto.IdKlienta  
AND Konto.IdBanku = Bank.IdBanku  
ORDER BY NazwaKlienta, NrKonta
```

## Złączenie naturalne – trzy tabele

NazwaKlienta	NrKonta	NazwaBanku
FH Klin SA	12345678901234567892022222	Bank BPH
FH Klin SA	43527897963543645632726336	Bank Polski
Firma Krok Sp zoo	46748329374637843254632546	Bank Niemiecki
Firma Krok Sp zoo	78789798979879879877878978	Bank BPH
Rower Polska SA	67686868768348364836483764	Bank Niemiecki
STALHANDEL	67876864376438209876473674	Bank Polski
STALHANDEL	98087079643906432786443324	Bank BPH

## Złączenia zewnętrzne – złączenie lewostronne

---

```
SELECT DISTINCT NazwaKlienta, DataZamowienia
FROM Klient LEFT JOIN Zamowienie USING
(IdKlienta)
```

NazwaKlienta	DataZamowienia
FH Klin SA	2004-04-04 00:00:00
FH Klin SA	2004-04-06 00:00:00
Firma Krok Sp zoo	2004-04-05 00:00:00
STALHANDEL	2004-04-06 00:00:00
Rower Polska SA	2004-04-07 00:00:00
PHPU OSA	[NULL]

## Złączenia zewnętrzne – złączenie lewostronne

---

```
SELECT DISTINCT NazwaTowaru, DataZamowienia, Ilosc  
FROM Towar LEFT JOIN LiniaZamowienia USING  
(IdTowaru) LEFT JOIN Zamowienie USING  
(IdZamowienia)  
ORDER BY NazwaTowaru
```

NazwaTowaru	DataZamowienia	Ilosc
Rura zgrz. fi 12,6 gr 0,2	2004-04-04 00:00:00	12
Rura zgrz. fi 12,6 gr 0,2	2004-04-07 00:00:00	50
Rura zgrz. fi 12,6 gr 0,3	2004-04-06 00:00:00	12
Rura zgrz. fi 6,3 gr 0,2	2004-04-04 00:00:00	20
Rura zgrz. fi 6,3 gr 0,2	2004-04-06 00:00:00	50
Rura zgrz. fi 6,3 gr 0,2	2004-04-05 00:00:00	100
Rura zgrz. fi 6,3 gr 0,3	2004-04-04 00:00:00	25
Rura zgrz. fi 6,3 gr 0,3	2004-04-06 00:00:00	50
Rura zgrz. kw 4 gr 0,2	2004-04-07 00:00:00	30
Rura zgrz. kw 4 gr 0,2	2004-04-05 00:00:00	6
Rura zgrz. kw 5 gr 0,3	[NULL]	[NULL]
Zlaczka 1'	2004-04-06 00:00:00	100
Zlaczka 2'	2004-04-06 00:00:00	50
Zlaczka 3/4'	[NULL]	[NULL]

## Złączenia zewnętrzne – złączenie prawostronne

---

```
SELECT NazwaKlienta, NrKonta, NazwaBanku  
FROM Klient JOIN Konto USING (IdKlienta)  
RIGHT JOIN Bank Using (IdBanku)  
ORDER BY NazwaKlienta, NazwaBanku
```

```
SELECT NazwaKlienta, NrKonta, NazwaBanku  
FROM Klient JOIN Konto ON (Klient.IdKlienta =  
Konto.IdKlienta) RIGHT JOIN Bank ON  
(Konto.IdBanku = Bank.IdBanku)  
ORDER BY NazwaKlienta, NazwaBanku
```





## Złączenie prawostronne

NazwaKlienta	NrKonta	NazwaBanku
FH Klin SA	12345678901234567892022222	Bank BPH
FH Klin SA	[NULL]	Bank Niemiecki
FH Klin SA	[NULL]	Bank Nowy
FH Klin SA	43527897963543645632726336	Bank Polski
Firma Krok Sp zoo	78789798979879879877878978	Bank BPH
Firma Krok Sp zoo	46748329374637843254632546	Bank Niemiecki
Firma Krok Sp zoo	[NULL]	Bank Nowy
Firma Krok Sp zoo	[NULL]	Bank Polski
PHPU OSA	[NULL]	Bank BPH
PHPU OSA	[NULL]	Bank Niemiecki
PHPU OSA	[NULL]	Bank Nowy
PHPU OSA	[NULL]	Bank Polski
Rower Polska SA	[NULL]	Bank BPH
Rower Polska SA	67686868768348364836483764	Bank Niemiecki
Rower Polska SA	[NULL]	Bank Nowy
Rower Polska SA	[NULL]	Bank Polski
STALHANDEL	98087079643906432786443324	Bank BPH
STALHANDEL	[NULL]	Bank Niemiecki
STALHANDEL	[NULL]	Bank Nowy
STALHANDEL	67876864376438209876473674	Bank Polski

# Podzapytania

---

- Wewnątrz klauzul WHERE i HAVING, a także SELECT i FROM, mogą wystąpić podzapytania, mające taką samą postać jak zapytania (tylko są ujęte w nawiasy).
- W podzapytaniu dostępne są nazwy kolumn wprowadzone w głównym zapytaniu.
- Podzapytanie nazywamy *zwykłym* jeśli nie zawiera odwołań do kolumn tabel określonych w głównym zapytaniu.
- Podzapytanie nazywamy *skorelowanym* jeśli zawiera odwołania do kolumn tabel określonych w głównym zapytaniu.
- W podzapytaniu zwykłym zbiór wynikowych wierszy nie zmienia się i nie zależy od wierszy w głównym zapytaniu.

- Wypisz osoby, które zarabiają najwięcej ze wszystkich pracowników.
- najpierw liczymy największą pensję za pomocą zapytania:

```
SELECT max(pensja) FROM pracownicy;
```

- Zapytanie to można z kolei użyć jako podzapytanie (bez średnika) w warunku WHERE, wtedy kiedy trzeba przyrównać zarobki pracownika do maksymalnych zarobków. W efekcie uzyskujemy możliwość wyszukania pracowników, których zarobki są równe tym maksymalnym.

```
SELECT nazwisko, pensja  
FROM pracownicy  
WHERE pensja = (SELECT max(pensja) FROM pracownicy);
```

W klauzuli WHERE może być więcej niż jedno podzapytanie.

## Zagnieżdżone podzapytania

---

```
SELECT      Towar.NazwaTowaru, LiniaZamowienia.Ilosc
FROM        LiniaZamowienia INNER JOIN Towar ON
LiniaZamowienia.IdTowaru = Towar.IdTowaru
WHERE       (LiniaZamowienia.IdZamowienia IN
            (SELECT      IdZamowienia
              FROM        Zamowienie
              WHERE       DataZamowienia = '2004-04-04' AND IdKlienta =
            (SELECT      IdKlienta
              FROM        Klient
              WHERE       NazwaKlienta = 'FH Klin SA'))))
```

## Zapytanie nieskorelowane

---

- wszystkie zapytania wewnętrzne mogą być wykonane przed realizacją zapytania zewnętrznego
- tego rodzaju zapytanie jest szybsze od operacji złączenia

## Zapytanie nieskorelowane

```
SELECT NazwaTowaru, Ilosc
FROM Towar JOIN LiniaZamowienia USING (IdTowaru) JOIN
Zamowienie USING (IdZamowienia) JOIN Klient Using
(IdKlienta)
WHERE DataZamowienia = '2004-04-04' AND NazwaKlienta =
'FH Klin SA'
```

NazwaTowaru	Ilosc
Rura zgrz. fi 6,3 gr 0,2	20
Rura zgrz. fi 12,6 gr 0,2	12
Rura zgrz. fi 6,3 gr 0,3	25

## Zapytania zawierające unię

---

```
SELECT NazwaKlienta, NazwaTowaru, Ilosc
FROM Towar JOIN LiniaZamowienia USING (IdTowaru) JOIN Zamowienie
USING (IdZamowienia) JOIN Klient Using (IdKlienta)
WHERE NazwaTowaru = 'Rura zgrz. fi 6,3 gr 0,3'
```

### **UNION**

```
SELECT NazwaKlienta, NazwaTowaru, Ilosc
FROM Towar JOIN LiniaZamowienia USING (IdTowaru) JOIN Zamowienie
USING (IdZamowienia) JOIN Klient Using (IdKlienta)
WHERE NazwaTowaru = 'Rura zgrz. fi 12,6 gr 0,2'
```

## Zapytania zawierające unię

```
SELECT NazwaKlienta, NazwaTowaru, Ilosc
FROM Towar JOIN LiniaZamowienia USING (IdTowaru) JOIN
Zamowienie USING (IdZamowienia) JOIN Klient Using
(IdKlienta)
WHERE NazwaTowaru = 'Rura zgrz. fi 6,3 gr 0,3' OR
NazwaTowaru = 'Rura zgrz. fi 12,6 gr 0,2'
```

NazwaKlienta	NazwaTowaru	Ilosc
FH Klin SA	Rura zgrz. fi 6,3 gr 0,3	25
STALHANDEL	Rura zgrz. fi 6,3 gr 0,3	50
FH Klin SA	Rura zgrz. fi 12,6 gr 0,2	12
Rower Polska SA	Rura zgrz. fi 12,6 gr 0,2	50



## Zapytania zawierające unię

---

```
SELECT NazwaKlienta
FROM Towar JOIN LiniaZamowienia USING (IdTowaru) JOIN Zamowienie
USING (IdZamowienia) JOIN Klient Using (IdKlienta)
WHERE NazwaTowaru = 'Rura zgrz. fi 6,3 gr 0,3'
```

### **UNION**

```
SELECT NazwaKlienta
FROM Towar JOIN LiniaZamowienia USING (IdTowaru) JOIN Zamowienie
USING (IdZamowienia) JOIN Klient Using (IdKlienta)
WHERE NazwaTowaru = 'Rura zgrz. fi 12,6 gr 0,2'
```

# Zapytania zawierające unię

---

```
+-----+  
| NazwaKlienta |  
+-----+  
| FH Klin SA   |  
| STALHANDEL  |  
| Rower Polska SA |  
+-----+
```

## Zapytania zawierające unię

---

```
SELECT NazwaKlienta
FROM Towar JOIN LiniaZamowienia USING (IdTowaru) JOIN
Zamowienie USING (IdZamowienia) JOIN Klient Using
(IdKlienta)
WHERE NazwaTowaru = 'Rura zgrz. fi 6,3 gr 0,3' OR
NazwaTowaru = 'Rura zgrz. fi 12,6 gr 0,2'
```

```
+-----+
| NazwaKlienta |
+-----+
| FH Klin SA   |
| FH Klin SA   |
| STALHANDEL   |
| Rower Polska SA |
+-----+
```

## Zapytania zawierające unię

---

```
SELECT DISTINCT NazwaKlienta
FROM Towar JOIN LiniaZamowienia USING (IdTowaru) JOIN
Zamowienie USING (IdZamowienia) JOIN Klient Using
(IdKlienta)
WHERE NazwaTowaru = 'Rura zgrz. fi 6,3 gr 0,3' OR
NazwaTowaru = 'Rura zgrz. fi 12,6 gr 0,2'
```

```
+-----+
| NazwaKlienta |
+-----+
| FH Klin SA   |
| STALHANDEL   |
| Rower Polska SA |
+-----+
```

## Funkcje agregujące (1)

---

– **Funkcje agregujące** SQL operują na zbiorach wartości (np. kolumna relacji) i obliczają pojedynczą wartość. Są to:

**AVG** - wartość średnia,

**MIN** - wartość minimalna,

**MAX** - wartość maksymalna,

**SUM** – suma,

**COUNT** – liczność zbioru.

» Znajdź ilość krotek w relacji *pracownicy*.

```
SELECT COUNT (*)  
FROM pracownicy;
```

## Funkcje agregujące (2)

- Jeżeli funkcja agregująca ma **ignorować duplikaty**, stosuje się klauzulę **DISTINCT**.
- Funkcje agregujące mogą być zastosowane do grup krotek. Uzyskuje się to przez zastosowanie klauzuli **GROUP BY** i odpowiednie uformowanie klauzuli **SELECT**.

» Znajdź ilość pracowników w każdym oddziale firmy.

```
SELECT nazwa_oddzialu, COUNT (DISTINCT nazwisko)  
FROM pracownicy, oddzialy  
WHERE pracownicy.id_oddzialu=oddzialy.id_oddzialu  
GROUP BY nazwa_oddzialu
```

Uwaga: atrybuty z klauzuli **SELECT** poza funkcją agregującą muszą wystąpić w liście **GROUP BY**.

## Funkcje agregujące (3)

- Funkcje agregujące mogą być użyte do nakładania **warunków na grupy krotek**. Wówczas stosuje się rozwinięcie klauzuli **GROUP BY** o postaci: **HAVING** funkcja agregująca.
  - » Znajdź nazwy wszystkich oddziałów, gdzie średnia płaca jest większa niż 1,200zł

```
SELECT nazwa_oddzialu, AVG (placa)
FROM pracownicy, oddzialy
WHERE pracownicy.id_oddzialu=oddzialy.id_oddzialu
GROUP BY nazwa_oddzialu
HAVING AVG (placa) > 1200;
```

Uwaga: warunki z klauzuli **HAVING** są stosowane po uformowaniu grup.

# Zbiór wejściowy

NazwaTowaru	Data	Ilosc	Cena
Rura zgrz. fi 12,6 gr 0,2	2004 04 04	12	1.75
Rura zgrz. fi 12,6 gr 0,2	2004 04 07	50	1.75
Rura zgrz. fi 12,6 gr 0,3	2004 04 06	12	2.05
Rura zgrz. fi 6,3 gr 0,2	2004 04 05	100	1.40
Rura zgrz. fi 6,3 gr 0,2	2004 04 04	20	1.50
Rura zgrz. fi 6,3 gr 0,2	2004 04 06	50	1.50
Rura zgrz. fi 6,3 gr 0,3	2004 04 04	25	2.10
Rura zgrz. fi 6,3 gr 0,3	2004 04 06	50	2.10
Rura zgrz. kw 4 gr 0,2	2004 04 05	6	2.20
Rura zgrz. kw 4 gr 0,2	2004 04 07	30	2.20
Rura zgrz. kw 5 gr 0,3	[NULL]	[NULL]	[NULL]
Zlaczka 1'	2004 04 06	100	0.90
Zlaczka 2'	2004 04 06	50	1.10
Zlaczka 3/4'	[NULL]	[NULL]	[NULL]



# Funkcja COUNT

---

```
SELECT COUNT(*) AS Liczba
FROM Towar LEFT JOIN LiniaZamowienia USING (IdTowaru) LEFT JOIN
Zamowienie USING (IdZamowienia)
ORDER BY NazwaTowaru
```

14

```
SELECT COUNT(Ilosc) AS Liczba
FROM Towar LEFT JOIN LiniaZamowienia USING (IdTowaru) LEFT JOIN
Zamowienie USING (IdZamowienia)
```

12

```
SELECT COUNT(DISTINCT NazwaTowaru) AS Liczba
FROM Towar LEFT JOIN LiniaZamowienia USING (IdTowaru) LEFT JOIN
Zamowienie USING (IdZamowienia)
ORDER BY NazwaTowaru
```

9

## Pozostałe funkcje

---

```
SELECT SUM(Ilosc*LiniaZamowienia.Cena)
FROM Towar LEFT JOIN LiniaZamowienia USING (IdTowaru) LEFT JOIN
Zamowienie USING (IdZamowienia)
```

759,8

```
SELECT AVG(LiniaZamowienia.Cena)
FROM Towar LEFT JOIN LiniaZamowienia USING (IdTowaru)
WHERE Towar.IdTowaru=1
```

1,46667

```
SELECT MAX( DATE_FORMAT(DataZamowienia, '%Y %m %d') ) AS Data
FROM Towar LEFT JOIN LiniaZamowienia USING (IdTowaru) LEFT JOIN
Zamowienie USING (IdZamowienia)
ORDER BY NazwaTowaru
```

2004 04 07

## Zapytania grupujące

```

SELECT DATE_FORMAT(DataZamowienia, '%Y %m %d') AS Data,
Towar.NazwaTowaru, SUM(Ilosc) AS Ilosc , SUM(Ilosc*Cena) AS Wartosc
FROM Zamowienie JOIN LiniaZamowienia USING (IdZamowienia) JOIN
Towar USING ( IdTowaru)
GROUP BY DataZamowienia
ORDER BY DataZamowienia
  
```

Data	NazwaTowaru	Ilosc	Wartosc
2004 04 04	Rura zgrz. fi 6,3 gr 0,2	57	103.5
2004 04 05	Rura zgrz. fi 6,3 gr 0,2	106	153.2
2004 04 06	Rura zgrz. fi 6,3 gr 0,2	262	349.6
2004 04 07	Rura zgrz. fi 12,6 gr 0,2	80	153.5

## Zapytania grupujące

---

```
SELECT NazwaKlienta, DATE_FORMAT(DataZamowienia, '%Y %m %d') AS
Data, SUM(Ilosc*Cena) AS Wartosc
FROM Klient JOIN Zamowienie USING (IdKlienta) JOIN
LiniaZamowienia USING (IdZamowienia)
GROUP BY NazwaKlienta
ORDER BY NazwaKlienta, DataZamowienia
```

```
+-----+-----+-----+
| NazwaKlienta      | Data          | Wartosc |
+-----+-----+-----+
| FH Klin SA        | 2004 04 04   | 203.1 |
| Firma Krok Sp zoo | 2004 04 05   | 153.2 |
| Rower Polska SA   | 2004 04 07   | 153.5 |
| STALHANDEL        | 2004 04 06   | 250 |
+-----+-----+-----+
```

## Zapytania grupujące

```
SELECT      NazwaKlienta, DATE_FORMAT(DataZamowienia, '%Y %m %d')
AS Data, SUM(Ilosc*Cena) AS Wartosc
FROM Klient JOIN Zamowienie USING (IdKlienta) JOIN
LiniaZamowienia USING (IdZamowienia)
GROUP BY NazwaKlienta, DataZamowienia
ORDER BY NazwaKlienta, DataZamowienia
```

NazwaKlienta	Data	Wartosc
FH Klin SA	2004 04 04	103.5
FH Klin SA	2004 04 06	99.6
Firma Krok Sp zoo	2004 04 05	153.2
Rower Polska SA	2004 04 07	153.5
STALHANDEL	2004 04 06	250

## Zapytania grupujące - ograniczenia

```
SELECT      NazwaKlienta, DATE_FORMAT(DataZamowienia, '%Y %m %d')
AS Data, SUM(Ilosc*Cena) AS Wartosc
FROM        Klient JOIN Zamowienie USING (IdKlienta) JOIN
LiniaZamowienia USING (IdZamowienia)
WHERE DataZamowienia > '2004-04-04'
GROUP BY NazwaKlienta, DataZamowienia
ORDER BY NazwaKlienta, DataZamowienia
```

NazwaKlienta	Data	Wartosc
FH Klin SA	2004 04 06	99.6
Firma Krok Sp zoo	2004 04 05	153.2
Rower Polska SA	2004 04 07	153.5
STALHANDEL	2004 04 06	250

## Zapytania grupujące - ograniczenia

---

```
SELECT  NazwaKlienta, DATE_FORMAT(DataZamowienia, '%Y %m %d') AS
Data, SUM(Ilosc*Cena) AS Wartosc
FROM    Klient JOIN Zamowienie USING (IdKlienta) JOIN
LiniaZamowienia USING (IdZamowienia)
GROUP BY NazwaKlienta, DataZamowienia
HAVING Data > '2004 04 04'
ORDER BY NazwaKlienta, DataZamowienia
```

NazwaKlienta	Data	Wartosc
FH Klin SA	2004 04 06	99.6
Firma Krok Sp zoo	2004 04 05	153.2
Rower Polska SA	2004 04 07	153.5
STALHANDEL	2004 04 06	250

– konkatencja łańcuchów

## ***SQL-92***

```
SELECT KodPocztowy || ' ' || Miejscowosc)
FROM `klient`
```

## ***Microsoft SQL Server***

```
SELECT SymbolTowaru + ' ' + NazwaTowaru AS NowaNazwa
FROM Towar
```

## ***MySQL***

```
SELECT concat(KodPocztowy, ' ', Miejscowosc)
FROM `klient`
```



## Wybrane funkcje tekstowe

---

- **UPPER** i **LOWER** konwertują łańcuchy tekstowe na duże lub małe litery
- **TRIM** (słowo) – usuwa określone znaki z początku lub końca łańcucha znaków. Domyślnie usuwa spacje z obu stron. Ilość usuwanych znaków jest ograniczona do jednego.
  - » **TRIM** (**BOTH** znak **FROM** słowo)
  - » **TRIM** (**LEADING** znak **FROM** słowo)
  - » **TRIM** (**TRAILING** znak **FROM** słowo)
- **SUBSTRING** (słowo **FROM** poz\_startowa **FOR** liczba\_znaków)

KodPocztowy	Miejscowosc
30-121	Kraków
30-321	Kraków
34-876	Sosnowiec
32-082	Zabierzów
30-432	Kraków

```
SELECT SUBSTRING(TRIM(LEADING '3' FROM CONCAT(KodPocztowy, ' ',
Miejscowosc)) FROM 1 FOR 12) AS Miasto
FROM klient
WHERE UPPER(Miejscowosc) = 'KRAKÓW'
```

Miasto
0-121 Kraków
0-321 Kraków
0-432 Kraków

## Działania dotyczące czasu

```
SELECT NazwaKlienta, DATE_FORMAT(DataZamowienia, '%Y %m %d') AS  
Data1, CURRENT_DATE AS Data2, TO_DAYS(CURRENT_DATE) - TO_DAYS(  
DataZamowienia) Dni  
FROM Zamowienie JOIN Klient USING (IdKlienta)
```

NazwaKlienta	Data1	Data2	Dni
FH Klin SA	2004 04 04	2004-05-03	29
FH Klin SA	2004 04 06	2004-05-03	27
Firma Krok Sp zoo	2004 04 05	2004-05-03	28
STALHANDEL	2004 04 06	2004-05-03	27
Rower Polska SA	2004 04 07	2004-05-03	26

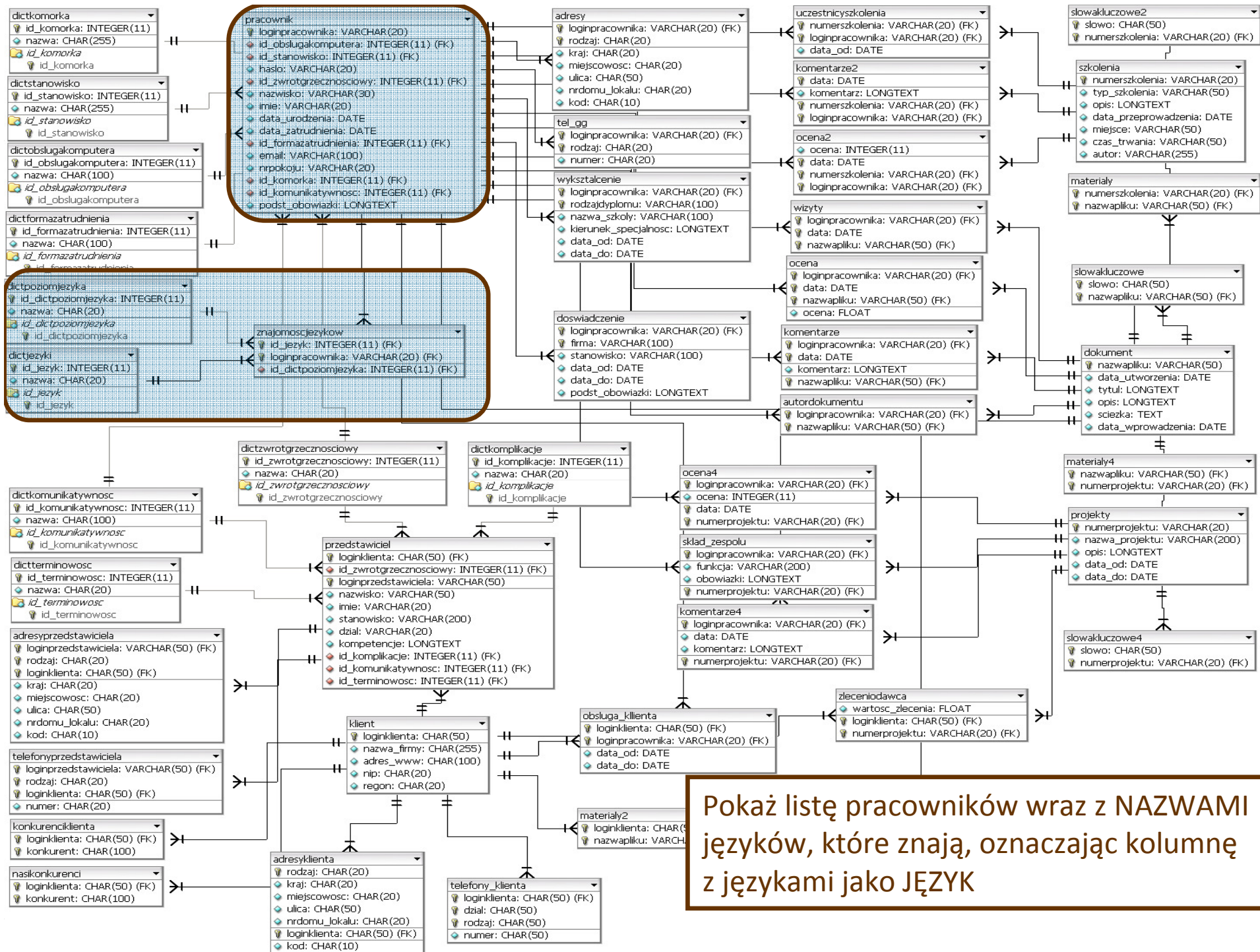
## Działania dotyczące czasu

---

```
SELECT NazwaKlienta, DATE_FORMAT(DataZamowienia, '%Y %m %d') AS  
DataFaktury, DATE_FORMAT(DATE_ADD(DataZamowienia, INTERVAL 14  
DAY), '%Y %m %d') AS TerminPlatnosci  
FROM Zamowienie JOIN Klient USING (IdKlienta)
```

NazwaKlienta	DataFaktury	TerminPlatnosci
FH Klin SA	2004 04 04	2004 04 18
FH Klin SA	2004 04 06	2004 04 20
Firma Krok Sp zoo	2004 04 05	2004 04 19
STALHANDEL	2004 04 06	2004 04 20
Rower Polska SA	2004 04 07	2004 04 21

# Przykłady zapytań

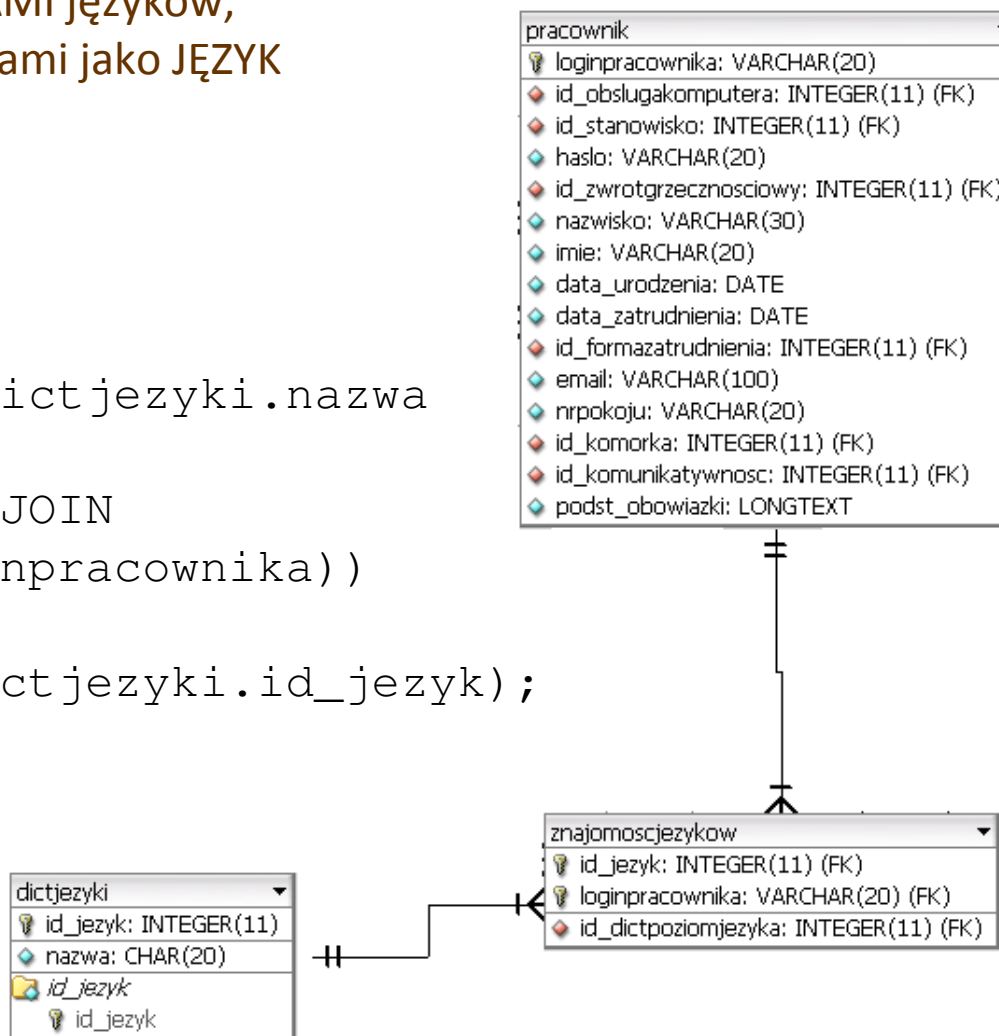


Pokaż listę pracowników wraz z NAZWAMI języków, które znają, oznaczając kolumnę z językami jako JĘZYK

# Zapytanie 1

Pokaż listę pracowników wraz z NAZWAMI języków, które znają, oznaczając kolumnę z językami jako JĘZYK

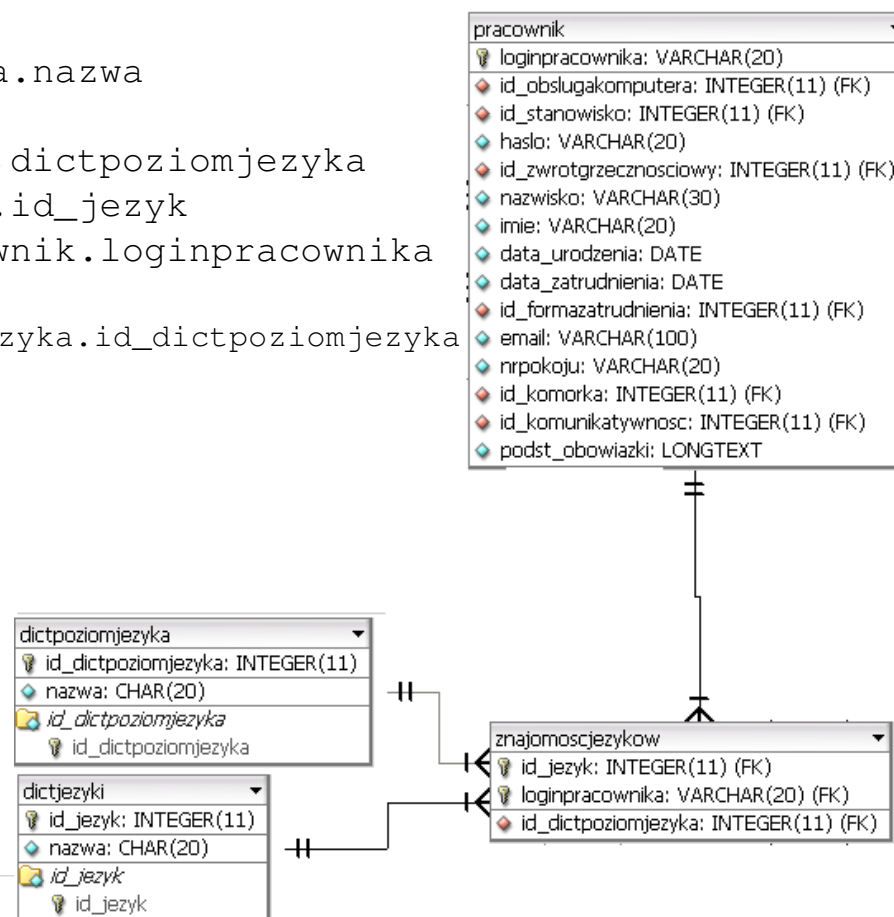
```
SELECT pracownik.nazwisko, dictjezyki.nazwa
AS JĘZYK
FROM ((pracownik LEFT OUTER JOIN
znajomoscjezykow USING (loginpracownika))
INNER JOIN dictjezyki ON
znajomoscjezykow.id_jezyk=dictjezyki.id_jezyk);
```



## Zapytanie 2

Znajdź wszystkich pracowników znających język angielski i pokaż stopień jego znajomości w kolumnie POZIOM ANGIELSKIEGO. Wyniki poukładaj alfabetycznie (wg nazwisk).

```
SELECT pracownik.nazwisko, dictpoziomjezyka.nazwa
AS 'POZIOM ANGIELSKIEGO'
FROM pracownik, znajomoscjezykow, dictjezyki, dictpoziomjezyka
WHERE dictjezyki.id_jezyk=znajomoscjezykow.id_jezyk
AND znajomoscjezykow.loginpracownika=pracownik.loginpracownika
AND
znajomoscjezykow.id_dictpoziomjezyka=dictpoziomjezyka.id_dictpoziomjezyka
AND znajomoscjezykow.id_jezyk=1
ORDER BY pracownik.nazwisko;
```





## Zapytanie 3

Znajdź średnią ocenę plików zawierających w nazwie słowo „opis” i umieść ją w kolumnie „Średnia ocena”.

```

select nazwapliku, avg(ocena) as 'Średnia ocena'
from ocena
group by nazwapliku
having nazwapliku like '%opis%';

```

