




**Studia podyplomowe ”Inżynieria oprogramowania” współfinansowane przez Unię Europejską w ramach Europejskiego Funduszu Społecznego**

Projekt ”Studia podyplomowe z zakresu wytwarzania oprogramowania oraz zarządzania projektami w firmach informatycznych” realizowany w ramach Programu Operacyjnego Kapitał Ludzki

# Konstruowanie Baz Danych

# DQL — join. Złączenia tablic

Antoni Ligęza
ligeza@agh.edu.pl
http://home.agh.edu.pl/~ligeza
http://home.agh.edu.pl/~ligeza/wiki

# Bazy Danych

**Wykład p.t.**

# Złączenia tablic:

# FROM, WHERE, JOIN

**Antoni Ligęza**

ligeza@agh.edu.pl
http://galaxy.uci.agh.edu.pl/~ligeza

Wykorzystano materiały:
http://www.postgresql.org/docs/8.3/interactive/index.html

# Komendy SQL: SELECT

```
SELECT [ ALL | DISTINCT [ ON ( expression [, ...] ) ] ]
    * | expression [ AS output_name ] [, ...]
    [ FROM from_item [, ...] ]
    [ WHERE condition ]
    [ GROUP BY expression [, ...] ]
    [ HAVING condition [, ...] ]
    [ { UNION | INTERSECT | EXCEPT }
[ ALL ] select ]
    [ ORDER BY expression
[ ASC | DESC | USING operator ] [, ...] ]
    [ LIMIT { count | ALL } ]
    [ OFFSET start ]
    [ FOR { UPDATE | SHARE }
[ OF table_name [, ...] ] [ NOWAIT ] [...] ]

where from_item can be one of:

    [ ONLY ] table_name [ * ]
[ [ AS ] alias [ ( column_alias [, ...] ) ] ]
    ( select ) [ AS ] alias [ ( column_alias [, ...] ) ]
    function_name ( [ argument [, ...] ] )
[ AS ] alias [ ( column_alias [, ...]
| column_definition [, ...] ) ]
    function_name ( [ argument [, ...] ] )
AS ( column_definition [, ...] )
    from_item [ NATURAL ] join_type from_item
[ ON join_condition | USING ( join_column [, ...] ) ]
```

## Proste łączenie tablic

`WHERE tablica1.atrybut1=tablica2.atrybut2` – proste łączenie można wykonać poprzez porównanie pól z dwóch tablic. Najczęściej jest to łączenie klucz-klucz obcy.

```
pracownicy=> SELECT id_prac,nazwisko, imie, nazwa,id_dzial
        FROM dzial, prac WHERE dzial.kierownik=prac.id_prac;
 id_prac | nazwisko  |   imie   |    nazwa    | id_dzial
---------+-----------+----------+-------------+----------
 110     | Kowalik   | Artur    | Produkcyjny | PD303
 101     | Kowalczyk | Amadeusz | Projektowy  | PK101
 111     | Kowalczuk | Adam     | Promocji    | PR202
(3 rows)

 id_prac |  nazwisko  |   imie   |    nazwa    | id_dzial
---------+------------+----------+-------------+----------
 1100    | Kowal      | Adam     | Produkcyjny | PD303
 110     | Kowalik    | Artur    | Produkcyjny | PD303
 1110    | Kowalewski | Adam     | Promocji    | PR202
 101     | Kowalczyk  | Amadeusz | Projektowy  | PK101
 1101    | Kowalski   | Antoni   | Produkcyjny | PD303
 1011    | Kowalowski | Alojzy   | Projektowy  | PK101
 111     | Kowalczuk  | Adam     | Promocji    | PR202
 1010    | Kawula     | Alojzy   | Projektowy  | PK101
 1111    | kudryk     | adam     | Produkcyjny | PD303
(9 rows)
```

## Uwaga: iloczyn kartezjański

Uwaga: bez (sensownego) warunku łączenia powstaje iloczyn kartezjański!

```
pracownicy=> SELECT * FROM dzial, prac;
 id_dzial |    nazwa    | lokalizacja | kierownik | id_prac |  nazwisko  |   imie
----------+-------------+-------------+-----------+---------+------------+----------
 PD303    | Produkcyjny | Mysiecko    | 110       | 1100    | Kowal      | Adam
 PD303    | Produkcyjny | Mysiecko    | 110       | 110     | Kowalik    | Artur
 PD303    | Produkcyjny | Mysiecko    | 110       | 1110    | Kowalewski | Adam
 PD303    | Produkcyjny | Mysiecko    | 110       | 101     | Kowalczyk  | Amadeusz
 PD303    | Produkcyjny | Mysiecko    | 110       | 1101    | Kowalski   | Antoni
 PD303    | Produkcyjny | Mysiecko    | 110       | 1011    | Kowalowski | Alojzy
 PD303    | Produkcyjny | Mysiecko    | 110       | 111     | Kowalczuk  | Adam
 PD303    | Produkcyjny | Mysiecko    | 110       | 1010    | Kawula     | Alojzy
 PD303    | Produkcyjny | Mysiecko    | 110       | 1111    | kudryk     | adam
 PK101    | Projektowy  | Mysieko     | 101       | 1100    | Kowal      | Adam
 PK101    | Projektowy  | Mysieko     | 101       | 110     | Kowalik    | Artur
 PK101    | Projektowy  | Mysieko     | 101       | 1110    | Kowalewski | Adam
 PK101    | Projektowy  | Mysieko     | 101       | 101     | Kowalczyk  | Amadeusz
 PK101    | Projektowy  | Mysieko     | 101       | 1101    | Kowalski   | Antoni
 PK101    | Projektowy  | Mysieko     | 101       | 1011    | Kowalowski | Alojzy
 PK101    | Projektowy  | Mysieko     | 101       | 111     | Kowalczuk  | Adam
 PK101    | Projektowy  | Mysieko     | 101       | 1010    | Kawula     | Alojzy
 PK101    | Projektowy  | Mysieko     | 101       | 1111    | kudryk     | adam
 PR202    | Promocji    | Mysieoczko  | 111       | 1100    | Kowal      | Adam
 PR202    | Promocji    | Mysieoczko  | 111       | 110     | Kowalik    | Artur
 PR202    | Promocji    | Mysieoczko  | 111       | 1110    | Kowalewski | Adam
 PR202    | Promocji    | Mysieoczko  | 111       | 101     | Kowalczyk  | Amadeusz
 PR202    | Promocji    | Mysieoczko  | 111       | 1101    | Kowalski   | Antoni
 PR202    | Promocji    | Mysieoczko  | 111       | 1011    | Kowalowski | Alojzy
 PR202    | Promocji    | Mysieoczko  | 111       | 111     | Kowalczuk  | Adam
 PR202    | Promocji    | Mysieoczko  | 111       | 1010    | Kawula     | Alojzy
 PR202    | Promocji    | Mysieoczko  | 111       | 1111    | kudryk     | adam
(27 rows)
```

## Możliwości łączenia tablic w SQL

Jeżeli informacja jest pobierana z wielu tablic to specyfikacja tych tablic oraz sposobu łączenia tablic umieszczona jest w klauzuli `FROM` (oraz ew. `WHERE`)zapytania `SELECT`. Bazowa konstrukcja `FROM`:

```
FROM table_reference [, table_reference [, ...]]
```

`table_reference` – to najczęściej nazwa tabeli; może to być także wynik złączenia tablic, podzapytanie, lub kombinacja powyższych. W praktyce może to być widok (perspektywa).

Istnieją dwie możliwości łączenia tablic:

- `FROM` – specyfikacja tablic oddzielonych przecinkami; warunki złączenia w `WHERE`,
- `FROM` – pełna specyfikacja łączenia tablic (typu i warunku złączenia) z wykorzystaniem słowa `JOIN`; w warunku `WHERE` pozostaja kryteria wyboru.

Najczęściej z wykorzystaniem warunku łączenia w `WHERE` wykonuje się złączenie wg schematu klucz–klucz obcy.

Uwaga!!! Podanie w `FROM` dwóch lub więcej tablic oddzielonych przecinkami bez specyfikacji warunku łączenia tworzy iloczyn kartezjański!!!

## Tablice

```
pracownicy=> SELECT * FROM dzial;
 id_dzial |    nazwa    | lokalizacja | kierownik
----------+-------------+-------------+-----------
 PD303    | Produkcyjny | Mysiecko    | 110
 PK101    | Projektowy  | Mysieko     | 101
 PR202    | Promocji    | Mysieoczko  | 111
(3 rows)

pracownicy=> SELECT id_prac,nazwisko,dzial,
                    stanowisko FROM prac;
 id_prac | nazwisko  | dzial | stanowisko
---------+-----------+-------+------------
 1100    | Kowal     | PD303 | robotnik
 110     | Kowalik   | PD303 | kierownik
 101     | Kowalczyk | PK101 | kierownik
 1101    | Kowalski  | PD303 | robotnik
 111     | Kowalczuk | PR202 | kierownik
 1010    | Kawula    | PK101 | robotnik
(6 rows)
```

## Iloczyn kartezjański

```
pracownicy=> SELECT id_dzial,nazwa,kierownik,
                id_prac,dzial FROM dzial,prac;
 id_dzial |    nazwa    | kierownik | id_prac | dzial
----------+-------------+-----------+---------+-------
 PD303    | Produkcyjny | 110       | 1100    | PD303
 PD303    | Produkcyjny | 110       | 110     | PD303
 PD303    | Produkcyjny | 110       | 101     | PK101
 PD303    | Produkcyjny | 110       | 1101    | PD303
 PD303    | Produkcyjny | 110       | 111     | PR202
 PD303    | Produkcyjny | 110       | 1010    | PK101
 PK101    | Projektowy  | 101       | 1100    | PD303
 PK101    | Projektowy  | 101       | 110     | PD303
 PK101    | Projektowy  | 101       | 101     | PK101
 PK101    | Projektowy  | 101       | 1101    | PD303
 PK101    | Projektowy  | 101       | 111     | PR202
 PK101    | Projektowy  | 101       | 1010    | PK101
 PR202    | Promocji    | 111       | 1100    | PD303
 PR202    | Promocji    | 111       | 110     | PD303
 PR202    | Promocji    | 111       | 101     | PK101
 PR202    | Promocji    | 111       | 1101    | PD303
 PR202    | Promocji    | 111       | 111     | PR202
 PR202    | Promocji    | 111       | 1010    | PK101
(18 rows)
```

## WHERE: dział-kierownik

Podaj dane działów i kierownika każdego działu.

```
pracownicy=> SELECT id_dzial,nazwa,kierownik AS kier,
                    id_prac,dzial,nazwisko
                    FROM dzial d, prac p

WHERE d.kierownik=p.id_prac;
 id_dzial |    nazwa    | kier  | id_prac | dzial | nazwisko
----------+-------------+-------+---------+-------+-----------
 PD303    | Produkcyjny | 110   | 110     | PD303 | Kowalik
 PK101    | Projektowy  | 101   | 101     | PK101 | Kowalczyk
 PR202    | Promocji    | 111   | 111     | PR202 | Kowalczuk
(3 rows)

pracownicy=> SELECT id_prac,nazwisko,dzial,
                      stanowisko FROM prac;

 id_prac | nazwisko  | dzial | stanowisko
---------+-----------+-------+------------
 1100    | Kowal     | PD303 | robotnik
 110     | Kowalik   | PD303 | kierownik
 101     | Kowalczyk | PK101 | kierownik
 1101    | Kowalski  | PD303 | robotnik
 111     | Kowalczuk | PR202 | kierownik
 1010    | Kawula    | PK101 | robotnik
(6 rows)
```

## WHERE: pracownik-dział

W jakich działach pracują poszczególni pracownicy?

```
pracownicy=> SELECT id_prac,nazwisko,dzial,id_dzial,nazwa
                FROM prac p, dzial d
                WHERE p.dzial=d.id_dzial;
 id_prac | nazwisko  | dzial | id_dzial |    nazwa
---------+-----------+-------+----------+-------------
 1100    | Kowal     | PD303 | PD303    | Produkcyjny
 110     | Kowalik   | PD303 | PD303    | Produkcyjny
 101     | Kowalczyk | PK101 | PK101    | Projektowy
 1101    | Kowalski  | PD303 | PD303    | Produkcyjny
 111     | Kowalczuk | PR202 | PR202    | Promocji
 1010    | Kawula    | PK101 | PK101    | Projektowy
(6 rows)
pracownicy=> SELECT id_prac,nazwisko,dzial FROM prac;
 id_prac | nazwisko  | dzial
---------+-----------+-------
 1100    | Kowal     | PD303
 110     | Kowalik   | PD303
 101     | Kowalczyk | PK101
 1101    | Kowalski  | PD303
 111     | Kowalczuk | PR202
 1010    | Kawula    | PK101
(6 rows)
pracownicy=> SELECT id_dzial, nazwa  FROM dzial;
id_dzial |    nazwa
----------+-------------
 PD303    | Produkcyjny
 PK101    | Projektowy
 PR202    | Promocji
```

## Przykład

Kto zarabia więcej niż jego kierownik?

```
pracownicy=> \i pobory-prac-kier.sql

 id_prac | nazwisko | pobory  | nazwisko  | pobory
---------+----------+---------+-----------+---------
 1101    | Kowalski | 4500.00 | Kowalik   | 1500.00
 1010    | Kawula   | 2500.00 | Kowalczyk | 1000.00
(2 rows)

pracownicy=> SELECT id_prac,nazwisko,pobory,
                  dzial,stanowisko FROM prac;

 id_prac | nazwisko  | pobory  | dzial | stanowisko
---------+-----------+---------+-------+------------
 1100    | Kowal     | 1500.00 | PD303 | robotnik
 110     | Kowalik   | 1500.00 | PD303 | kierownik
 101     | Kowalczyk | 1000.00 | PK101 | kierownik
 1101    | Kowalski  | 4500.00 | PD303 | robotnik
 111     | Kowalczuk | 2500.00 | PR202 | kierownik
 1010    | Kawula    | 2500.00 | PK101 | robotnik
(6 rows)

pracownicy=> SELECT * FROM dzial;
 id_dzial |    nazwa    | lokalizacja | kierownik
----------+-------------+-------------+-----------
 PD303    | Produkcyjny | Mysiecko    | 110
 PK101    | Projektowy  | Mysieko     | 101
 PR202    | Promocji    | Mysieoczko  | 111
(3 rows)
```

## Rozwiązanie

```
SELECT p.id_prac, p.nazwisko, p.pobory,
                  k.nazwisko, k.pobory
   FROM prac p, dzial d, prac k
   WHERE p.dzial=d.id_dzial AND
         d.kierownik=k.id_prac
         AND
         p.pobory > k.pobory;
```

## JOIN – specyfikacja złączenia

`FROM T1 CROSS JOIN T2` – złączenie krzyżowe (iloczyn kartezjański; równoważne formy:
`FROM T1, T2` oraz `FROM T1 INNER JOIN T2 ON TRUE`.

Złączenia kwalifikowane:

Złączenie ze względu na warunek ($\theta$-złączenie):

```
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] }
                    JOIN T2 ON boolean_expression
```

Równozłączenie według wybranych kolumn (o identycznych nazwach):

```
T1 { [INNER] | { LEFT | RIGHT | FULL } [OUTER] }
                    JOIN T2 USING ( join column list )
```

Złączenie naturalne:

```
T1 NATURAL { [INNER] | { LEFT | RIGHT | FULL } [OUTER] }
                    JOIN T2
```

- `INNER JOIN` – złączenie wewnętrzne,
- `OUTER JOIN` – złączenie zewnętrzne,
- `LEFT OUTER JOIN` – lewostronne złączenie zewnętrzne,
- `RIGHT OUTER JOIN` – prawostronne złączenie zewnętrzne,
- `FULL OUTER JOIN` – złączenie zewnętrzne pełne,

## PostgreSQL doc – przyłady wyjaśniające

```
 num | name
-----+------
   1 | a
   2 | b
   3 | c
```

and t2:

```
 num | value
-----+-------
   1 | xxx
   3 | yyy
   5 | zzz
```

then we get the following results for the various joins:

```
=> SELECT * FROM t1 CROSS JOIN t2;
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   1 | a    |   3 | yyy
   1 | a    |   5 | zzz
   2 | b    |   1 | xxx
   2 | b    |   3 | yyy
   2 | b    |   5 | zzz
   3 | c    |   1 | xxx
   3 | c    |   3 | yyy
   3 | c    |   5 | zzz
(9 rows)

=> SELECT * FROM t1 INNER JOIN t2 ON t1.num = t2.num;
```

```
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   3 | c    |   3 | yyy
(2 rows)

=> SELECT * FROM t1 INNER JOIN t2 USING (num);
 num | name | value
-----+------+-------
   1 | a    | xxx
   3 | c    | yyy
(2 rows)

=> SELECT * FROM t1 NATURAL INNER JOIN t2;
 num | name | value
-----+------+-------
   1 | a    | xxx
   3 | c    | yyy
(2 rows)

=> SELECT * FROM t1 LEFT JOIN t2 ON t1.num = t2.num;
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   2 | b    |     |
   3 | c    |   3 | yyy
(3 rows)

=> SELECT * FROM t1 LEFT JOIN t2 USING (num);
 num | name | value
-----+------+-------
   1 | a    | xxx
   2 | b    |
   3 | c    | yyy
(3 rows)
```

```
=> SELECT * FROM t1 RIGHT JOIN t2 ON t1.num = t2.num;
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   3 | c    |   3 | yyy
     |      |   5 | zzz
(3 rows)

=> SELECT * FROM t1 FULL JOIN t2 ON t1.num = t2.num;
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   2 | b    |     |
   3 | c    |   3 | yyy
     |      |   5 | zzz
(4 rows)

The join condition specified with ON can also contain conditions

=> SELECT * FROM t1 LEFT JOIN t2 ON t1.num = t2.num AND t2.value
 num | name | num | value
-----+------+-----+-------
   1 | a    |   1 | xxx
   2 | b    |     |
   3 | c    |     |
(3 rows)
```

## JOIN ON: dział-prac

Podaj działy i zatrudnionych w nich pracowników.
W tablicy dzial dodano 2 nowe działy (ale bez pracowników).

```
pracownicy=> SELECT * FROM dzial;

 id_dzial |    nazwa     | lokalizacja | kierownik
----------+--------------+-------------+-----------
 PD303    | Produkcyjny  | Mysiecko    | 110
 PK101    | Projektowy   | Mysieko     | 101
 PR202    | Promocji     | Mysieoczko  | 111
 PZ404    | Zaopatrzenia | Myszków     |
 PZ505    | Zbytu        | Maszków     |
(5 rows)

pracownicy=> SELECT id_dzial,nazwa,id_prac,dzial,nazwisko
           FROM dzial d JOIN prac p ON d.id_dzial=p.dzial;

 id_dzial |    nazwa    | id_prac | dzial | nazwisko
----------+-------------+---------+-------+-----------
 PD303    | Produkcyjny | 1100    | PD303 | Kowal
 PD303    | Produkcyjny | 110     | PD303 | Kowalik
 PK101    | Projektowy  | 101     | PK101 | Kowalczyk
 PD303    | Produkcyjny | 1101    | PD303 | Kowalski
 PR202    | Promocji    | 111     | PR202 | Kowalczuk
 PK101    | Projektowy  | 1010    | PK101 | Kawula
(6 rows)
```

## INNER JOIN – złączenie wewnętrzne

```
pracownicy=> SELECT id_dzial,nazwa,id_prac,dzial,nazwisko
     FROM dzial d INNER JOIN prac p ON d.id_dzial=p.dzial;

id_dzial |    nazwa    | id_prac | dzial | nazwisko
----------+-------------+---------+-------+-----------
 PD303    | Produkcyjny | 1100    | PD303 | Kowal
 PD303    | Produkcyjny | 110     | PD303 | Kowalik
 PK101    | Projektowy  | 101     | PK101 | Kowalczyk
 PD303    | Produkcyjny | 1101    | PD303 | Kowalski
 PR202    | Promocji    | 111     | PR202 | Kowalczuk
 PK101    | Projektowy  | 1010    | PK101 | Kawula
(6 rows)
```

## OUTER JOIN – Złączenie zewnętrzne

Podaj *wszystkie* działy i zatrudnionych w nich pracowników.

```
pracownicy=> SELECT id_dzial,nazwa,id_prac,dzial,nazwisko
   FROM dzial d LEFT OUTER JOIN prac p ON d.id_dzial=p.dzial;

 id_dzial |    nazwa     | id_prac | dzial | nazwisko
----------+--------------+---------+-------+-----------
 PD303    | Produkcyjny  | 1101    | PD303 | Kowalski
 PD303    | Produkcyjny  | 110     | PD303 | Kowalik
 PD303    | Produkcyjny  | 1100    | PD303 | Kowal
 PK101    | Projektowy   | 1010    | PK101 | Kawula
 PK101    | Projektowy   | 101     | PK101 | Kowalczyk
 PR202    | Promocji     | 111     | PR202 | Kowalczuk
 PZ404    | Zaopatrzenia |         |       |
 PZ505    | Zbytu        |         |       |
(8 rows)
```

Podaj *wszystkie* działy w których nie zatrudniono pracowników.

```
pracownicy=> SELECT id_dzial,nazwa,id_prac,dzial,nazwisko
  FROM dzial d LEFT OUTER JOIN prac p ON d.id_dzial=p.dzial
  WHERE id_prac IS NULL;

 id_dzial |    nazwa     | id_prac | dzial | nazwisko
----------+--------------+---------+-------+----------
 PZ404    | Zaopatrzenia |         |       |
 PZ505    | Zbytu        |         |       |
(2 rows)
```

## Warunek ON – możliwości

Specyfikacja dowolnie złozonego warunku po ON daje szerokie możliwości łączenia i jego zastosowań/interpretacji.

```
pracownicy=> SELECT p.nazwisko, p.stanowisko, p.pobory,
                    q.nazwisko, q.stanowisko, q.pobory
                    FROM prac p JOIN prac q
                    ON p.stanowisko='kierownik'
                   AND q.stanowisko='robotnik'
                   AND q.pobory-p.pobory > 500;

 nazwisko | stanowisko |pobory| nazwisko |stanowisko|pobory
----------+------------+------+----------+----------+-------
 Kowalik  | kierownik  | 1500 | Kowalski | robotnik | 4500
 Kowalik  | kierownik  | 1500 | Kawula   | robotnik | 2500
 Kowalczyk| kierownik  | 1000 | Kowalski | robotnik | 4500
 Kowalczyk| kierownik  | 1000 | Kawula   | robotnik | 2500
 Kowalczuk| kierownik  | 2500 | Kowalski | robotnik | 4500
(5 rows)

pracownicy=> SELECT nazwisko, stanowisko, pobory FROM prac;
 nazwisko  | stanowisko | pobory
-----------+------------+---------
 Kowal     | robotnik   | 1500.00
 Kowalik   | kierownik  | 1500.00
 Kowalczyk | kierownik  | 1000.00
 Kowalski  | robotnik   | 4500.00
 Kowalczuk | kierownik  | 2500.00
 Kawula    | robotnik   | 2500.00
(6 rows)
```

## Brak łączności złączeń zewnętrznych

```
T1: studenci

id  |  nazwisko
----+-----------
17  | Kowal
18  | Malik
19  | Noga

T2: oceny

id  |  ocena
----+--------
17  |  2
17  |  3
19  |  4

T3: opis

ocena | opis
------+------
2     | ndst
3     | dst
4     | db
5     | bdb
```

```
(T1 LEFT JOIN T2) JOIN T3

id  |  nazwisko  |  ocena  |  opis
----+------------+---------+-------
17  |  Kowal     | 2       | ndst
17  |  Kowal     | 3       | dst
19  |  Noga      | 4       | db

T1 LEFT JOIN (T2 JOIN T3)

id  |  nazwisko  |  ocena  |  opis
----+------------+---------+-------
17  |  Kowal     | 2       | ndst
17  |  Kowal     | 3       | dst
19  |  Noga      | 4       | db
18  |  Malik     | NULL    | NULL
```


**Studia podyplomowe ”Inżynieria oprogramowania” współfinansowane przez Unię Europejską w ramach Europejskiego Funduszu Społecznego**

Projekt ”Studia podyplomowe z zakresu wytwarzania oprogramowania oraz zarządzania projektami w firmach informatycznych” realizowany w ramach Programu Operacyjnego Kapitał Ludzki