
Bazy Danych

Wykład p.t.

Instrukcje DML

INSERT, UPDATE, DELETE.

COPY

Antoni Ligeza

ligeza@agh.edu.pl

<http://galaxy.uci.agh.edu.pl/~ligeza>

Wykorzystano materiały:

[http:](http://www.postgresql.org/docs/8.3/interactive/index.html)

[//www.postgresql.org/docs/8.3/interactive/index.html](http://www.postgresql.org/docs/8.3/interactive/index.html)

DML – Data Manipulation Language

DML obejmuje instrukcje służące do wprowadzania trwałych zmian w bazie danych oraz dostępu do danych; są to instrukcje:

- `INSERT INTO` – Ładowanie nowych danych (rekordów) do tabeli,
- `UPDATE` – zmiana istniejących danych w tabeli,
- `DELETE FROM` – kasowanie rekordów z tabeli,
- `SELECT` – selektywny dostęp do danych.

Niektóre podręczniki do języka DML zaliczają tylko instrukcje powodujące trwałe zmiany stanu bazy (`INSERT`, `UPDATE`, `DELETE`), a instrukcję `SELECT` do Data Query Language (DQL).

Instrukcje `UPDATE` oraz `DELETE` wykorzystują mechanizm selekcji rekordów analogiczny do instrukcji `SELECT`, oparty na warunkach zdefiniowanych w `WHERE`.

Uwaga: Instrukcje `UPDATE` oraz `DELETE` bez podanych kryteriów wybory rekordów działają na całej tabeli. Wykonane zmiany są nieodwracalne.

INSERT INTO

```
INSERT INTO table [ ( column [, ...] ) ]  
    { DEFAULT VALUES | VALUES ( { expression | DEFAULT }  
    [, ...] ) [, ...] | query }  
    [ RETURNING * | output_expression [ AS output_name ]  
    [, ...] ]
```

`table` – nazwa tablicy,

`column` – nazwa kolumny,

`VALUES` – deklaruje listę wartości dla wstawianego rekordu,

`DEFAULT VALUES` – kolumny będą wypełnione wartościami defaultowymi,

`expression` – dana kolumna będzie wypełniona wartością wyrażenia,

`DEFAULT` – wypełnienie kolumny wartością defaultową,

`query` – sapytanie wyznaczające wstawiane rekordy,

`output_expression` – zwracane wyrażenie (po każdym insercie).

INSERT INTO – pełny rekord

Wprowadzamy pełny rekord nie specyfikując kolumn:

```
pracownicy=> \d pra
```

```
Table "public.pra"
  Column          |          Type          | Modifiers
-----+-----+-----
 id_prac         | character(5)           |
 nazwisko        | character varying(32) |
 imie            | character varying(16) |
 dzial           | character(5)           |
 stanowisko      | character varying(24) |
 pobory          | numeric(8,2)           |
```

```
pracownicy=> select * from pra;
```

```
 id_prac | nazwisko | imie | dzial | stanowisko | pobory
-----+-----+-----+-----+-----+-----
 1100    | Kowal    | Adam | PD303 | robotnik   | 1500.00
 110     | Kowalik  | Artur | PD303 | kierownik  | 1500.00
 1101    | Kowalski | Antoni | PD303 | robotnik   | 4500.00
 111     | Kowalczuk | Adam | PR202 | kierownik  | 2500.00
 1010    | Kawula   | Alojzy | PK101 | robotnik   | 2500.00
```

```
(5 rows)
```

```
pracownicy=> INSERT INTO pra VALUES
  ('100', 'Janik', 'Jan', 'PK101', 'analitik', 5000);
INSERT 0 1
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
100	Janik	Jan	PK101	analitik	5000.00

```
(6 rows)
```

INSERT INTO – niepełny rekord

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
100	Janik	Jan	PK101	analitik	5000.00

(6 rows)

```
pracownicy=> INSERT INTO pra VALUES
```

```
  ('101', 'Janiak', 'Jan', 'PK101', 'analitik');
```

```
INSERT 0 1
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
100	Janik	Jan	PK101	analitik	5000.00
101	Janiak	Jan	PK101	analitik	

(7 rows)

Niepełne rekordy (bez specyfikacji kolumn) nie są dopuszczalne w standardzie SQL.

INSERT INTO – pełna specyfikacja kolumn

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

(5 rows)

```
pracownicy=> INSERT INTO pra
```

```
  (id_prac,nazwisko,imie,dzial,stanowisko,pobory)
```

```
  VALUES ('100','Janik','Jan','PK101','analitik',5000);
```

```
INSERT 0 1
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
100	Janik	Jan	PK101	analitik	5000.00

(6 rows)

INSERT INTO – konsekwentna zmiana kolejności kolumn

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczuk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

(5 rows)

```
pracownicy=> INSERT INTO pra
```

```
  (nazwisko,imie,dzial,stanowisko,pobory,id_prac)
```

```
  VALUES ('Janik','Jan','PK101','analitik',5000,'100');
```

```
INSERT 0 1
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczuk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
100	Janik	Jan	PK101	analitik	5000.00

(6 rows)

Przy zmianie kolejności kolumn należy również zmienić kolejność danych.

INSERT INTO – niepełna specyfikacja kolumn

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

(5 rows)

```
pracownicy=> INSERT INTO pra  
  (id_prac,nazwisko,imie,dzial)  
  VALUES ('100','Janik','Jan','PK101');  
INSERT 0 1
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
100	Janik	Jan	PK101		

(6 rows)

Kolumny niewyspecyfikowane będą wypełniane wartościami defaultowymi.

INSERT INTO – jawna specyfikacja DEFAULT i NULL

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

(5 rows)

```
pracownicy=> INSERT INTO pra
```

```
  (id_prac,nazwisko,imie,dzial,stanowisko,pobory)
```

```
  VALUES ('100','Janik','Jan',DEFAULT,NULL,2000);
```

```
INSERT 0 1
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
100	Janik	Jan			2000.00

(6 rows)

INSERT INTO – wprowadzanie wielu rekordów

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczuk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

(5 rows)

```
pracownicy=> INSERT INTO pra
```

```
(id_prac,nazwisko,imie,dzial,stanowisko,pobory)
```

```
VALUES ('100','Janik','Jan','PK101','analitik',5000),
```

```
('101','Janiak','Jerzy','PR202','projektant',6000);
```

```
INSERT 0 2
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczuk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
100	Janik	Jan	PK101	analitik	5000.00
101	Janiak	Jerzy	PR202	projektant	6000.00

(7 rows)

INSERT INTO z SELECT

Możliwe jest załadowanie do tablicy rekordów utworzonych instrukcją SELECT.

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

(5 rows)

```
pracownicy=> INSERT INTO pra
```

```
(id_prac,nazwisko,imie,dzial,stanowisko,pobory) SELECT
id_prac,nazwisko,imie,dzial,stanowisko,pobory FROM prac
WHERE nazwisko LIKE 'J%';
```

```
INSERT 0 2
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczyk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
100	Janik	Jan	PK101	analitik	5000.00
102	Janiak	Jerzy	PK101	analitik	6000.00

(7 rows)

INSERT INTO z SELECT: forma uproszczona

```
pracownicy=> select * from pra;
```

```
 id_prac | nazwisko | imie | dzial | stanowisko | pobory  
-----+-----+-----+-----+-----+-----  
(0 rows)
```

```
pracownicy=> INSERT INTO pra
```

```
    SELECT id_prac,nazwisko,imie,dzial,stanowisko,pobory  
          FROM prac;
```

```
INSERT 0 9
```

```
pracownicy=> select * from pra;
```

```
 id_prac | nazwisko | imie | dzial | stanowisko | pobory  
-----+-----+-----+-----+-----+-----  
 1100   | Kowal    | Adam | PD303 | robotnik   | 1500.0  
 110    | Kowalik  | Artur | PD303 | kierownik  | 1500.0  
 1110   | Kowalewski | Adam | PR202 | robotnik   | 3500.0  
 101    | Kowalczyk | Amadeusz | PK101 | kierownik  | 1000.0  
 1101   | Kowalski | Antoni | PD303 | robotnik   | 4500.0  
 1011   | Kowalowski | Alojzy | PK101 | robotnik   | 2500.0  
 111    | Kowalczuk | Adam | PR202 | kierownik  | 2500.0  
 1010   | Kawula   | Alojzy | PK101 | robotnik   | 2500.0  
 102    | Janiak   | Jerzy | PK101 | analityk   | 6000.0  
(9 rows)
```

INSERT INTO – podsumowanie

INSERT INTO to podstawowa instrukcja do wprowadzania danych do tabel.

- wprowadzane rekordy specyfikowane są po słowie VALUES,
- możliwe jest wprowadzanie 1 lub wielu rekordów,
- przy wprowadzaniu danych bez jawnej specyfikacji kolumn należy podać wszystkie wartości pól w odpowiedniej liczbie i kolejności (PostgreSQL dopuszcza krótszą listę wartości),
- bezpieczna forma INSERT posiada jawną specyfikację wartości,
- jawna specyfikacja wartości pozwala na:
 - zmianę kolejności pól,
 - pominięcie pewnych kolumn,
- w specyfikacji pól możliwe jest użycie słów DEFAULT oraz NULL,
- wprowadzanie wartości NULL jest możliwe poprzez:
 - jawne podanie wartości NULL,
 - pominięcie specyfikacji kolumny i nazwy,
- pusty łańcuch (""), ' NULL ' oraz NULL to różne wartości,
- znaki specjalne, np. ' poprzedzamy \.
- nie należy wprowadzać wartości do kolumn typu *serial* – są one wprowadzane automatycznie.

SELECT INTO

`SELECT INTO` tworzy nową tabelę korzystając z kolumn i danych istniejącej tabeli.

```
pracownicy=> \dt
```

```
      List of relations
```

Schema	Name	Type	Owner
public	dzial	table	ali
public	pra	table	ali
public	prac	table	ali

(3 rows)

```
pracownicy=> SELECT * INTO new_pra FROM pra;
SELECT
```

```
pracownicy=> \dt
```

```
      List of relations
```

Schema	Name	Type	Owner
public	dzial	table	ali
public	new_pra	table	ali
public	pra	table	ali
public	prac	table	ali

(4 rows)

```
pracownicy=> select * from new_pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.0
110	Kowalik	Artur	PD303	kierownik	1500.0
1110	Kowalewski	Adam	PR202	robotnik	3500.0
101	Kowalczyk	Amadeusz	PK101	kierownik	1000.0
1101	Kowalski	Antoni	PD303	robotnik	4500.0

1011		Kowalowski		Alojzy		PK101		robotnik		2500.0
111		Kowalczyk		Adam		PR202		kierownik		2500.0
1010		Kawula		Alojzy		PK101		robotnik		2500.0
102		Janiak		Jerzy		PK101		analitik		6000.0

(9 rows)

UPDATE

UPDATE pozwala na modyfikację zawartości istniejących rekordów.

```
UPDATE [ ONLY ] table [ [ AS ] alias ]
    SET { column = { expression | DEFAULT } |
        ( column [, ...] ) =
        ( { expression | DEFAULT } [, ...] ) } [, ...]
    [ FROM fromlist ]
    [ WHERE condition ]
    [ RETURNING * | output_expression
    [ AS output_name ] [, ...] ]
```

table – tabela w której wprowadzane są zmiany,

FROM – wprowadzenie kolumn z innych tabel do WHERE,

WHERE – specyfikacja zmiennych rekordów,

DEFAULT – wartość defaultową.

UPDATE

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
101	Kowalczyk	Amadeusz	PK101	kierownik	1000.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczuk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

(6 rows)

```
pracownicy=> UPDATE pra
```

```
SET pobory = pobory*1.2 WHERE stanowisko='kierownik';
```

```
UPDATE 3
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
110	Kowalik	Artur	PD303	kierownik	1800.00
101	Kowalczyk	Amadeusz	PK101	kierownik	1200.00
111	Kowalczuk	Adam	PR202	kierownik	3000.00

(6 rows)

UPDATE – testowanie

UPDATE wprowadza trwałe zmiany w tabeli. Bezpiecznie jest przetestować jego zakres i efekty.

```
pracownicy=> SELECT * FROM pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00
110	Kowalik	Artur	PD303	kierownik	1800.00
101	Kowalczyk	Amadeusz	PK101	kierownik	1200.00
111	Kowalczuk	Adam	PR202	kierownik	3000.00

(6 rows)

Definiujemy warunek wyboru:

```
pracownicy=> SELECT * FROM pra WHERE dzial='PD303';
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
110	Kowalik	Artur	PD303	kierownik	1800.00

(3 rows)

Przerabiamy SELECT na UPDATE:

```
pracownicy=> UPDATE pra
```

```
    SET pobory=pobory+1234 WHERE dzial='PD303';
```

```
UPDATE 3
```

```
pracownicy=> SELECT * FROM pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1010	Kawula	Alojzy	PK101	robotnik	2500.00

101		Kowalczyk		Amadeusz		PK101		kierownik		1200.00
111		Kowalczuk		Adam		PR202		kierownik		3000.00
1100		Kowal		Adam		PD303		robotnik		2734.00
1101		Kowalski		Antoni		PD303		robotnik		5734.00
110		Kowalik		Artur		PD303		kierownik		3034.00

(6 rows)

DELETE FROM – kasowanie rekordów

```
DELETE FROM [ ONLY ] table [ [ AS ] alias ]  
    [ USING usinglist ]  
    [ WHERE condition | WHERE CURRENT OF cursor_name ]  
    [ RETURNING * | output_expression [ AS output_name ]  
    [ , ... ] ]
```

ONLY – gdy wyspecyfikowane, zmiany odnoszą się tylko do bieżącej tablicy; jeżeli nie, również do tabel dziedziczących.

USING – pozwala na specyfikację tablic z których kolumny można użyć w WHERE,

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1010	Kawula	Alojzy	PK101	robotnik	2500.00
101	Kowalczyk	Amadeusz	PK101	kierownik	1200.00
111	Kowalczuk	Adam	PR202	kierownik	3000.00
1100	Kowal	Adam	PD303	robotnik	2734.00
1101	Kowalski	Antoni	PD303	robotnik	5734.00
110	Kowalik	Artur	PD303	kierownik	3034.00

```
(6 rows)
```

```
pracownicy=> DELETE FROM pra WHERE nazwisko LIKE 'Kowal%';  
DELETE 5
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1010	Kawula	Alojzy	PK101	robotnik	2500.00

```
(1 row)
```

DELETE – stosowanie

DELETE wprowadza trwałe zmiany w tabeli. Bezpiecznie jest przetestować jego zakres.

```
pracownicy=> SELECT * FROM pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
101	Kowalczyk	Amadeusz	PK101	kierownik	1000.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczuk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

(6 rows)

Testujemy kryterium wyboru:

```
pracownicy=> SELECT * FROM pra WHERE id_prac = '111';
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
111	Kowalczuk	Adam	PR202	kierownik	2500.00

(1 row)

Przerabiamy SELECT na DELETE:

```
pracownicy=> DELETE FROM pra WHERE id_prac = '111';
```

```
DELETE 1
```

```
pracownicy=> SELECT * FROM pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
101	Kowalczyk	Amadeusz	PK101	kierownik	1000.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00

```
1010      | Kawula      | Alojzy     | PK101 | robotnik    | 2500.00  
(5 rows)
```


TRUNCATE – efektywne kasowanie zawartości tabeli

W PostgreSQL efektywne kasowanie zawartości tabeli można uzyskać instrukcją TRUNCATE.

```
TRUNCATE [ TABLE ] name [, ...] [ CASCADE | RESTRICT ]
```

TRUNCATE – efektywne dla dużych tabel. Odzyskuje pamięć. Może opróżnić wiele tabel na raz. CASCADE – kaskadowe usuwanie powiązanych rekordów, a więc czyszczenie tabel podrzędnych zawierających klucze obce,

RESTRICT – zapobiega usuwaniu zawartości tabel, jeżeli są do nich referencje z tabel podrzędnych.

```
pracownicy=> SELECT * FROM pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
101	Kowalczyk	Amadeusz	PK101	kierownik	1000.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

(5 rows)

```
pracownicy=> TRUNCATE pra;
```

```
TRUNCATE TABLE
```

```
pracownicy=> SELECT * FROM pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
---------	----------	------	-------	------------	--------

(0 rows)

COPY FROM, COPY TO

Instrukcje COPY w wersji PostgreSQL 8.3.

```
COPY tablename [ ( column [, ...] ) ]
  FROM { 'filename' | STDIN }
  [ [ WITH ]
    [ BINARY ]
    [ OIDS ]
    [ DELIMITER [ AS ] 'delimiter' ]
    [ NULL [ AS ] 'null string' ]
    [ CSV [ HEADER ]
      [ QUOTE [ AS ] 'quote' ]
      [ ESCAPE [ AS ] 'escape' ]
      [ FORCE NOT NULL column [, ...] ]
    ]
  ]

COPY { tablename [ ( column [, ...] ) ] | ( query ) }
  TO { 'filename' | STDOUT }
  [ [ WITH ]
    [ BINARY ]
    [ HEADER ]
    [ OIDS ]
    [ DELIMITER [ AS ] 'delimiter' ]
    [ NULL [ AS ] 'null string' ]
    [ CSV [ HEADER ]
      [ QUOTE [ AS ] 'quote' ]
      [ ESCAPE [ AS ] 'escape' ]
      [ FORCE QUOTE column [, ...] ]
    ]
  ]
```

Instrukcje COPY w wersji PostgreSQL 7.3.

```
COPY [ BINARY ] tablename [ WITH OIDS ]  
FROM { 'filename' | STDIN }  
[ [USING] DELIMITERS 'delimiter' ]  
[ WITH NULL AS 'null string' ]
```

```
COPY [ BINARY ] tablename [ WITH OIDS ]  
TO { 'filename' | STDOUT }  
[ [USING] DELIMITERS 'delimiter' ]  
[ WITH NULL AS 'null string' ]
```

Instrukcja COPY wymaga uprawnień administratora.

Instrukcja psql copy

Instrukcja `\copy` jest poleceniem psql. Nie wymaga uprawnień administratora. Pozwala kopiować do i z plików.

```
\copy { table [ ( column_list ) ] | ( query ) }
      { from | to } { filename | stdin | stdout | pstdin | pstdout
      [ with ] [ binary ] [ oids ] [ delimiter [ as ] 'character'
      [ null [ as ] 'string' ] [ csv [ header ]
      [ quote [ as ] 'character' ] [ escape [ as ] 'character' ]
      [ force quote column_list ] [ force not null column_list
```

```
pracownicy=> select * from pra;
```

id_prac	nazwisko	imie	dzial	stanowisko	pobory
1100	Kowal	Adam	PD303	robotnik	1500.00
110	Kowalik	Artur	PD303	kierownik	1500.00
101	Kowalczyk	Amadeusz	PK101	kierownik	1000.00
1101	Kowalski	Antoni	PD303	robotnik	4500.00
111	Kowalczuk	Adam	PR202	kierownik	2500.00
1010	Kawula	Alojzy	PK101	robotnik	2500.00

```
(6 rows)
```

```
\copy pra to 'pra.csv' with csv header
```

```
id_prac,nazwisko,imie,dzial,stanowisko,pobory
1100 ,Kowal,Adam,PD303,robotnik,1500.00
110  ,Kowalik,Artur,PD303,kierownik,1500.00
101  ,Kowalczyk,Amadeusz,PK101,kierownik,1000.00
1101 ,Kowalski,Antoni,PD303,robotnik,4500.00
111  ,Kowalczuk,Adam,PR202,kierownik,2500.00
1010 ,Kawula,Alojzy,PK101,robotnik,2500.00
```

Plik tekstowy w formacie CSV (bez nagłówka):

```
1100 ,Kowal,Adam,PD303,robotnik,1500.00
110 ,Kowalik,Artur,PD303,kierownik,1500.00
101 ,Kowalczyk,Amadeusz,PK101,kierownik,1000.00
1101 ,Kowalski,Antoni,PD303,robotnik,4500.00
111 ,Kowalczuk,Adam,PR202,kierownik,2500.00
1010 ,Kawula,Alojzy,PK101,robotnik,2500.00
```

```
pracownicy=> truncate pra;
```

```
TRUNCATE TABLE
```

```
pracownicy=> select * from pra;
```

```
 id_prac | nazwisko | imie | dzial | stanowisko | pobory
-----+-----+-----+-----+-----+-----
(0 rows)
```

```
pracownicy=> \copy pra from 'pra-no-header.csv' using delimiter
```

```
pracownicy=> select * from pra;
```

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

Kopiowanie – uwagi

- format CSV – wartości (pola) oddzielone przecinkami; możliwe inne separatory,
- standardowym separatorem jest znak tabulacji,
- możliwy znak separatora – potok |,
- opcja `WITH NULL AS` pozwala zdefiniować ciąg znaków, który będzie interpretowany jako `NULL`,
- domyślnie jako `NULL` jest interpretowany ciąg `\N`,
- ważne: czystość danych,
- ważne: pola daty,
- ważne: kolejność i kompletność kolumn.