Polecenia DDL w phpMyAdmin.

Tworzenie tabel i relacji w phpMyAdmin.

DML w phpMyAdmin.

Polecenia DQL w phpMyAdmin.

# Przed przystąpieniem do tematów należy zapoznać się z popularnymi typami danych MySQL[[1]](#endnote-1)

|  |  |  |
| --- | --- | --- |
| Typ | Rozmiar | Opis |
| CHAR[Length] | Length bajtów | Pole o stałej długości, przechowuje od 0 do 255 znaków |
| VARCHAR[Length] | Długość łańcucha + 1 bajt | Pole tekstowe o zmiennej długości |
| TINYTEXT | Długość łańcucha + 1 bajt | Łańcuch o maksymalnej długości 255 znaków |
| TEXT | Długość łańcucha + 2 bajty | Łańcuch o maksymalnej długości 65535 znaków |
| MEDIUMTEXT | Długość łańcucha + 3 bajty | Łańcuch o maksymalnej długości 16777215 znaków |
| LONGTEXT | Długość łańcucha + 4 bajty | Łańcuch o maksymalnej długości 4294967295 znaków |
| TINYINT[Length] | 1 bajt | Liczby z zakresu od -128 do 127 lub liczby dodatnie od 0 do 255 |
| SMALLINT[Length] | 2 bajty | Liczby z zakresu od -32768 do 32767 lub liczby dodatnie od 0 do 65535 |
| MEDIUMINT[Length] | 3 bajty | Liczby z zakresu od -8388608 do 8388607 lub liczby dodatnie od 0 do 16777215 |
| INT[Length] | 4 bajty | Liczby z zakresu od -2147483648 do 2147483647 lub liczby dodatnie od 0 do 4294967295 |
| BIGINT[Length] | 8 bajtów | Liczby z zakresu od -9223372036854775808 do 9223372036854775807 lub liczby dodatnie od 0 do 18446744073709551615 |
| FLOAT | 4 bajty | Mała liczba rzeczywista, zmiennoprzecinkowa |
| DOUBLE[Length, Decimals] | 8 bajtów | Duża liczba rzeczywista, zmiennoprzecinkowa |
| DECIMAL[Length, Decimals] | Length + 1 lub Length + 2 bajtów | Liczba typu DOUBLE przechowywana w postaci łańcucha co pozwala na zastosowanie stałej liczby miejsc po przecinku |
| DATE | 3 bajty | Data w formacie YYYY-MM-DD |
| DATETIME | 8 bajtów | Data w formacie YYYY-MM-DD HH:MM:SS |
| TIMESTAMP | 4 bajty | Data w formacie YYYYMMDDHHMMSS; dopuszczalny zakres kończy się na rok 2037 |
| TIME | 3 bajty | Data w formacie HH:MM:SS |
| ENUM | 1 lub 2 bajty | Enumeracja (wyliczenie). W kolumnie może się znaleźć jedna z podanych wartości |
| SET | 1, 2, 3, 4 lub 8 bajtów | Tak samo jak ENUM z tym że w kolumnie może się znaleźć kilka wartości jednocześnie |

## Atrybuty w MySQL

### UNSIGNED

Informujemy bazę, że będziemy przechowywać tylko wartości dodatnie. Analogicznie SIGNED informuje, że będziemy przechowywać również ujemne.

### AUTO INCREMENT

Służy do generowania unikalnej wartości liczbowej dla nowych wierszy.

### NULL / NOT NULL

Informujemy bazę czy pole musi być uzupełnione czy dopuszczamy by nie zawierało wartości

# **Polecenia DDL w phpMyAdmin**

Relacyjne bazy danych składają się z tabel. Dodatkowo w bazach występują inne obiekty jak indeksy (ang. index), klucze obce (ang. foreign key), klucze główne (ang. primary key), ograniczenia (ang. constraint), wyzwalacze (ang. trigger) czy widoki (ang. view). Część języka odpowiedzialna za zarządzanie tymi obiektami to DDL. Zapytania należące do DDL to:

* CREATE – tworzą obiekty bazy danych,
* ALTER – modyfikują tabele bazy danych,
* DROP – usuwają obiekty bazy danych,
* TRUNCATE – usuwa wszystkie dane z tabeli.

# Tworzenie i usuwanie bazy

Najpierw zaprojektujemy prostą bazę danych ***sklepik\_abc***, aby potem ją utworzyć, wypełnić danymi i wykonywać operacje na danych.

## Identyfikatory MySQL

Podsumujmy zasady nazewnictwa, które będziemy stosować podczas pracy z bazą MySQL.

**Nazwa obiektu bazy danych to jego identyfikator**. W MySQL mamy takie identyfikatory jak **bazy danych, tabele, kolumny i indeksy oraz aliasy.** Tworząc nazwy będziemy stosować następujące zasady:

* Używamy liter a-z oraz A-Z, bez "polskich ogonków", a także cyfr 0-9. Co prawda od wersji 3.23.6 serwera MySQL, można stosować standard Unicode zawierający znaki różnych języków, ale będziemy w tym zakresie zgodni ze standardami obowiązującymi w językach programowania.
* Pierwszym znakiem nie może być cyfra.
* Zamiast spacji będziemy uzywać podkreślnika np. nazwa\_tabeli
* Należy stosować nazwy czytelne, oddające rzeczywistość np. produkty, a nie np. tabela1.

Polecenia SQL będziemy pisać wielkimi literami.

## Schemat bazy *sklepik\_abc*

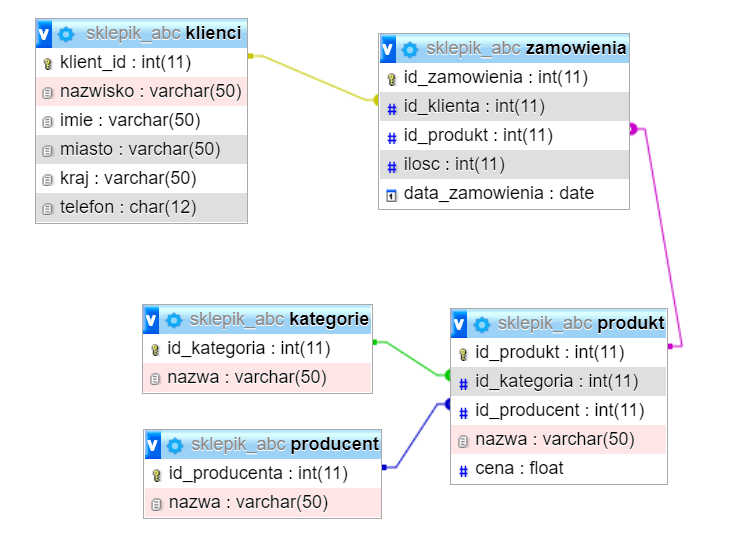
Nasza baza będzie rejestrować sprzedaż książek w księgarni internetowej. W tym celu będą utworzone następujące tabele:

* klienci(id\_klient, nazwisko\_imie, miasto,kraj,telefon)
* zamowienia(id\_zamowienia, id\_klient, id\_produkt, ilosc, data\_zamowienia)
* produkt(id\_produkt,id\_kategoria,id\_producent, nazwa,cena)
* kategorie(id\_kategoria,nazwa)
* producent(id\_producenta,nazwa)

## Diagram tabel i relacji bazy *sklepik\_abc*

Poniższy diagram pokazuje układ tabel i relacji. Na tej podstawie możemy przeanalizować przepływ danych między poszczególnymi tabelami bazy.

Tabela 1. Schemat bazy sklepik\_abc



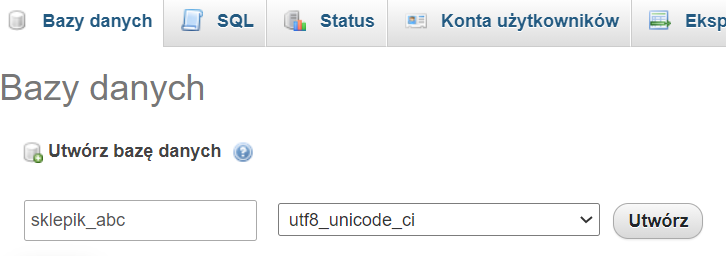
## Utworzenie bazy danych sposoby:

1. Wpisać

CREATE DATABASE ksiegarnia\_internetowa; i kliknąć *Wykonaj*, przy czym zastosujemy bardziej rozbudowane polecenie

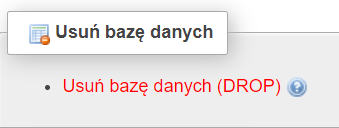
CREATE DATABASE sklepik\_abc DEFAULT CHARACTER SET utf8 COLLATE utf8\_unicode\_ci;, które zagwarantuje nam poprawne wyświetlanie polskich liter.

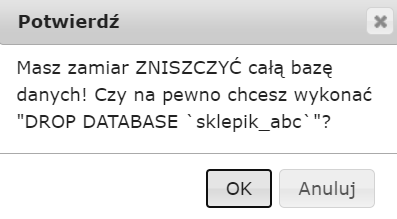
1. Lub



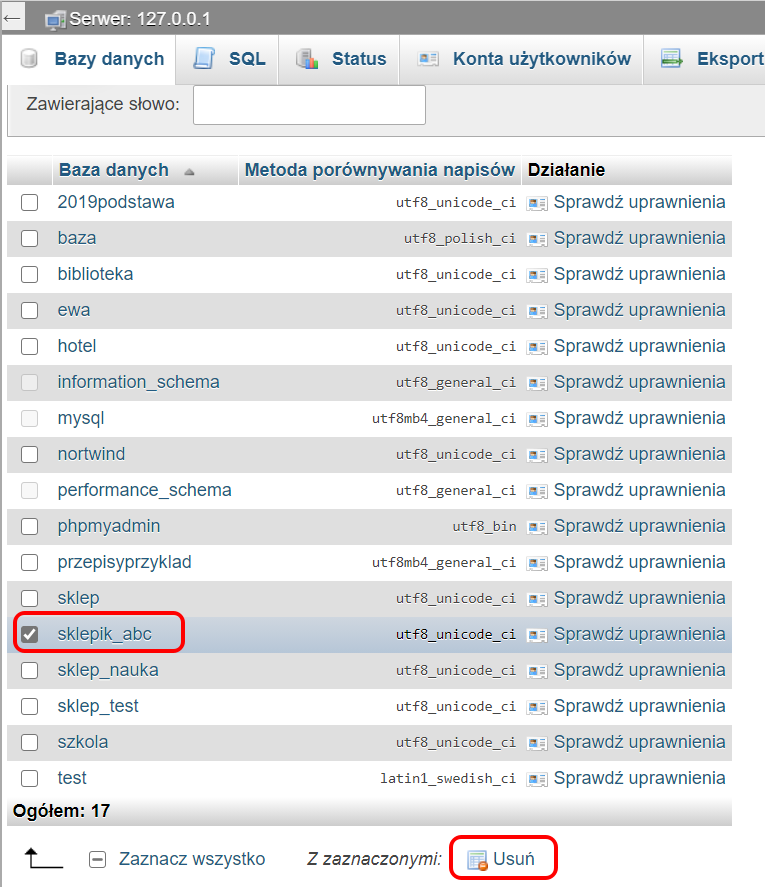
## Usunięcie bazy danych *sklepik\_abc*

Teraz możemy spokojnie przećwiczyć usuwanie bazy danych, ponieważ nic ona jeszcze nie zawiera. **Po usunięciu bazy, należy ją ponownie utworzyć**, aby na następnych zajęciach z niej korzystać.

1. Polecenie DROP DATABASE *sklepik\_abc*; powinno bazę usunąć.
2. Jeżeli pojawi się komunikat, że jest to niemożliwe, wtedy klikamy nazwę naszej bazy na liście baz (po lewej).
3. Otwieramy zakładkę *Operacje*, a następnie *Usuń bazę danych*. 



Lub: wchodzimy do bazy danych🡪zaznaczamy naszą bazę i usuń



## Tworzenie tabel bazy *sklepik\_abc*

Uruchom *phpMyAdmina*, otwórz bazę *sklepik\_abc*, napisz poniższy kod SQL tworzący tabele i wciśnij *Wykonaj*.

CREATE TABLE klienci

(

klient\_id INT AUTO\_INCREMENT PRIMARY KEY,

nazwisko VARCHAR(50) ,

imie VARCHAR(50),

miasto VARCHAR(50),

kraj VARCHAR(50),

telefon CHAR(12)

);

CREATE TABLE kategorie

(

id\_kategoria INT AUTO\_INCREMENT PRIMARY KEY,

nazwa VARCHAR(50)

);

CREATE TABLE producent

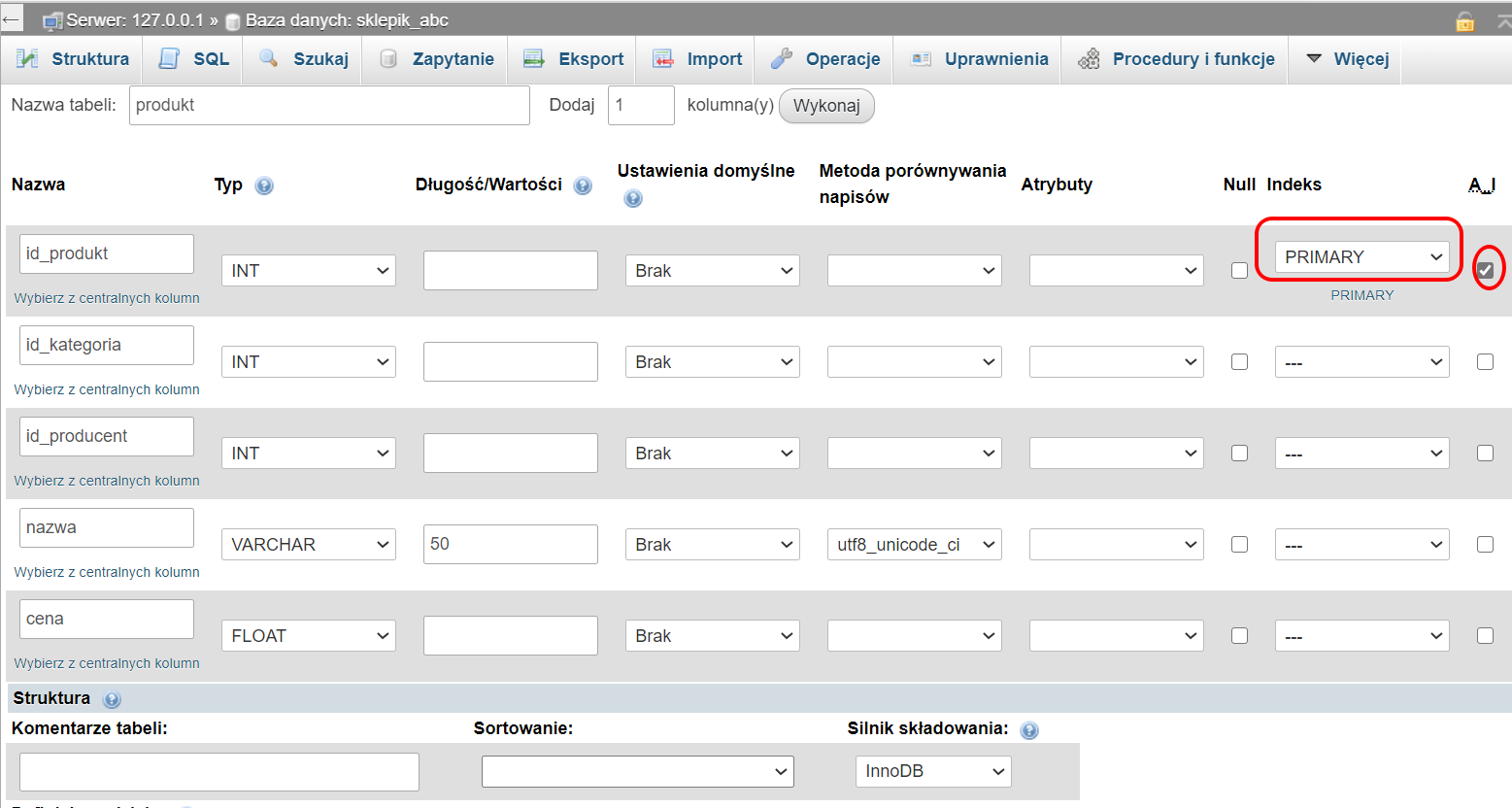
(

id\_producenta INT AUTO\_INCREMENT PRIMARY KEY,

nazwa VARCHAR(50)

);

Graficzny sposób tworzenia tabeli produkt:



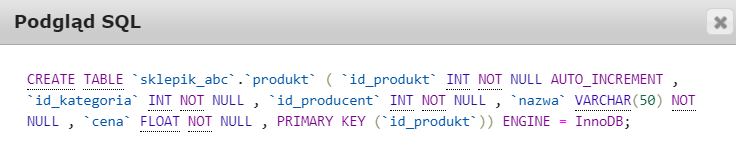
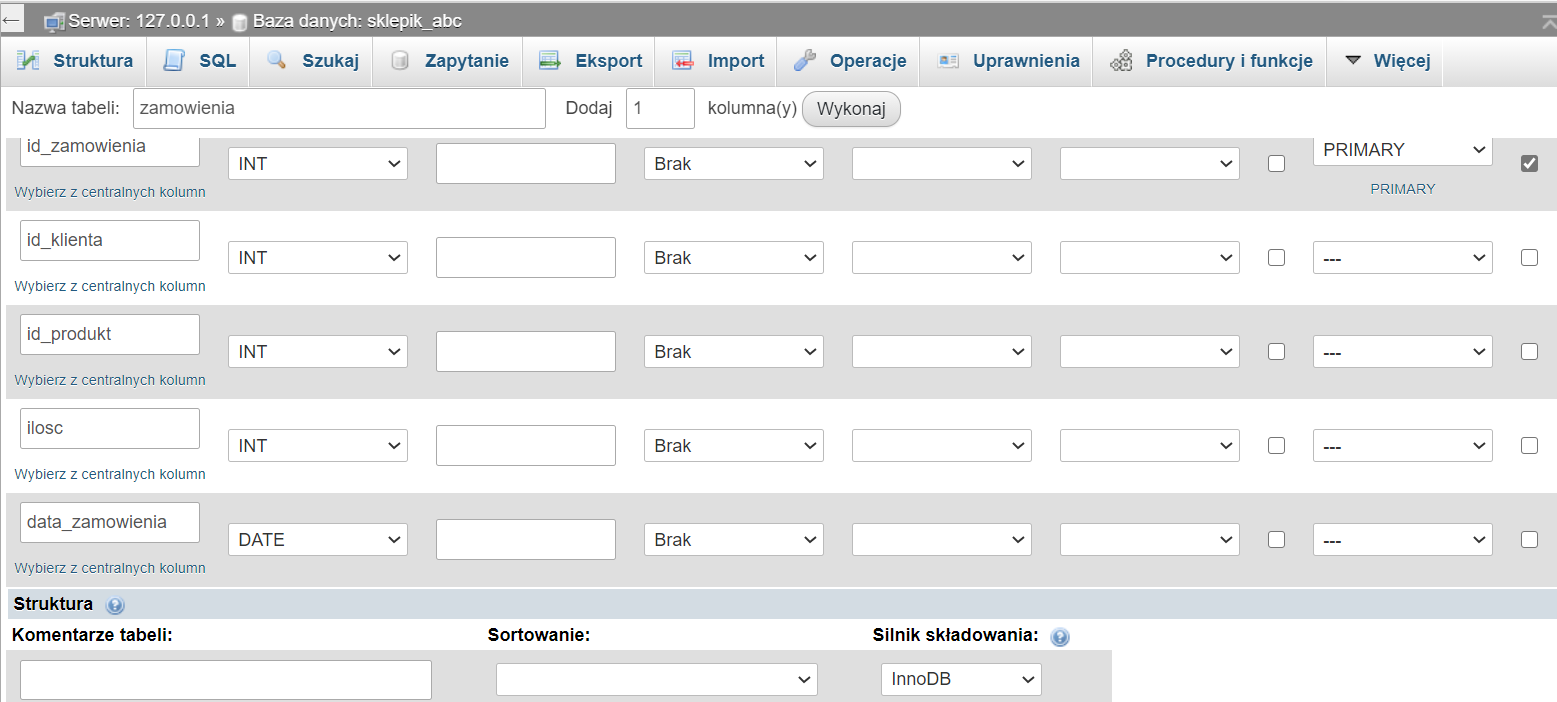


Tabela 2. tworzenie tabeli zamowienia

****

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-table.html) `sklepik\_abc`.`zamowienia` ( `id\_zamowienia` [INT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/numeric-types.html) [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL AUTO\_INCREMENT , `id\_klienta` [INT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/numeric-types.html) [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL , `id\_produkt` [INT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/numeric-types.html) [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL , `ilosc` [INT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/numeric-types.html) [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL , `data\_zamowienia` [DATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/date-and-time-types.html) [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL , PRIMARY KEY (`id\_zamowienia`)) ENGINE = InnoDB;

# Tworzenie tabel i relacji w phpMyAdmin.

## Tworzenie Kluczy obcych (FOREIGN KEY) w MySQL

**FOREIGN KEY** znany jako klucz obcy to pewnego rodzaju odnośnik łączący tabelę w którym występuje klucz obcy z inną tabelą. Klucz obcy zapobiega wszelkim operacjom, które mogłyby zerwać taką więź między tabelami. Również klucz obcy nie może być wartością dowolną. Musi wskazywać na konkretny wiersz w drugiej tabeli. Klucz obcy można ustalić w trakcie tworzenia nowej tabeli lub dla już istniejącej tabeli.

Tabela 3. Składnia na utworzenie tabeli z kluczem obcym FOREIGN KEY

|  |  |
| --- | --- |
| MySQL: | SQL Server / Oracle / MS Access: |
| CREATE TABLE nazwa\_tabeli1 ( nazwa\_kolumny1 typ\_danych, nazwa\_kolumny2 typ\_danych, PRIMARY KEY (nazwa\_kolumny1), FOREIGN KEY (nazwa\_kolumny2) REFERENCES nazwa\_tabeli2(nazwa\_kolumny3)  ); | CREATE **TABLE nazwa\_tabeli1** ( nazwa\_kolumny1 typ\_danych PRIMARY KEY, nazwa\_kolumny2 typ\_danych FOREIGN KEY REFERENCES **nazwa\_tabeli2**(nazwa\_kolumny3) );  Przykład na zwany klucz obcy:  CREATE TABLE Pracownicy (       ID\_pracownik int NOT NULL      ,Nazwisko varchar(40) NOT NULL      ,Imie varchar(20) NOT NULL      ,ID\_miasto int NOT NULL      ,PRIMARY KEY (ID\_pracownik)     , CONSTRAINT fk\_PracownikMiasto FOREIGN KEY (ID\_miasto)      REFERENCES Miasta(ID\_miasto) ) |

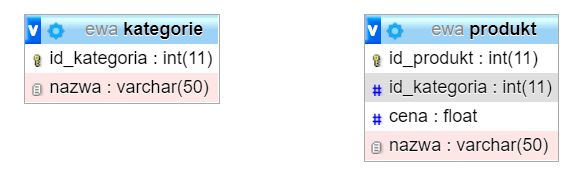
#### Składnia na usunięcie klucza obcego z istniejącej już tabeli

|  |  |
| --- | --- |
| MySQL: | SQL Server / Oracle / MS Access: |
| ALTER TABLE nazwa\_tabeli DROP FOREIGN KEY nazwa\_klucza\_obcego | ALTER TABLE nazwa\_tabeli DROP CONSTRAINT nazwa\_klucza\_obcego  Np.  ALTER TABLE Pracownicy DROP FOREIGN KEY fk\_PracownikMiasto |

#### **Składnia na ustalenie klucza obcego dla istniejącej już tabeli**

ALTER TABLE nazwa\_tabeli  
ADD FOREIGN KEY (nazwa\_kolumny)  
REFERENCES nazwa\_tabeli2(nazwa\_kolumny3);

Chcemy utworzyć relacje miedzy tabelą produkt a kategorie na poniższym przykładzie:



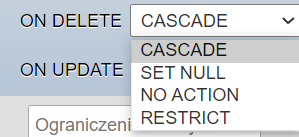
Klucze obce, czyli tzw. FOREIGN KEYS, to nic innego jak powiązania pomiędzy tabelą główną a podrzędną. Klucz obcy w tabeli podrzędnej jest kluczem głównym tabeli nadrzędnej. Są one najczęściej numeryczne ale nic nie stoi na przeszkodzie by klucz główny tabeli był tekstowy.

Tabele mogą używać jednej z dwóch rodzajów relacji:

* [http://iwona.giat.pl/wp-content/uploads/2019/02/narzedzie-relacja-1-n.png](http://iwona.giat.pl/wp-content/uploads/2019/02/narzedzie-relacja-1-n.png)jeden do wielu (1:n) np. relacja pomiędzy produktem i kategorie
* [http://iwona.giat.pl/wp-content/uploads/2019/02/narzedzie-relacja-n-m.png](http://iwona.giat.pl/wp-content/uploads/2019/02/narzedzie-relacja-n-m.png) wiele do wielu (n:m) np. w sklepie wiele różnych produktów może być zakupionych przez wielu różnych klientów

**on update” i „on delete”**

Służą one do wyzwolenia akcji po zaktualizowaniu bądź usunięciu rekordu. Najczęściej używaną opcją jest cascade, jednak przyjrzyjmy się wszystkim:



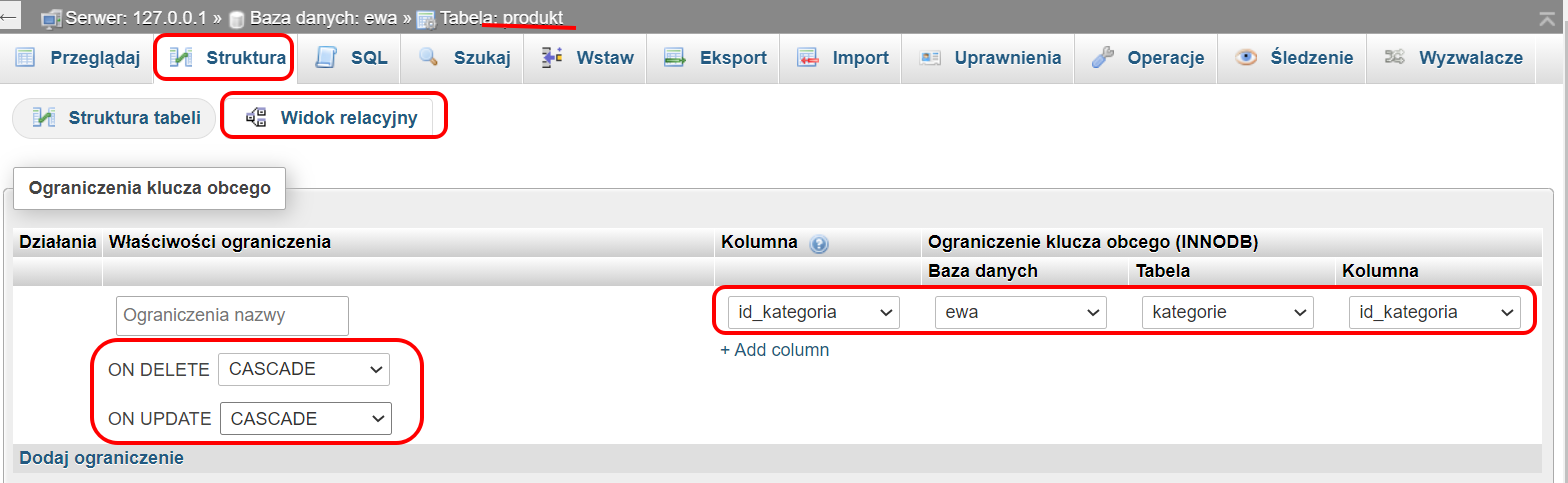
ON UPDATE:

* restrict oznacza, iż nie będzie można zaktualizować wiersza z tabeli kategorie, jeśli posiada powiązanie w tabeli produkt
* cascade oznacza, że jeżeli z jakiś względów będziemy potrzebowali zmienić np. długość klucza głównego w tabeli kategorie (np. poprzedzić ID zerami) , to automatycznie zmiana zostanie naniesiona na kolumnę id\_kategoria w tabeli produkt
* set null oznacza, że jeżeli zmieni się wartość id\_kategoria w tabeli kategorie, wiersze w tabeli produkt, które utracą powiązanie (brak dopasowania id\_kategoria), zmienią wartość id\_kategoria na NULL.
* no action oznacza, że nie zostanie wykonana żadna akcja

ON DELETE:

* restrict oznacza, iż nie będzie można usunąć wiersza z tabeli kategorie, jeśli posiada powiązanie w tabeli produkt
* cascade oznacza, że wraz z wierszem w tabeli kategorie usunięte zostaną wszystkie wiersze z tabeli produkt, powiązane z tym rekordem
* set null oznacza, że zostanie wprowadzona wartość null w tabeli produkt w polu id\_kategoria, jeśli powiązany wiersz w tabeli kategorie zostanie usunięty
* no action oznacza, że nie zostanie wykonana żadna akcja

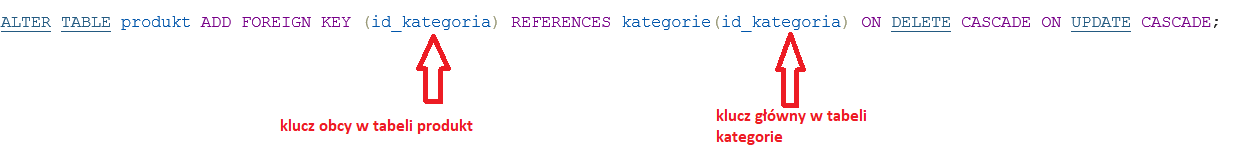
Tworząc relacje, zawsze wchodzimy w strukturę tej tabeli, która posiada klucze obce, inaczej robimy modyfikację tabeli przez polecenie ALTER TABLE .

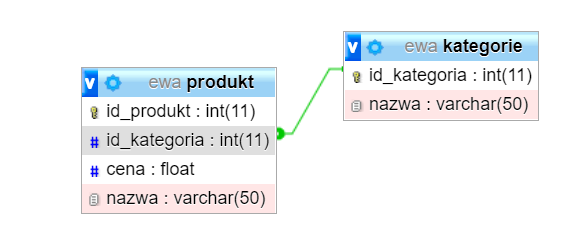


Rysunek 1. Tworzenie klucza obcego w tabeli produkt po kolumnie id\_kategoria

Polecenie na tworzenie powyższego przykładu:

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) produkt ADD FOREIGN KEY (id\_kategoria) REFERENCES kategorie(id\_kategoria) ON [DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) CASCADE ON [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) CASCADE;

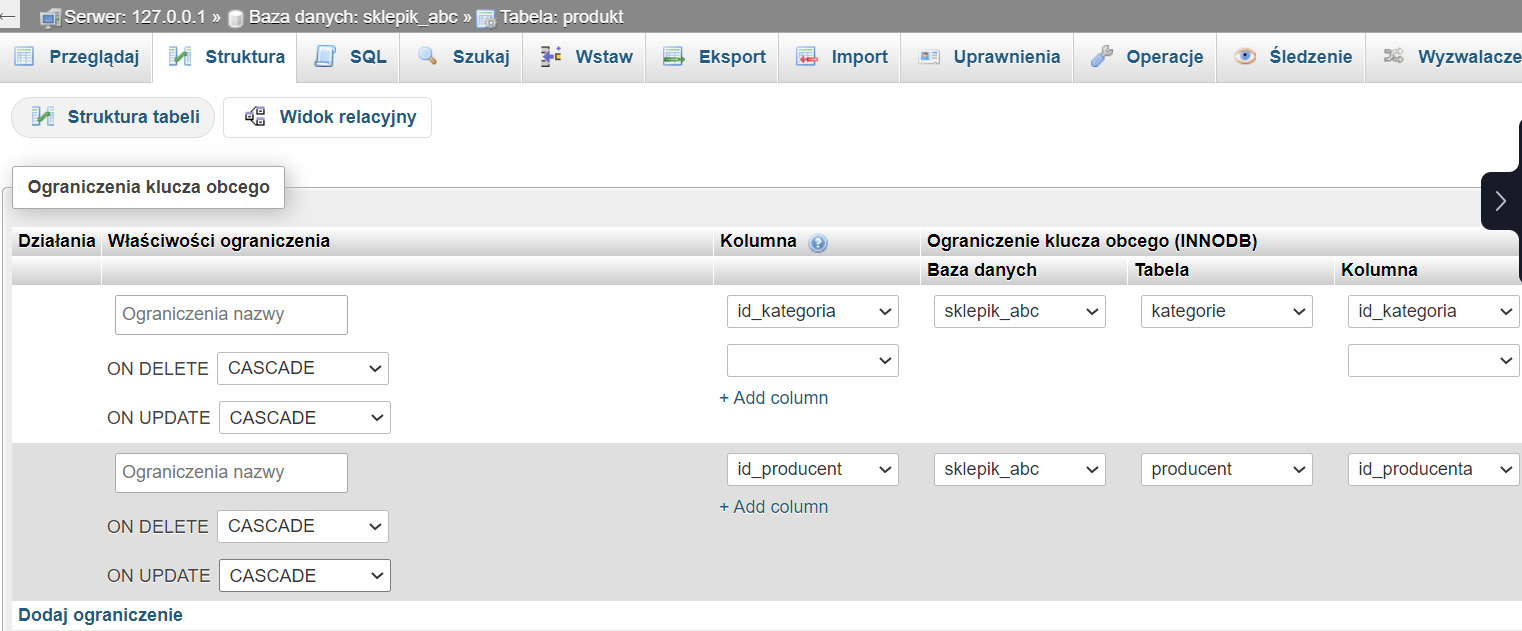




Rysunek 2. Widok projektu bazy- pokazana relacja

Tworzenie kluczy obcych na bazie sklepik\_abc- tworzenie relacji:

Modyfikacja tabeli produkt:



[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) `produkt` ADD FOREIGN KEY (`id\_kategoria`) REFERENCES `kategorie`(`id\_kategoria`) ON [DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) CASCADE ON [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) CASCADE;

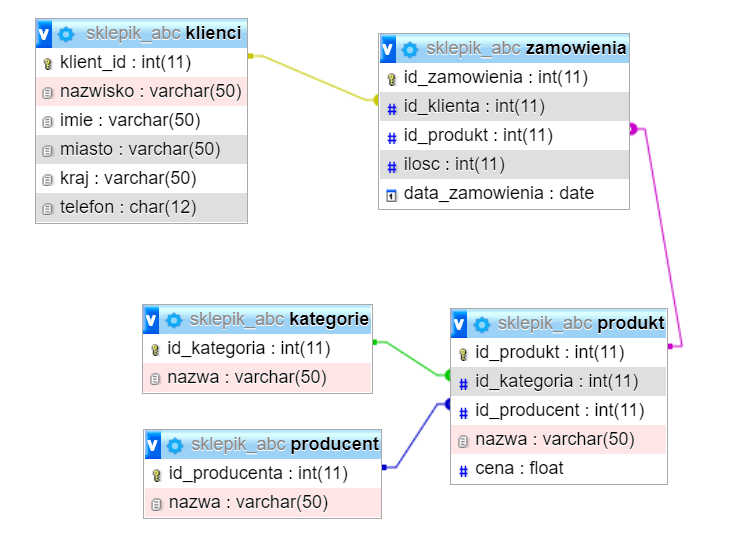
[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) `produkt` ADD FOREIGN KEY (`id\_producent`) REFERENCES `producent`(`id\_producenta`) ON [DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) CASCADE ON [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) CASCADE;

Tworzenie kluczy obcych w tabeli zamówienia:

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) `zamowienia` ADD FOREIGN KEY (`id\_klienta`) REFERENCES `klienci`(`klient\_id`) ON [DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) CASCADE ON [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) CASCADE;

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) `zamowienia` ADD FOREIGN KEY (`id\_produkt`) REFERENCES `produkt`(`id\_produkt`) ON [DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) CASCADE ON [UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) CASCADE;

Utworzone relacje można zobaczyć na schemacie w widoku projektu



**Przykłady poleceń DDL:**

* Utworzenie tabeli

CREATE TABLE klienci\_do\_usuniecia

(

klient\_id INT AUTO\_INCREMENT PRIMARY KEY,

nazwisko VARCHAR(50) ,

imie VARCHAR(50),

miasto VARCHAR(50),

kraj VARCHAR(50),

telefon CHAR(12)

);

* Wstawianie rekordów do tabeli klienci\_do\_usuniecia: - to nie jest polecenie DDL

INSERT INTO `klienci\_do\_usuniecia` (`klient\_id`, `nazwisko`, `imie`, `miasto`, `kraj`, `telefon`) VALUES (NULL, 'Kowalski', 'Jan', 'Rzeszów', 'Polska', '123654788'), (NULL, 'Czechosłowski', 'Piotr', 'Kraków', 'Polska', '321456999'), (NULL, 'Janicki', 'Adam', 'Berlin', 'Niemcy', '336654547'), (NULL, 'Adamski', 'Tomasz', 'Poznań', 'Polska', '363636363');

* Utworzenie widoku- z\_polski:

### Zapytanie CREATE VIEW

VIEW - widok jest to dostęp do wybranej części tabeli wcześniej zdefiniowanej za pomocą zapytania CREATE VIEW. Składnia zapytania CREATE VIEW wygląda następująco:

CREATE VIEW nazwa\_widoku (kolumny) AS SELECT kolumny FROM tabela;

[CREATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-view.html) [VIEW](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/create-view.html) z\_polski AS [SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM `klienci\_do\_usuniecia` WHERE `kraj`='Polska';

* Usunięcie widoku:

DROP VIEW z\_polski;

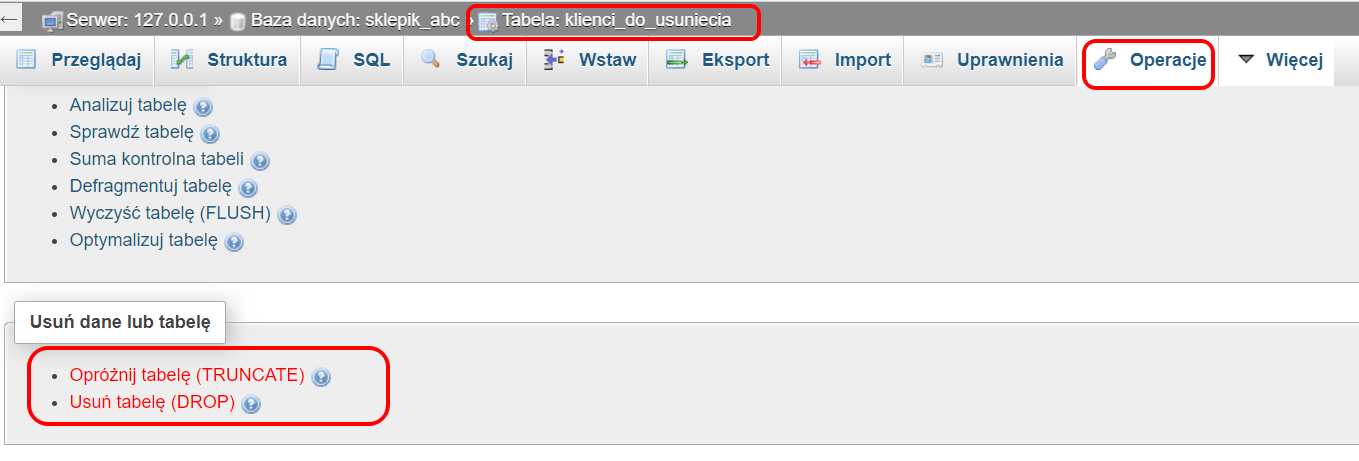
* Usunięcie zawartości tabeli

TRUNCATE TABLE klienci\_do\_usuniecia;

* Usunięcie tabeli

DROP TABLE klienci\_do\_usuniecia ;

Można również wykonać to w formie graficznej:



* Modyfikacja tabeli producenci przez dodanie kolumny kraj\_producenta z domyślną wartością Polska

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) `producent` ADD `kraj\_producenta` [VARCHAR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-types.html)(50) [NOT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/logical-operators.html#operator_not) NULL [DEFAULT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_default) 'Polska' AFTER `nazwa`;

* Modyfikacja tabeli producenct w istniejącej tabeli nazwa ustawienie wartości NOT NULL

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) producent MODIFY nazwa varchar(50) NOT NULL;

* Modyfikacja tabeli producenct przez usuniecie kolumny kraj\_producenta

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) producent DROP COLUMN kraj\_producenta;

* Modyfikacja tabeli producent przez dodanie klucza głównego id\_producenta

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) producent ADD CONSTRAINT PRIMARY KEY (id\_producenta);

* Modyfikacja tabeli klienci przez zmianę kolumny kraj z domyślną wartością Polska

[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) klienci CHANGE kraj kraj [VARCHAR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-types.html)(50) [DEFAULT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_default) 'Polska'

* Dodanie wartości unique (warość nie może powtarzać się) w tabeli producent na kolumnie nazwa

ALTER TABLE producent ADD UNIQUE (nazwa);

* Zmiana nazwy kolumny

ALTER TABLE producent CHANGE COLUMN nazwa nazwa\_producenta varchar(50);

**Dodawanie ograniczeń CHECK w MySQL jest możliwe od wersji 8.0.16 przykład**

* Dodanie warunku logicznego CHECK

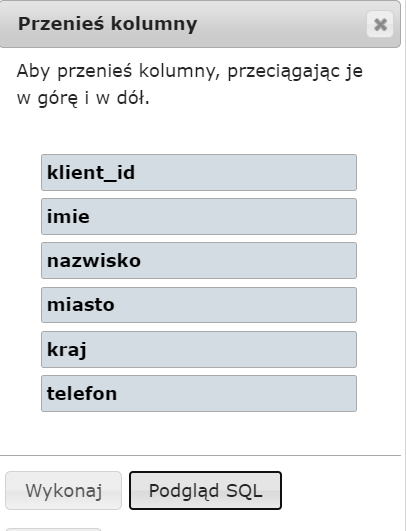
ALTER TABLE produkt ADD CHECK (cena BETWEEN 2 AND 5000); LUB

ALTER TABLE produkt ADD CHECK (cena >= 2);

ALTER TABLE produkt DROP CONSTRAINT CONSTRAINT\_1;

W związku, że posiadam wersję 7.4.8 mysql nie będziemy testować warunków check

Graficzne przenoszenie kolumny – wystarczy przeciągnąć w inne miejsce



[ALTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) [TABLE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/alter-table.html) `klienci` CHANGE `imie` `imie` [VARCHAR](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-types.html)(50) [CHARACTER](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/charset.html) [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/charset.html) utf8 [COLLATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/charset-collations.html) utf8\_unicode\_ci NULL [DEFAULT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_default) NULL AFTER `klient\_id`

DML w phpMyAdmin.

**DML – Data Manipulation Language**. Instrukcje manipulacji danymi. Możemy do nich zaliczyć polecenia takie jak

* INSERT,
* UPDATE,
* DELETE.

Najważniejszą cechą tych instrukcji jest to, że za ich pomocą możemy manipulować danymi w obiektach takich jak tabele.

Zapytanie **INSERT** służy do wprowadzania wierszy do tabeli. Składnia zapytania INSERT wygląda następująco:

INSERT INTO nazwa\_tabeli (kolumna1, kolumna2, ... , kolumnaN) VALUES (wartość1, wartość2, ... , warośćN);

**Przykład 1: wstawianie do tabeli klienci**

[INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO klienci(klient\_id, nazwisko, imie, miasto, kraj, telefon)

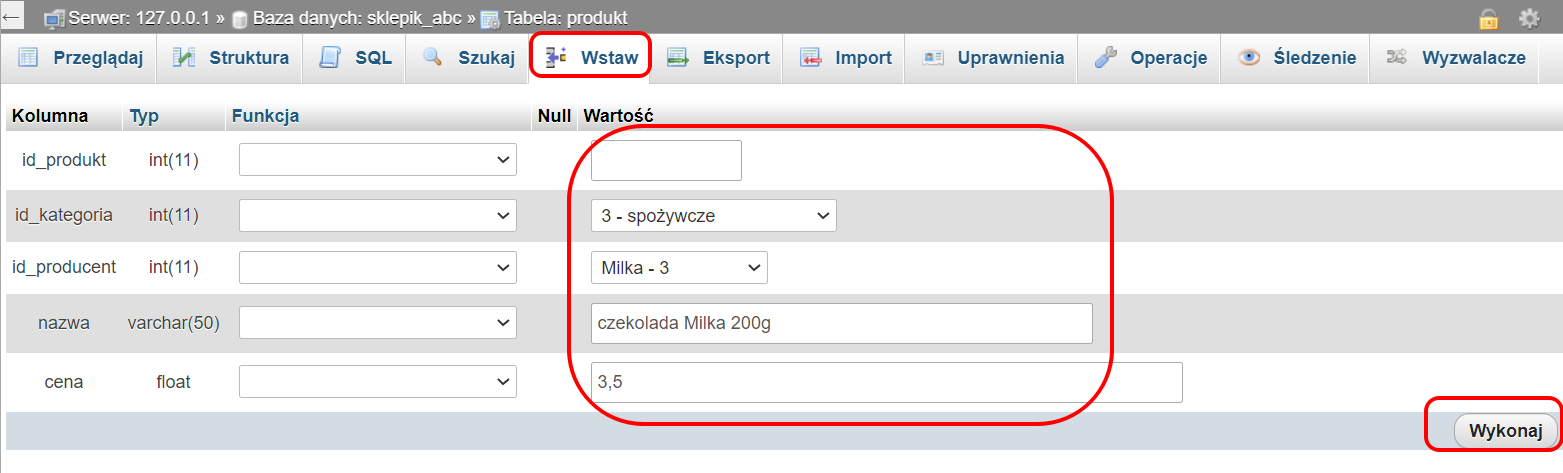
[VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (NULL, 'Anna', 'Mucha', 'Warszawa', 'Polska', '123321321');

W tej wersji musimy pamiętać by wartości umieszczać zgodnie z kolejnością kolumn. Jeżeli chcemy uzupełnić cały wiersz danej tabeli, to nie musimy specyfikować, które kolumny będziemy uzupełniać po nazwie tabeli, ale musimy pamiętać o odpowiedniej kolejności. Kolejność wpisywania kolumn jest dowolna, ale muszą one korespondować z wpisywanymi wartościami. Musimy również pamiętać, że wartości tekstowe wpisujemy w apostrofach 'Mucha'. Jeśli mamy autonumerowanie to przy wartości wpisujemy NULL.

**Za pomocą jednego zapytania INSERT możemy wprowadzić do tabeli więcej niż jeden wiersz.**

[INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO `kategorie` (`id\_kategoria`, `nazwa`) [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (NULL, 'spożywcze'), (NULL, 'chemiczne'), (NULL, 'sprzęt sportowy'), (NULL, 'instrumenty muzyczne'), (NULL, 'RTV');

Wstawianie rekordów w formie graficznej do tabeli produkt.



Cena 3.5 – nie używa się przecinków w MySQL

Dzięki utworzonym związkom po uzupełnieniu tabeli kategorie i producent, można wybrać z listy rozwijalnej możliwe wartości pól.

## Zapytanie UPDATE

Zapytanie **UPDATE** służy do aktualizacji pola lub całego wiersza. Składnia zapytania UPDATE wygląda następująco:  
UPDATE tabela SET kolumna = wartość WHERE warunek;

* Przykład: zmiana ceny czekolady na 2.75

[UPDATE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/update.html) produkt [SET](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/set.html) cena = '2.75' WHERE produkt.id\_produkt = 2;

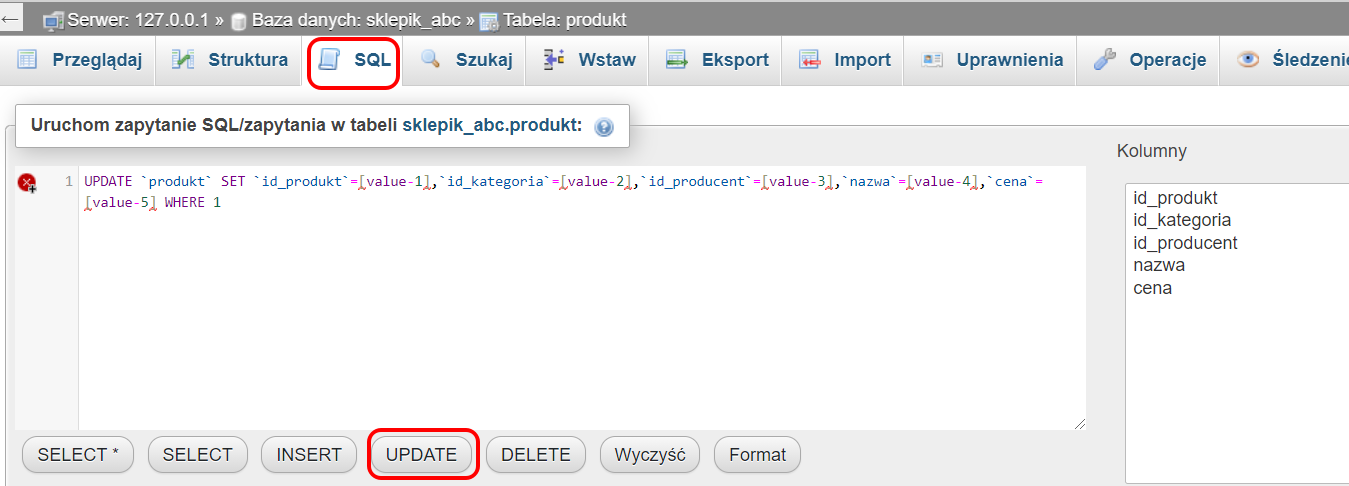
* Przykład: zmiana miasta Krakowa na Niepołomice

UPDATE klienci SET miasto='Niepołomice' WHERE miasto='Kraków';

* Przykład: zwiększenie ceny wszystkich produktów o 25%

UPDATE `produkt` SET cena=cena \*1.25

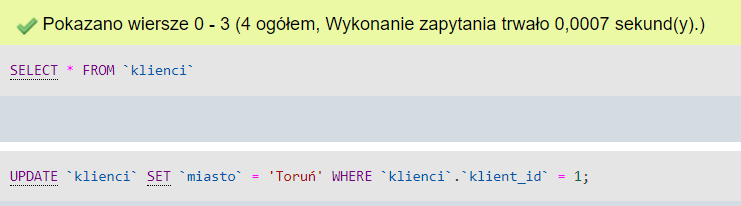
UPDATE można wykonać w formie graficznej dostosowując odpowiednio wygląd struktury:



Lub najłatwiej pojedynczą wartość- wystarczy dwukrotnie kliknąć i wpisać odpowiednią wartość i ENTER.



Wygeneruje się polecenie na UPDATE:



## Zapytanie DELETE

Instrukcja DELETE usuwa dane(wiersze) z tabeli. Składnia instrukcji DELETE:

DELETE FROM tabela WHERE warunek;

**Przykład 1:** Proszę usunąć z tabeli produkt, pozycję o id = 2;

[DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) FROM `produkt` WHERE `id\_produkt`=2;

**Przykład 2:** Proszę usunąć wszystkie wiersze z tabeli produkt

[DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) FROM produkt;

Alternatywnym zapytaniem wykonującym to samo, czyli usunięcie wierszy z tabeli samochody jest instrukcja TRUNCATE. Składnia zapytania TRUNCATE wygląda następująco:  
TRUNCATE TABLE nazwa\_tabeli;

**Przed przystąpieniem wykonania przykładu 3 proszę wstawić do tabeli produkt 4 dowolne produkty, np.**

[INSERT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/insert.html) INTO `produkt` (`id\_produkt`, `id\_kategoria`, `id\_producent`, `nazwa`, `cena`) [VALUES](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/miscellaneous-functions.html#function_values) (NULL, '1', '1', 'odkurzacz', '150'), (NULL, '3', '3', 'Krem orzechowy', '7.50'), (NULL, '6', '5', 'pianino', '6000'), (NULL, '1', '1', 'Mikser', '300');

**Przykład 3:** Proszę usunąć z tabeli produkt najtańszy produkt.

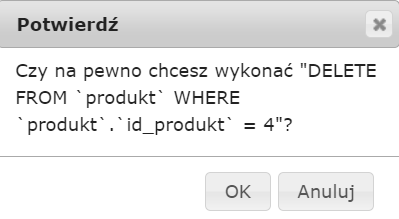
DELETE FROM produkt ORDER BY cena LIMIT 1;

**Przykład 4:** Proszę usunąć z tabeli produkt najdroższy produkt.

[DELETE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/delete.html) FROM produkt ORDER BY cena DESC LIMIT 1;

Oczywiście usunąć rekordy można w sposób graficzny:





**Zadanie.** Proszę usunąć z tabeli producent, jakiegoś producenta, który ma powiązanie z tabelą produkt. U mnie np. Mikla. Proszę odpowiedzieć na pytanie co się stało z rekordem w tabeli produkt po usunięciu producenta Milka?

Polecenia DQL w phpMyAdmin.

Składnia polecenia SELECT  
SELECT kolumna/y  
FROM tabela/e  
WHERE warunek/nki  
GROUP BY kolumna/y  
HAVING   
ORDER BY kolumna/y  
LIMIT

**SELECT** - oznacza wybierz poszczególne kolumny  
**FROM** - z tabeli  
**WHERE** - gdzie warunki  
  
Przy warunkach korzystamy z operatorów =, >, <, >=, <=, <>, LIKE.  
Operator **LIKE** służy do wyszukiwania odpowiednich wzorców w rekordach tabeli zawężając wyświetlane wyniki.

Operatory porównania, najczęściej stosowane w klauzuli WHERE

|  |  |  |  |
| --- | --- | --- | --- |
| Operator | Nazwa, jeżeli jest stosowana | Przykład | Opis |
| = | równy | klient\_id=5 | sprawdza, czy obie wartości są sobie równe |
| > | większy od | wartosc>70.00 | sprawdza, czy pierwsza wartość, jest większa od drugiej |
| < | mniejszy od | wartosc<70.00 | sprawdza, czy pierwsza wartość, jest mniejsza od drugiej |
| >= | większy lub równy | wartosc>=66.30 | sprawdza, czy pierwsza wartość, jest większa lub równa drugiej |
| <= | mniejszy lub równy | wartosc<=66.30 | sprawdza, czy pierwsza wartość, jest mniejsza lub równa drugiej |
| != lub <> | różny | wartosc !=66.30 | sprawdza, czy dwie wartości są różne |
| IS NOT NULL |  | adres IS NOT NULL | sprawdza, czy pole posiada wartość |
| IS NULL |  | adres IS NULL | sprawdza, czy pole jest puste |
| BETWEEN |  | cena BETWEEN 25.30 AND 90.20 | Sprawdza czy wartość należy do podanego przedziału wartości |
| IN |  | miejscowosc IN('Krasnystaw', 'Warszawa') | sprawdza, czy wartość należy do podanego zbioru wartości |
| NOT NOT IN |  | miejscowosc NOT IN('Krasnystaw', 'Warszawa') | sprawdza, czy wartość nie należy do podanego zbioru wartości |
| LIKE | wyszukiwanie wzorca | nazwisko LIKE 'Józef %' | sprawdza, czy wartość pasuje do określonego wzorca |
| NOT LIKE | wyszukiwanie wzorca | nazwisko NOT LIKE 'Józef %' | sprawdza, czy wartość nie pasuje do określonego wzorca |
| REGEXP | wyrażenie regularne | nazwisko REGEXP '^Na.\*' | sprawdza, czy wartość pasuje do określonego wyrażenia regularnego. Nie znamy teorii wyrażeń regularnych, więc tego operatora nie będziemy stosować |

**LIKE** – wybór wierszy spełniających określone wyrażenie znakowe, np.

SELECT \* FROM nazwa\_tabeli WHERE nazwa\_kolumny LIKE wyrażenie

**Przykład 1:**

Dla tabeli klienci możemy wyświetlić wszystkie rekordy których nazwiska zaczynają się na „Kow”. Polecenie będzie miało następującą postać:

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM klienci WHERE nazwisko [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-comparison-functions.html#operator_like) 'Kow%'

**Przykład 2:**

Aby wyświetlić klientów których numer telefonu kończy się frazą 123 możemy także skorzystać z operatora **LIKE**.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM klienci WHERE telefon [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-comparison-functions.html#operator_like) '%123'

Przykład 3:

Jeżeli chcielibyśmy wyświetlić listę klientów których druga litera imienia to „a” także jest to możliwe:

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) \* FROM klienci WHERE imie [LIKE](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/string-comparison-functions.html#operator_like) '\_a%'

Poniżej przedstawiona została tabela ze wszelkimi możliwymi znakami specjalnymi, wraz ze znaczeniem i przykładem.

Tabela 4. Tabela znaków specjalnych

|  |  |  |
| --- | --- | --- |
| Znak specjalny: | Znaczenie: | Przykład: |
| \_ | Dowolny pojedynczy znak | LIKE ‘\_c\_’ – oznacza wszystkie trzy literowe wyrazy, których środkowa litera to c |
| % | Dowolny ciąg znaków o dowolnej długości, także zerowej | LIKE ‘c%’ – oznacza wszystkie wyrazy o dowolnej długości rozpoczynające się od litery c |
| [] | Dowolny ciąg znaków z podanego zakresu | LIKE ‘[a-g]%’ – oznacza wszystkie wyrazy o dowolnej długości których pierwsza litera należy do zakresu a–g. |
| [^] | Dowolny ciąg znaków nie pasujący do zakresu | LIKE ‘[^a-g]%’ – oznacza wszystkie wyrazy o dowolnej długości których pierwsza litera jest z poza zakresu a–g. |

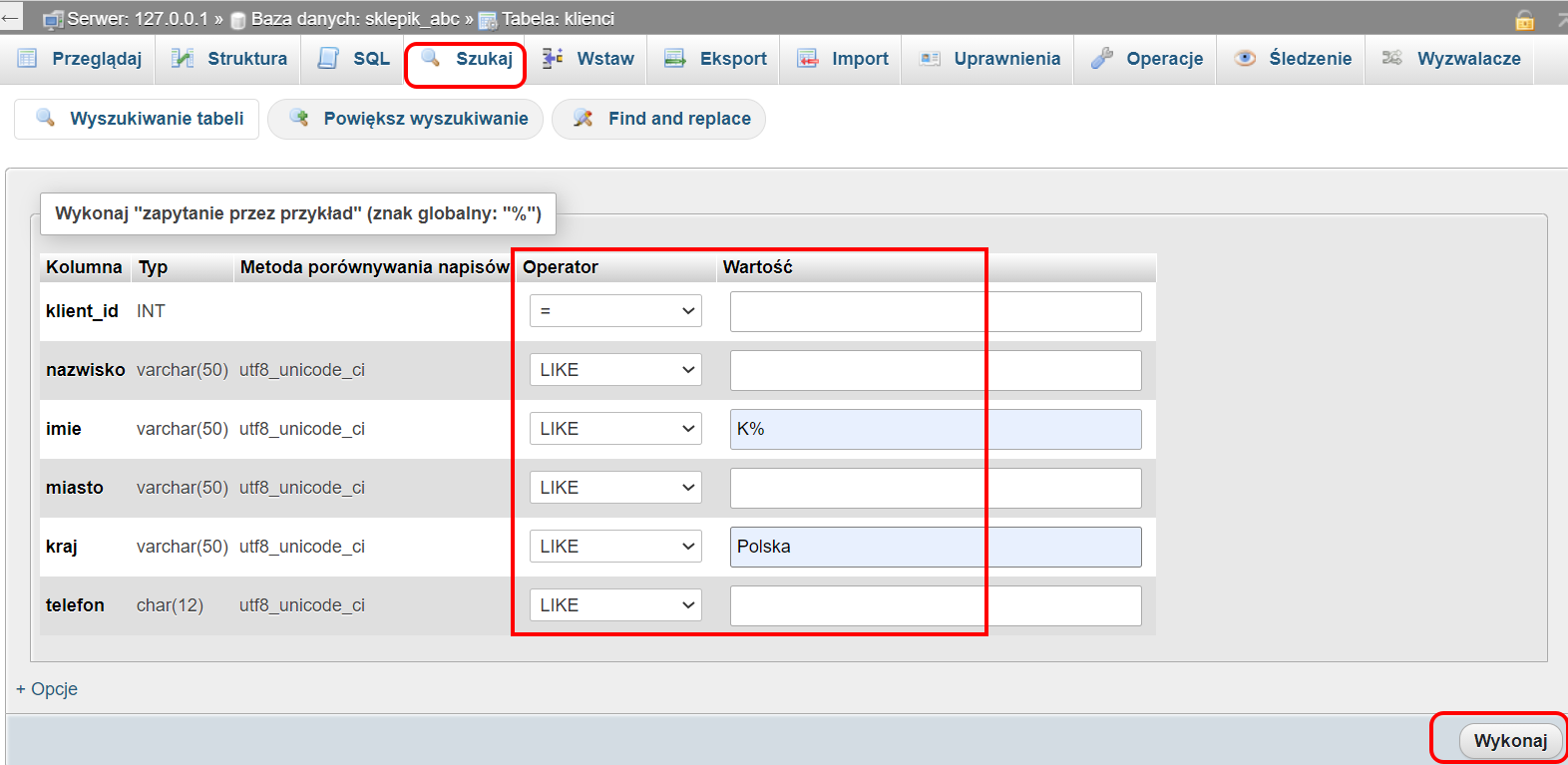
• **BETWEEN** – pozwala przy wykorzystaniu operatora WHERE określić zakresy, np.

SELECT \* FROM nazwa\_tabeli WHERE (nazwa\_kolumny BETWEEN x AND y)

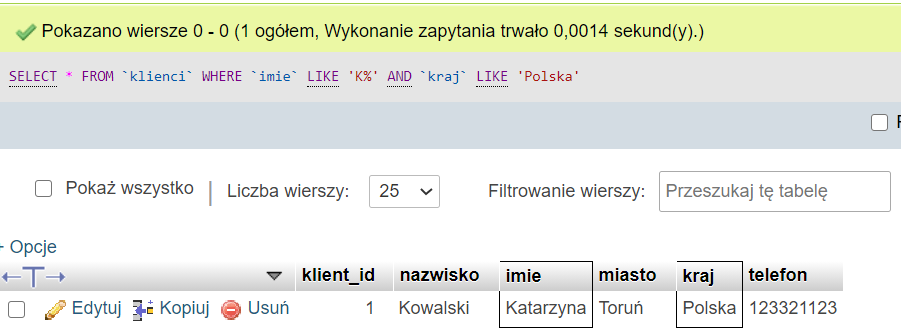
* **IN** – pozwala na wskazanie rekordów należących do określonego zbioru. Poniższe polecenie:

SELECT \* FROM produkt WHERE (id\_produkt IN (5, 15));

Najprostszym zapytaniem SQL jakie możemy wykonać na stworzonej przez nas tabeli to forma graficzna:

****

Po wykonaniu tego polecenia wygeneruje się zapytanie:

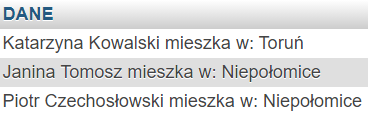


Zapytanie to zwróci nam wszystkie rekordy z tabeli posortowane wg klucza głównego rosnąco.

* Jeśli chcemy odwrócić sortowanie, musimy użyć **ORDER BY**, wskazać kolumnę po której będziemy sortować oraz metodę (ASC – rosnąco, DESC – malejąco)
* Możemy również wyświetlić ilość rekordów dzięki poleceniu **LIMIT**, stosowanemu na końcu zapytania
* Klauzula **DISTINCT** powoduje usunięcie powtarzających się wierszy. (SELECT DISTINCT miasto FROM klienci; - wyświetla miasta z których są klienci)

Jeśli chcemy połączyć kilka kolumn w jedną stosujemy funkcję **CONCAT**.

[SELECT](http://localhost/phpmyadmin/url.php?url=https://dev.mysql.com/doc/refman/5.5/en/select.html) CONCAT(imie,' ', nazwisko, ' mieszka w: ',miasto) AS DANE FROM klienci



**Zadanie 1:** Wstaw do tabel w bazie sklepik\_abc co najmniej po 20 rekordów (kategorie i producenci mogą mieć po 10 rekordów). Po wykonaniu wstawiania rekordów wyeksportuj bazę do pliku sql.

1. <http://www.pytlak.pr.radom.pl/mysql/typy_danych_mysql.html>

   ## Relacje w MySQL – poznajemy zbiory i relacje w MySQL za pomocą zapytań SQL

   <https://iwona.giat.pl/2019/02/23/kurs-mysql-relacje-miedzy-tabelami-cz-2/#more-6704> [↑](#endnote-ref-1)