

# 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<sup>i</sup>

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

|             |                         |                                                                                    |
|-------------|-------------------------|------------------------------------------------------------------------------------|
| <b>TIME</b> | 3 bajty                 | Data w formacie HH:MM:SS                                                           |
| <b>ENUM</b> | 1 lub 2 bajty           | Enumeracja (wyliczenie). W kolumnie może się znaleźć jedna z podanych wartości     |
| <b>SET</b>  | 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 używać 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*

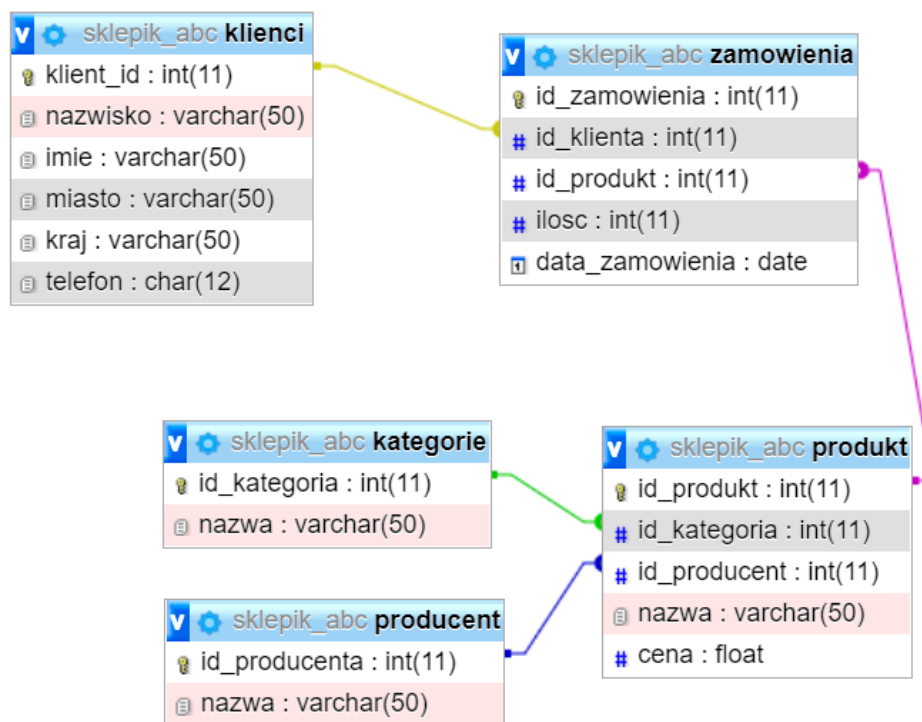
Nasza baza będzie rejestrować sprzedaż książek w księgarni internetowej. W tym celu będą utworzone następujące tabele:

- 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 księgarnia\_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.

#### 2. Lub



## Bazy danych

**Utwórz bazę danych**




### 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).

**Usuń bazę danych**

- **Usuń bazę danych (DROP)**

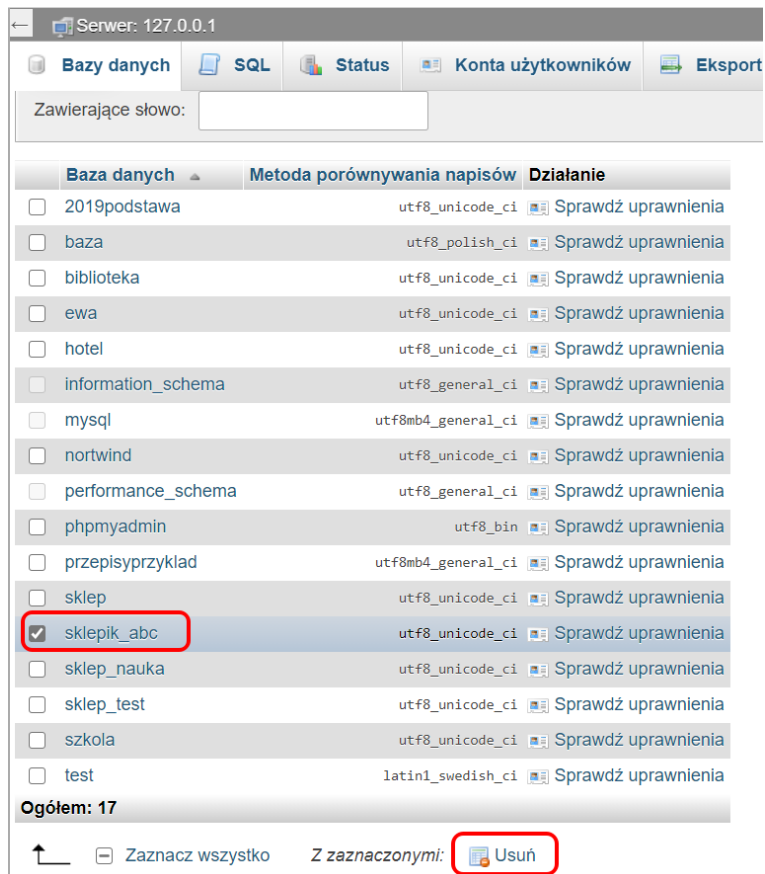
3. Otwieramy zakładkę *Operacje*, a następnie *Usuń bazę danych*.

**Potwierdź**

Masz zamiar ZNISZCZYĆ całą bazę danych! Czy na pewno chcesz wykonać "DROP DATABASE `sklepik\_abc`"?



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:

Serwer: 127.0.0.1 » Baza danych: sklepik\_abc

Struktura SQL Szukaj Zapytanie Eksport Import Operacje Uprawnienia Procedury i funkcje Więcej

Nazwa tabeli: produkt Dodaj 1 kolumna(y) Wykonaj

| Nazwa                        | Typ     | Długość/Wartości | Ustawienia domyślne | Metoda porównywania napisów | Atrybuty | Null                     | Indeks  | A..I                                |
|------------------------------|---------|------------------|---------------------|-----------------------------|----------|--------------------------|---------|-------------------------------------|
| id_produkt                   | INT     |                  | Brak                |                             |          | <input type="checkbox"/> | PRIMARY | <input checked="" type="checkbox"/> |
| Wybierz z centralnych kolumn |         |                  |                     |                             |          |                          |         |                                     |
| id_kategoria                 | INT     |                  | Brak                |                             |          | <input type="checkbox"/> | ---     | <input type="checkbox"/>            |
| Wybierz z centralnych kolumn |         |                  |                     |                             |          |                          |         |                                     |
| id_producent                 | INT     |                  | Brak                |                             |          | <input type="checkbox"/> | ---     | <input type="checkbox"/>            |
| Wybierz z centralnych kolumn |         |                  |                     |                             |          |                          |         |                                     |
| nazwa                        | VARCHAR | 50               | Brak                | utf8_unicode_ci             |          | <input type="checkbox"/> | ---     | <input type="checkbox"/>            |
| Wybierz z centralnych kolumn |         |                  |                     |                             |          |                          |         |                                     |
| cena                         | FLOAT   |                  | Brak                |                             |          | <input type="checkbox"/> | ---     | <input type="checkbox"/>            |
| Wybierz z centralnych kolumn |         |                  |                     |                             |          |                          |         |                                     |

**Struktura**

Komentarze tabeli: Sortowanie: Silnik składowania: InnoDB

## Podgląd SQL

```
CREATE TABLE `sklepik_abc`.`produkt` ( `id_produkt` INT NOT NULL AUTO_INCREMENT ,
`id_kategoria` INT NOT NULL , `id_producent` INT NOT NULL , `nazwa` VARCHAR(50) NOT
NULL , `cena` FLOAT NOT NULL , PRIMARY KEY (`id_produkt`)) ENGINE = InnoDB;
```

Tabela 2. tworzenie tabeli zamówienia

Serwer: 127.0.0.1 » Baza danych: sklepik\_abc

Struktura SQL Szukaj Zapytanie Eksport Import Operacje Uprawnienia Procedury i funkcje Więcej

Nazwa tabeli: zamówienia Dodaj 1 kolumna(y) Wykonaj

| Nazwa                        | Typ  | Długość/Wartości | Ustawienia domyślne | Metoda porównywania napisów | Atrybuty | Null                     | Indeks  | A..I                                |
|------------------------------|------|------------------|---------------------|-----------------------------|----------|--------------------------|---------|-------------------------------------|
| id_zamowienia                | INT  |                  | Brak                |                             |          | <input type="checkbox"/> | PRIMARY | <input checked="" type="checkbox"/> |
| Wybierz z centralnych kolumn |      |                  |                     |                             |          |                          |         |                                     |
| id_klienta                   | INT  |                  | Brak                |                             |          | <input type="checkbox"/> | ---     | <input type="checkbox"/>            |
| Wybierz z centralnych kolumn |      |                  |                     |                             |          |                          |         |                                     |
| id_produkt                   | INT  |                  | Brak                |                             |          | <input type="checkbox"/> | ---     | <input type="checkbox"/>            |
| Wybierz z centralnych kolumn |      |                  |                     |                             |          |                          |         |                                     |
| ilosc                        | INT  |                  | Brak                |                             |          | <input type="checkbox"/> | ---     | <input type="checkbox"/>            |
| Wybierz z centralnych kolumn |      |                  |                     |                             |          |                          |         |                                     |
| data_zamowienia              | DATE |                  | Brak                |                             |          | <input type="checkbox"/> | ---     | <input type="checkbox"/>            |
| Wybierz z centralnych kolumn |      |                  |                     |                             |          |                          |         |                                     |

**Struktura**

Komentarze tabeli: Sortowanie: Silnik składowania: InnoDB

```
CREATE TABLE `sklepik_abc`.`zamowienia` ( `id_zamowienia` INT NOT NULL AUTO_INCREMENT , `id_klienta` INT NOT
NULL , `id_produkt` INT NOT NULL , `ilosc` INT NOT NULL , `data_zamowienia` DATE 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:                                                                                                                                                                                                                                                                                                                                                                                                                                           |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre>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) );</pre> | <pre>CREATE TABLE nazwa_tabeli1 ( nazwa_kolumny1 typ_danych PRIMARY KEY, nazwa_kolumny2 typ_danych FOREIGN KEY REFERENCES nazwa_tabeli2 (nazwa_kolumny3) );</pre> <p>Przykład na zwany klucz obcy:</p> <pre>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) )</pre> |

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

| MySQL:                                                                   | SQL Server / Oracle / MS Access:                                                                                                       |
|--------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <pre>ALTER TABLE nazwa_tabeli DROP FOREIGN KEY nazwa_klucza_obcego</pre> | <pre>ALTER TABLE nazwa_tabeli DROP CONSTRAINT nazwa_klucza_obcego Np. ALTER TABLE Pracownicy DROP FOREIGN KEY fk_PracownikMiasto</pre> |

### 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 między tabelą produkt a kategorie na poniższym przykładzie:

| ewa_kategorie |                        |
|---------------|------------------------|
| 💡             | id_kategorie : int(11) |
| 📄             | nazwa : varchar(50)    |

| ewa_produkty |                        |
|--------------|------------------------|
| 💡            | id_produkty : int(11)  |
| #            | id_kategorie : int(11) |
| #            | cena : float           |
| 📄            | nazwa : varchar(50)    |

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 nie stoi na przeszkodzie by klucz główny tabeli był tekstowy.

Tabele mogą używać jednej z dwóch rodzajów relacji:

- 
 jeden do wielu (1:n) np. relacja pomiędzy produktem i kategorię
- 
 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 DELETE    | CASCADE   |
| ON UPDATE    | CASCADE   |
| Ograniczenia | SET NULL  |
|              | NO ACTION |
|              | RESTRICT  |

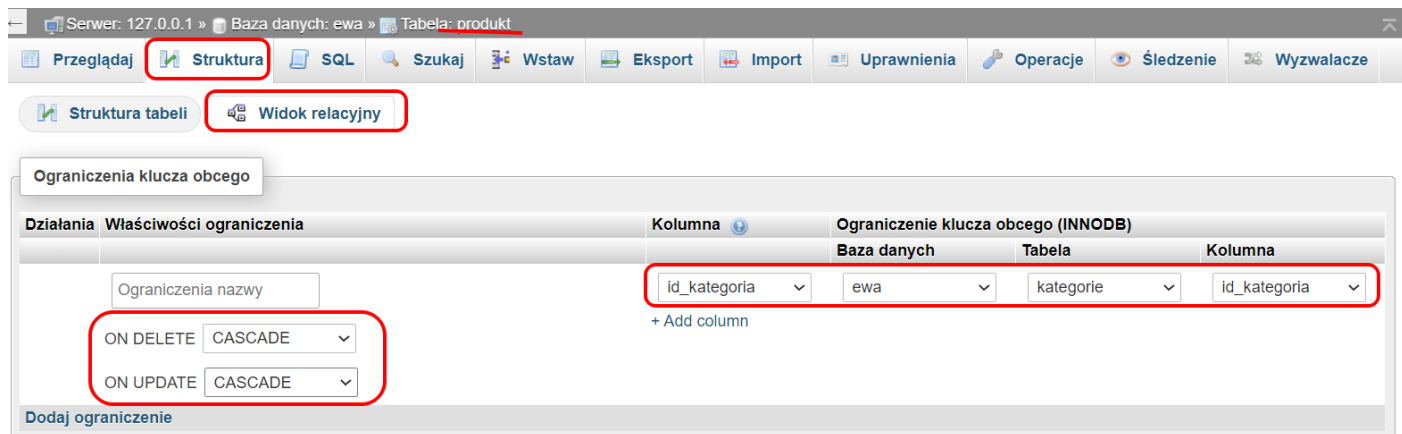
ON UPDATE:

- restrict** oznacza, iż nie będzie można zaktualizować wiersza z tabeli *kategorie*, jeśli posiada powiązanie w tabeli *produkty*
- cascade** oznacza, że jeżeli z jakiegoś względów będziemy potrzebowali zmienić np. długość klucza głównego w tabeli *kategorie* (np. poprzedzić ID zerami), to automatycznie zmiana zostanie naniesiona na kolumnę *id\_kategorie* w tabeli *produkty*
- set null** oznacza, że jeżeli zmieni się wartość *id\_kategorie* w tabeli *kategorie*, wiersze w tabeli *produkty*, które utracą powiązanie (brak dopasowania *id\_kategorie*), zmienią wartość *id\_kategorie* 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 *produkty*
- cascade** oznacza, że wraz z wierszem w tabeli *kategorie* usunięte zostaną wszystkie wiersze z tabeli *produkty*, powiązane z tym rekordem
- set null** oznacza, że zostanie wprowadzona wartość null w tabeli *produkty* w polu *id\_kategorie*, 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

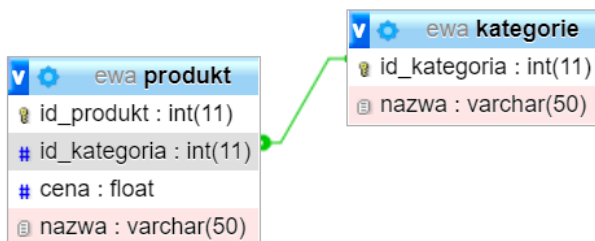
Polecenie na tworzenie powyższego przykładu:

```
ALTER TABLE produkt ADD FOREIGN KEY (id_kategoria) REFERENCES kategorie(
id_kategoria) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE produkt ADD FOREIGN KEY (id_kategoria) REFERENCES kategorie(id_kategoria) ON DELETE CASCADE ON UPDATE CASCADE;
```

↑  
klucz obcy w tabeli produkt

↑  
klucz główny w tabeli kategorie



Rysunek 2. Widok projektu bazy- pokazana relacja

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

Modyfikacja tabeli produkt:

Serwer: 127.0.0.1 » Baza danych: sklepik\_abc » Tabela: produkt

Przeglądaj Struktura SQL Szukaj Wstaw Eksport Import Uprawnienia Operacje Śledzenie Wyzwalacze

Struktura tabeli Widok relacyjny

### Ograniczenia klucza obcego

| Działania          | Właściwości ograniczenia | Kolumna      | Ograniczenie klucza obcego (INNODB) |           |               |
|--------------------|--------------------------|--------------|-------------------------------------|-----------|---------------|
|                    |                          |              | Baza danych                         | Tabela    | Kolumna       |
| Ograniczenia nazwy |                          | id_kategoria | sklepik_abc                         | kategorie | id_kategoria  |
| ON DELETE          | CASCADE                  |              |                                     |           |               |
| ON UPDATE          | CASCADE                  |              |                                     |           |               |
| + Add column       |                          |              |                                     |           |               |
| Ograniczenia nazwy |                          | id_producent | sklepik_abc                         | producent | id_producenta |
| ON DELETE          | CASCADE                  |              |                                     |           |               |
| ON UPDATE          | CASCADE                  |              |                                     |           |               |
| + Add column       |                          |              |                                     |           |               |

Dodaj ograniczenie

```
ALTER TABLE `produkt` ADD FOREIGN KEY (`id_kategoria`) REFERENCES `kategorie` (`id_kategoria`) ON DELETE CASCADE ON UPDATE CASCADE;
```

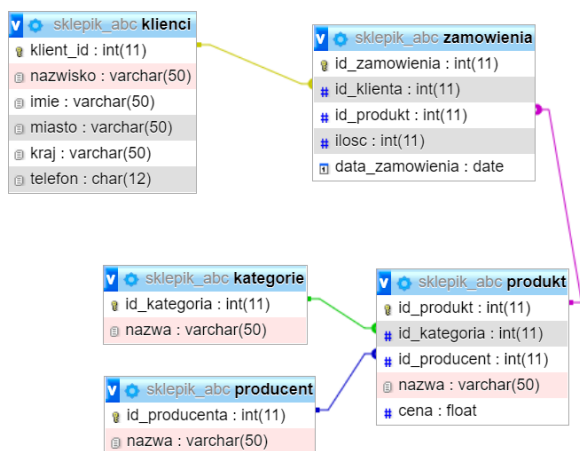
```
ALTER TABLE `produkt` ADD FOREIGN KEY (`id_producent`) REFERENCES `producent` (`id_producenta`) ON DELETE CASCADE ON UPDATE CASCADE;
```

Tworzenie kluczy obcych w tabeli zamówienia:

```
ALTER TABLE `zamowienia` ADD FOREIGN KEY (`id_klienta`) REFERENCES `klienci` (`klient_id`) ON DELETE CASCADE ON UPDATE CASCADE;
```

```
ALTER TABLE `zamowienia` ADD FOREIGN KEY (`id_produkt`) REFERENCES `produkt` (`id_produkt`) ON DELETE CASCADE ON UPDATE 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 VIEW z_polski AS SELECT * FROM `klienci_do_usuniecia` WHERE `kraj`
='Polska';
```

- Usunięcie widoku:

```
DROP VIEW z_polski;
```

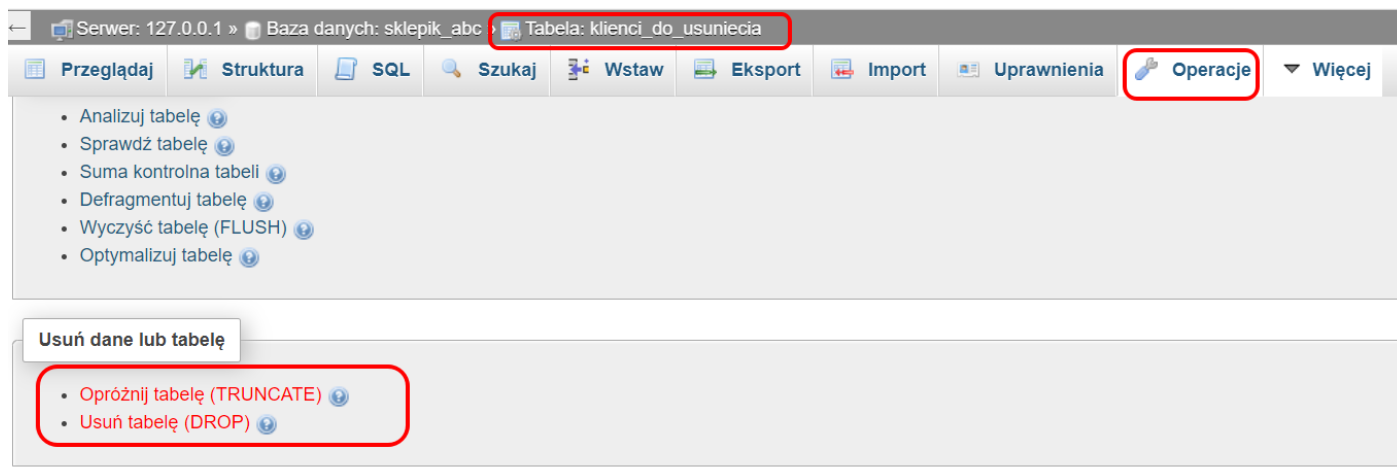
- Usunięcie zawartości tabeli

```
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 TABLE `producent` ADD `kraj_producenta` VARCHAR(50) NOT NULL DEFAULT 'Polska' AFTER `nazwa`;
```

- Modyfikacja tabeli producent w istniejącej tabeli nazwa ustawienie wartości NOT NULL

```
ALTER TABLE producent MODIFY nazwa varchar(50) NOT NULL;
```

- Modyfikacja tabeli producent przez usunięcie kolumny kraj\_producenta

```
ALTER TABLE producent DROP COLUMN kraj_producenta;
```

- Modyfikacja tabeli producent przez dodanie klucza głównego id\_producenta

```
ALTER TABLE producent ADD CONSTRAINT PRIMARY KEY (id_producenta);
```

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

```
ALTER TABLE klienci CHANGE kraj kraj VARCHAR(50) DEFAULT 'Polska';
```

- Dodanie wartości unique (wartość 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

Przenieś kolumny

Aby przenieś kolumny, przeciągając je w górę i w dół.

klient\_id

imie

nazwisko

miasto

kraj

telefon

Wykonaj

Podgląd SQL

```
ALTER TABLE `klienci` CHANGE `imie` `imie` VARCHAR(50) CHARACTER SET utf8 COLLATE utf8_unicode_ci NULL 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, ... , wartośćN);
```

### Przykład 1: wstawianie do tabeli klienci

```
INSERT INTO klienci(klient_id, nazwisko, imie, miasto, kraj, telefon)
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 INTO `kategorie` (`id_kategoria`, `nazwa`) VALUES (NULL, 'spożywcze'), (NULL, 'chemiczne'), (NULL, 'sprzęt sportowy'), (NULL, 'instrumenty muzyczne'), (NULL, 'RTV');
```

Wstawianie rekordów w formie graficznej do tabeli produkt.

| Kolumna      | Typ         | Funkcja | Null | Wartość              |
|--------------|-------------|---------|------|----------------------|
| id_produkt   | int(11)     |         |      |                      |
| id_kategoria | int(11)     |         |      | 3 - spożywcze        |
| id_producent | int(11)     |         |      | Milka - 3            |
| nazwa        | varchar(50) |         |      | czekolada Milka 200g |
| cena         | float       |         |      | 3,5                  |

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 produkt SET 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:

Server: 127.0.0.1 » Baza danych: sklepik\_abc » Tabela: produkt

Przeglądaj Struktura **SQL** Szukaj Wstaw Eksport Import Uprawnienia Operacje Śledzenie

Uruchom zapytanie SQL/zapytania w tabeli sklepik\_abc.produkt:

```
1 UPDATE `produkt` SET `id_produkt`=[value-1], `id_kategoria`=[value-2], `id_producent`=[value-3], `nazwa`=[value-4], `cena`=[value-5] WHERE 1
```

Kolumny

- id\_produkt
- id\_kategoria
- id\_producent
- nazwa
- cena

SELECT \* SELECT INSERT **UPDATE** DELETE Wyczyść Format

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

+ Opcje

|                          |                    | klient_id | nazwisko      | imie        | miasto | kraj   | telefon   |
|--------------------------|--------------------|-----------|---------------|-------------|--------|--------|-----------|
| <input type="checkbox"/> | Edytuj Kopiuj Usuń | 1         | Anna          | Mucha       | Toruń  | Polska | 123321321 |
| <input type="checkbox"/> | Edytuj Kopiuj Usuń | 2         | Janina        | Tomaszewska |        |        | 54        |
| <input type="checkbox"/> | Edytuj Kopiuj Usuń | 3         | Czechosłowski | Piotr       |        |        | 2         |
| <input type="checkbox"/> | Edytuj Kopiuj Usuń | 4         | Janicki       | Adam        | Berlin | Niemcy | 32323262  |

Null: ☐  
Naciśnij ESC, aby anulować edycję.

Wygeneruje się polecenie na UPDATE:

✓ Pokazano wiersze 0 - 3 (4 ogółem, Wykonanie zapytania trwało 0,0007 sekund(y).)

```
SELECT * FROM `klienci`
```

```
UPDATE `klienci` SET `miasto` = 'Toruń' WHERE `klienci`.`klient_id` = 1;
```

## 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 FROM `produkt` WHERE `id_produkt`=2;
```

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

```
DELETE 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 INTO `produkt` (`id_produkt`, `id_kategoria`, `id_producent`, `nazwa`, `cena`) VALUES (NULL, '1', '1', 'odkurzacz', '150'), (NULL, '3', '1', '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 FROM produkt ORDER BY cena DESC LIMIT 1;
```

Oczywiście usunąć rekordy można w sposób graficzny:

|                                                     | id_produkt | id_kategoria | id_producent | nazwa     | cena |
|-----------------------------------------------------|------------|--------------|--------------|-----------|------|
| <input type="checkbox"/> Edytuj  Kopiuj <b>Usun</b> | 4          | 1            | 1            | odkurzacz | 150  |
| <input type="checkbox"/> Edytuj  Kopiuj  Usun       | 7          | 1            | 1            | Mikser    | 300  |

☐ Zaznacz wszystko    Z zaznaczonymi: Edytuj Kopiuj Usun Eksport

**Potwierdź**

Czy na pewno chcesz wykonać "DELETE FROM `produkt` WHERE `produkt`.`id\_produkt` = 4"?

**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                                                                                                               |
| <b>IS NOT NULL</b> |                              | adres IS NOT NULL                           | sprawdza, czy pole posiada wartość                                                                                                                 |
| <b>IS NULL</b>     |                              | adres IS NULL                               | sprawdza, czy pole jest puste                                                                                                                      |
| <b>BETWEEN</b>     |                              | cena BETWEEN 25.30 AND 90.20                | Sprawdza czy wartość należy do podanego przedziału wartości                                                                                        |
| <b>IN</b>          |                              | miescowosc IN('Krasnystaw', 'Warszawa')     | sprawdza, czy wartość należy do podanego zbioru wartości                                                                                           |
| <b>NOT NOT IN</b>  |                              | miescowosc NOT IN('Krasnystaw', 'Warszawa') | sprawdza, czy wartość nie należy do podanego zbioru wartości                                                                                       |
| <b>LIKE</b>        | wyszukiwanie wzorca          | nazwisko LIKE 'Józef %'                     | sprawdza, czy wartość pasuje do określonego wzorca                                                                                                 |
| <b>NOT LIKE</b>    | wyszukiwanie wzorca          | nazwisko NOT LIKE 'Józef %'                 | sprawdza, czy wartość nie pasuje do określonego wzorca                                                                                             |
| <b>REGEXP</b>      | wyrażenie regularne          | nazwisko REGEXP '^Na.*'                     | sprawdza, czy wartość pasuje do określonego wyrażenia regularnego. Nie znamy teorii wyrażen 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 * FROM klienci WHERE nazwisko LIKE 'Kow%'
```

**Przykład 2:**

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

```
SELECT * FROM klienci WHERE telefon LIKE '%123'
```

Przykład 3:

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

```
SELECT * FROM klienci WHERE imie 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:

Serwer: 127.0.0.1 » Baza danych: sklepik\_abc » Tabela: klienci

Przeglądaj Struktura SQL Szukaj Wstaw Eksport Import Uprawnienia Operacje Śledzenie Wyzwalacze

Wyszukiwanie tabeli Powiększ wyszukiwanie Find and replace

Wykonaj "zapytanie przez przykład" (znak globalny: "%")

| Kolumna   | Typ                         | Metoda porównywania napisów | Operator | Wartość |
|-----------|-----------------------------|-----------------------------|----------|---------|
| klient_id | INT                         |                             | =        |         |
| nazwisko  | varchar(50) utf8_unicode_ci |                             | LIKE     |         |
| imie      | varchar(50) utf8_unicode_ci |                             | LIKE     | K%      |
| miasto    | varchar(50) utf8_unicode_ci |                             | LIKE     |         |
| kraj      | varchar(50) utf8_unicode_ci |                             | LIKE     | Polska  |
| telefon   | char(12) utf8_unicode_ci    |                             | LIKE     |         |

+ Opcje

Wykonaj

Po wykonaniu tego polecenia wygeneruje się zapytanie:

✓ Pokazano wiersze 0 - 0 (1 ogółem, Wykonanie zapytania trwało 0,0014 sekund(y).)

```
SELECT * FROM `klienci` WHERE `imie` LIKE 'K%' AND `kraj` LIKE 'Polska'
```

☐ Pokaż wszystko | Liczba wierszy: 25 ▼ Filtrowanie wierszy: Przeszukaj tę tabelę

- Opcje

|                                                                                               | klient_id | nazwisko | imie      | miasto | kraj   | telefon   |
|-----------------------------------------------------------------------------------------------|-----------|----------|-----------|--------|--------|-----------|
| <input type="checkbox"/> Edytuj <input type="checkbox"/> Kopiuj <input type="checkbox"/> Usuń | 1         | Kowalski | Katarzyna | Toruń  | Polska | 123321123 |

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 CONCAT(imie, ' ', nazwisko, ' ', mieszka w:
',miasto) AS DANE FROM klienci
```

#### DANE

Katarzyna Kowalski mieszka w: Toruń

Janina Tomosz mieszka w: Niepołomice

Piotr Czechosłowski mieszka w: Niepołomice

**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.

---

<sup>i</sup> [http://www.pytlak.pr.radom.pl/mysql/typy\\_danych\\_mysql.html](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>