Kurs
–
Podstawy MySQL

Krótkie info.

Autorem kursu jest Piotr Jędrusik.
Kurs jest własnością serwisu MySQL FAQ www.mysqlfaq.prv.pl , email: mysqlfaq@twister.pl.

1. Tworzymy bazę.

Stworzymy pierwszą bazę o nazwie user.

mysql> create database user;

Poprawne wykonanie polecenia powinno dać wynik:

Query OK, 1 row affected (0.00 sec)

```
mysql> create database user;
Query OK, 1 row affected (0.01 sec)
```

Po stworzeniu bazy, możemy przejść do tworzenia tabel. Jednak samo stworzenie bazy, nie oznacza że staje się ona bazą bieżącą i możemy na niej wykonywać operacje.
Pokazuje to poniższe polecenie, sprawdzające jaka baza jest obecnie używana.

mysql> select database();

Efektem polecenia będzie:

```
mysql> select database();
+------------+
| database() |
+------------+
|            |
+------------+
1 row in set (0.03 sec)
```

Aby móc korzystać z konkretnej bazy, należy wybrać ją jako bazę bieżącą:

mysql> use user;

Po poprawnym wykonaniu polecenia, MySQL powinien zwrócić komunikat:

Database changed

```
mysql> use user;
Database changed
```

OK. Teraz możemy ponownie sprawdzić jaka baza jest bazą bieżącą:

mysql> select database();

```
mysql> select database();
+------------+
| database() |
+------------+
| user       |
+------------+
1 row in set (0.00 sec)
```

Od tej pory wszystkie operacje będą wykonywane na bazie user.

Innym sposobem wyboru bazy jest wymienienie jej nazwy w wierszu poleceń podczas wywołania mysql:

C:/>mysql user
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 5 to server version: 3.23.32

mysql> select database();

```
mysql> select database();
+------------+
| database() |
+------------+
| user       |
+------------+
1 row in set (0.00 sec)
```

2. Tworzymy tabele.

Stworzymy teraz tabelę dane, w której będziemy przechowywać dane odnośnie użytkowników.

```
mysql> create table dane
    -> (
    -> imie varchar(15) not null,
    -> nazwisko varchar(20) not null,
    -> email varchar(50) null,
    -> ulica varchar(50) null,
    -> miasto varchar(50) null,
    -> data_ur date null default '0000-00-00'
    -> );
```

Aby sprawdzić czy tabela została stworzona poprawnie wykonujemy poniższe polecenie:

mysql> describe dane;

```
mysql> describe dane;
+----------+-------------+------+-----+------------+-------+
| Field    | Type        | Null | Key | Default    | Extra |
+----------+-------------+------+-----+------------+-------+
| imie     | varchar(15) |      |     |            |       |
| nazwisko | varchar(20) |      |     |            |       |
| email    | varchar(50) | YES  |     | NULL       |       |
| ulica    | varchar(50) | YES  |     | NULL       |       |
| miasto   | varchar(50) | YES  |     | NULL       |       |
| data_ur  | date        | YES  |     | 0000-00-00 |       |
+----------+-------------+------+-----+------------+-------+
6 rows in set (0.03 sec)
```

Instrukcja describe (można ją używać skrótowo desc) pokazuje informacje odnośnie podanej tabeli.

A jak sprawdzić jakie tabele zawiera konkretna baza ?
Umożliwia to polecenie SHOW.

mysql> show tables;

```
mysql> show tables;
+----------------+
| Tables_in_user |
+----------------+
| dane           |
+----------------+
1 row in set (0.00 sec)
```

W ten sam sposób możemy sprawdzić jakie bazy istnieją na serwerze.

mysql> show databases;

```
mysql> show databases;
+----------+
| Database |
+----------+
| Test     |
| mysql    |
| user     |
+----------+
3 rows in set (0.00 sec)
```

Wracając do tworzenia tabel. Aby nie wpisywać całej instrukcji tworzenia tabeli możemy wcześniej stworzyć plik z rozszerzeniem .sql, w którym wpisujemy całą składnię tworzenia tabeli, a następnie wywołujemy jedynie proste polecenie:

C:/>mysql user < user.sql

3. Dodawanie nowych rekordów.

Stworzyliśmy bazę user i tabelę dane. Teraz zajmijmy się wprowadzeniem kilku rekordów do tabeli z danymi użytkowników.

Dane dodawanego usera:
imię: Piotr,
nazwisko: Walczak
email: walus@plus.pl
ulica: Lesna
miasta: Warszawa
data urodzenia: 1980-02-25

A więc wprowadzamy dane:

```
mysql> insert into dane
    -> values('Piotr','Walczak','walus@plus.pl','Lesna','Warszawa','1980-02-25');
Query OK, 1 row affected (0.00 sec)
```

Dodajmy jeszcze jednego użytkownika, tym razem znając tylko jego imię, nazwisko i email.
imię: Adam,
nazwisko: Kowalski,
email: adam@firma.com.pl

```
mysql> insert into dane
    -> (imie,nazwisko,email)
    -> values ('Adam','Kowalski','adam@firma.com.pl');
Query OK, 1 row affected (0.00 sec)
```

Oczywiście zamiast wpisywać wszystkich naszych użytkowników ręcznie, możemy ich dane zgromadzić w pliku, z którego dane możemy wczytać do tabeli:

```
mysql>load data local infile ”dane.txt” into table dane;
```

Domyślnie instrukcja Load Data zakłada, że wartości kolumn będą ograniczone przez tabulatory i że wiersze będą się kończyć przy użyciu znaków końca wierszy. Zakłada się również, że wartości występują w kolejności, w której kolumny są przechowywane w tabeli. Możliwe jest zdefiniowanie innego formatu pliku lub określenie innego porządku kolumn. Jednak w tym kursie tym zajmować się nie będziemy.

Aby załadować dane z pliku możemy również użyć programu mysqlimport, który generuje instrukcję LOAD DATA, która pozwala na załadowanie danych z pliku dane.txt do tabeli dane.

```
mysqlimport --local user dane.txt
```

4. Pobieranie informacji.

Instrukcja SELECT pozwala pobierać i wyświetlać informacje z tabel, ogólnie lub bardzo szczegółowo.
Ogólna postać instrukcji SELECT jest następująca:

SELECT co_pobrać
FROM tabela lub tabele
WHERE warunki_które_muszą_spełniać_dane

Aby wyświetlić całą zawartość tabeli wykonujemy:

mysql> select * from dane;

```
mysql> select * from dane;
+---------+----------+----------------------+---------+-----------+------------+
| imie    | nazwisko | email                | ulica   | miasto    | data_ur    |
+---------+----------+----------------------+---------+-----------+------------+
| Piotr   | Walczak  | walus@plus.pl        | Lesna   | Warszawa  | 1980-02-25 |
| Grzegorz| Marzec   | grzegorz@firma.com.pl| Wiejska | Katowice  | 1955-01-26 |
| Maria   | Krzak    | krzak@polsska.pl     | Ulicowa | Warszawa  | 1965-03-15 |
| Daria   | Kurak    | daria@firma.pl       | Wiejska | Sosnowiec | 1980-05-01 |
| Wiesiek | Walc     | wiesiek@serwer.pl    | Miejska | Krakow    | 1973-12-24 |
| Karol   | Rutkowski| karol@opoka.pl       | Polna   | Krakow    | 1977-09-11 |
+---------+----------+----------------------+---------+-----------+------------+
6 rows in set (0.00 sec)
```

Można wybrać tylko jedną kolumnę z tabeli:

mysql> select email from dane;

```
mysql> select email from dane;
+----------------------+
| email                |
+----------------------+
| walus@plus.pl        |
| grzegorz@firma.com.pl|
| krzak@polsska.pl     |
| daria@firma.pl       |
| wiesiek@serwer.pl    |
| karol@opoka.pl       |
+----------------------+
6 rows in set (0.00 sec)
```

A także kilka kolumn:

mysql> select ulica,miasto from dane;

```
mysql> select ulica,miasto from dane;
+---------+-----------+
| ulica   | miasto    |
+---------+-----------+
| Lesna   | Warszawa  |
| Wiejska | Katowice  |
| Ulicowa | Warszawa  |
| Wiejska | Sosnowiec |
| Miejska | Krakow    |
| Polna   | Krakow    |
+---------+-----------+
6 rows in set (0.00 sec)
```

Można także przestawiać kolejność kolumn, a także podawać kilka razy jedną kolumnę.

mysql> select miasto,email,imie from dane;

```
mysql> select imie,miasto,email,imie from dane;
+---------+----------+----------------------+---------+
| imie    | miasto   | email                | imie    |
+---------+----------+----------------------+---------+
| Piotr   | Warszawa | walus@plus.pl        | Piotr   |
| Grzegorz| Katowice | grzegorz@firma.com.pl| Grzegorz|
| Maria   | Warszawa | krzak@polsska.pl     | Maria   |
| Daria   | Sosnowiec| daria@firma.pl       | Daria   |
| Wiesiek | Krakow   | wiesiek@serwer.pl    | Wiesiek |
| Karol   | Krakow   | karol@opoka.pl       | Karol   |
+---------+----------+----------------------+---------+
6 rows in set (0.01 sec)
```

5. Określenie kryteriów pobierania danych.

Aby ograniczyć zbiór pobranych rekordów przez instrukcję SELECT, należy użyć klauzuli WHERE, która określa kryteria wyboru wierszy.

Wybierzmy osoby, które urodziły się przed 01-01-1970.

mysql> select imie,nazwisko from dane where data_ur<'1970-01-01';

```
mysql> select imie,nazwisko from dane where data_ur<'1970-01-01';
+---------+----------+
| imie    | nazwisko |
+---------+----------+
| Grzegorz| Marzec   |
| Maria   | Krzak    |
+---------+----------+
2 rows in set (0.02 sec)
```

W kolejnym przykładzie spróbujmy wybrać osoby zamieszkałe w Krakowie.

mysql> select imie,nazwisko from dane where miasto='krakow';

```
mysql> select imie,nazwisko from dane where miasto='krakow';
+--------+----------+
| imie   | nazwisko |
+--------+----------+
| Wiesiek| Walc     |
| Karol  | Rutkowski|
+--------+----------+
2 rows in set (0.00 sec)
```

A teraz połączmy dwa poprzednie przykłady i wybierzmy osoby zamieszkałe poza Krakowem i urodzone po 01-01-1970.

```
mysql> select imie,nazwisko,miasto,data_ur from dane
   -> where data_ur>'1970-01-01'
   -> and miasto <> 'Krakow';
```

```
mysql> select imie,nazwisko,miasto,data_ur from dane
    -> where data_ur>'1970-01-01'
    -> and miasto <> 'Krakow';
+-------+----------+-----------+------------+
| imie  | nazwisko | miasto    | data_ur    |
+-------+----------+-----------+------------+
| Piotr | Walczak  | Warszawa  | 1980-02-25 |
| Daria | Kurak    | Sosnowiec | 1980-05-01 |
+-------+----------+-----------+------------+
2 rows in set (0.01 sec)
```

Przy wyborze danych możemy stosować operatory arytmetyczne, porównania i logiczne.

Operatory arytmetyczne:

+ dodawanie
- odejmowanie
* mnożenie
/ dzielenie

Operatory porównania:

< mniejsze niż
<= mniejsze niż lub równe
= równe
!= lub <> nierówne
>= większe niż lub równe
> większe niż

Operatory logiczne:

AND logiczne „i”
OR logiczne „lub”
NOT logiczne przeczenie

6. Sortowanie wyników zapytania.

Aby uporządkować pobierane rekordy, należy użyć klauzuli ORDER BY.
Na początek spróbujmy posortować pobierane dane według nazwiska (od A do Z).

mysql> select imie,nazwisko from dane order by nazwisko;

```
mysql> select imie,nazwisko from dane order by nazwisko;
+----------+-----------+
| imie     | nazwisko  |
+----------+-----------+
| Maria    | Krzak     |
| Daria    | Kurak     |
| Grzegorz | Marzec    |
| Karol    | Rutkowski |
| Wiesiek  | Walc      |
| Piotr    | Walczak   |
+----------+-----------+
6 rows in set (0.02 sec)
```

Możemy określić w jaki sposób sortować rekordy, czy w kierunku rosnącym ASC lub malejącym DESC.

A więc pobierzmy te same dane co w poprzednim przykładzie ale posortujmy je w odwrotnej kolejności czyli malejącej:

mysql> select imie,nazwisko from dane order by nazwisko desc;

```
mysql> select imie,nazwisko from dane order by nazwisko desc;
+----------+-----------+
| imie     | nazwisko  |
+----------+-----------+
| Piotr    | Walczak   |
| Wiesiek  | Walc      |
| Karol    | Rutkowski |
| Grzegorz | Marzec    |
| Daria    | Kurak     |
| Maria    | Krzak     |
+----------+-----------+
6 rows in set (0.00 sec)
```

Porządek rosnący jest domyślny, jeśli nie określimy inaczej.

Możemy sortować dwie kolumny:

mysql> select imie,nazwisko from dane order by imie asc, nazwisko asc;

```
mysql> select imie,nazwisko from dane
    -> order by imie asc, nazwisko asc;
+----------+-----------+
| imie     | nazwisko  |
+----------+-----------+
| Daria    | Kurak     |
| Grzegorz | Marzec    |
| Karol    | Rutkowski |
| Maria    | Krzak     |
| Piotr    | Walczak   |
| Wiesiek  | Walc      |
+----------+-----------+
6 rows in set (0.00 sec)
```

7. Ograniczanie wyników zapytań.

Gdy rezultatem zapytania jest bardzo dużo wierszy, możemy ograniczyć wynik używając klauzuli LIMIT. Klauzula LIMIT pozwala ograniczyć wynik zapytania do n wierszy całego rezultatu.

W naszej tabeli mamy 6 rekordów. Ograniczmy wynik zapytania do 3 rekordów:

mysql> select imie,email from dane limit 3;

```
mysql> select imie,email from dane limit 3;
+---------+----------------------+
| imie    | email                |
+---------+----------------------+
| Piotr   | walus@plus.pl        |
| Grzegorz | grzegorz@firma.com.pl |
| Maria   | krzak@polsska.pl     |
+---------+----------------------+
3 rows in set (0.00 sec)
```

Możemy wyświetlić wynik zapytania począwszy od jakiegoś rekordu, w naszym przypadku wyświetlmy rekordy począwszy od 2 (pierwszy rekord ma numer 0, nie 1), wyświetlając dwa kolejne rekordy:

mysql> select imie,email from dane limit 1,2;

```
mysql> select imie,email from dane limit 1,2;
+---------+----------------------+
| imie    | email                |
+---------+----------------------+
| Grzegorz | grzegorz@firma.com.pl |
| Maria   | krzak@polsska.pl     |
+---------+----------------------+
2 rows in set (0.00 sec)
```

8. Łączenie i nazywanie wartości kolumn wyjściowych.

Wybierzmy z naszej tabeli imiona i nazwiska, połączmy je w jedną kolumną o nazwie Imię – Nazwisko.

mysql> select concat(imie," ",nazwisko) as "Imie - Nazwisko" from dane;

```
mysql> select concat(imie," ",nazwisko) as "Imie - Nazwisko"
    -> from dane;
+-----------------+
| Imie - Nazwisko |
+-----------------+
| Piotr Walczak   |
| Grzegorz Marzec |
| Maria Krzak     |
| Daria Kurak     |
| Wiesiek Walc    |
| Karol Rutkowski |
+-----------------+
6 rows in set (0.02 sec)
```

9. Praca z datami.

Przetestujmy kilka przykładów, myślę że nie sprawia one zbytnich kłopotów.

mysql> select email,data_ur from dane where data_ur = '1980-02-25';

```
mysql> select email,data_ur from dane
    -> where data_ur = '1980-02-25';
+--------------+------------+
| email        | data_ur    |
+--------------+------------+
| walus@plus.pl | 1980-02-25 |
+--------------+------------+
1 row in set (0.00 sec)
```

mysql> select email,data_ur from dane where data_ur >= '1955-01-01' and data_ur <= '1980-01-01';

```
mysql> select email,data_ur from dane where
    -> data_ur >= '1955-01-01'
    -> and data_ur <= '1980-01-01';
+---------------------+------------+
| email               | data_ur    |
+---------------------+------------+
| grzegorz@firma.com.pl | 1955-01-26 |
| krzak@polsska.pl    | 1965-03-15 |
| wiesiek@serwer.pl   | 1973-12-24 |
| karol@opoka.pl      | 1977-09-11 |
+---------------------+------------+
4 rows in set (0.00 sec)
```

Aby sprawdzić lub pobrać część daty, można użyć funkcji takich jak YEAR(), MONTH(), DAYOFMONTH().

mysql> select imie,data_ur from dane where month(data_ur) = 9;

```
mysql> select imie,data_ur from dane
    -> where month(data_ur) = 9;
+-------+------------+
| imie  | data_ur    |
+-------+------------+
| Karol | 1977-09-11 |
+-------+------------+
1 row in set (0.02 sec)
```

mysql> select imie,data_ur from dane where year(data_ur) > 1970;

```
mysql> select imie,data_ur from dane
    -> where year(data_ur) > 1970;
+--------+------------+
| imie   | data_ur    |
+--------+------------+
| Piotr  | 1980-02-25 |
| Daria  | 1980-05-01 |
| Wiesiek | 1973-12-24 |
| Karol  | 1977-09-11 |
+--------+------------+
4 rows in set (0.00 sec)
```

Spróbujmy teraz stworzyć trochę bardziej skomplikowane zapytanie, obliczające wiek danej osoby.

```
mysql> select imie,nazwisko,
   -> floor((to_days(curdate()) - to_days(data_ur))/365) as wiek
   -> from dane;
```

```
mysql> select imie,nazwisko,
    -> floor((to_days(curdate()) - to_days(data_ur))/365) as wiek
    -> from dane;
+---------+-----------+------+
| imie    | nazwisko  | wiek |
+---------+-----------+------+
| Piotr   | Walczak   |   22 |
| Grzegorz | Marzec   |   47 |
| Maria   | Krzak     |   37 |
| Daria   | Kurak     |   22 |
| Wiesiek | Walc      |   28 |
| Karol   | Rutkowski |   24 |
+---------+-----------+------+
6 rows in set (0.02 sec)
```

Użyta tutaj funkcja FLOOR obcina część ułamkową wieku aby uzyskać liczbę całkowitą.

Po więcej informacji odnośnie funkcji operujących na datach odsyłam do działu PODSTAWY->Funkcje cz.IV.

10. Dopasowanie do wzorca.

Przy wybieraniu rekordów możemy wykonać operację dopasowania do wzorca, w ten sposób można wybrać rekordy bez podawania dokładnej wartości.
Aby wykonać operacje dopasowania należy użyć specjalnych operatorów LIKE lub NOT LIKE i określić ciąg znaków zawierających znaki zastępcze.

```
mysql> select imie,nazwisko from dane where nazwisko like 'w%';
```

```
mysql> select imie,nazwisko from dane
    -> where nazwisko like 'w%';
+---------+----------+
| imie    | nazwisko |
+---------+----------+
| Piotr   | Walczak  |
| Wiesiek | Walc     |
+---------+----------+
2 rows in set (0.00 sec)
```

Znak % oznacza dopasowanie do jakiejkolwiek sekwencji znaków.

mysql> select imie,email from dane where imie like '_____';

```
mysql> select imie,email from dane
    -> where imie like '_____';
+-------+------------------+
| imie  | email            |
+-------+------------------+
| Piotr | walus@plus.pl    |
| Maria | krzak@polsska.pl |
| Daria | daria@firma.pl   |
| Karol | karol@opoka.pl   |
+-------+------------------+
4 rows in set (0.00 sec)
```

Znak _ określa pojedynczy znak. W powyższym przykładzie aby wybrać imiona składające się z pięciu liter należy podać znak _ pięć razy.

I jeszcze jeden prosty przykład. Znajdźmy osoby, które skrzynki e-mail mają założone na serwerze firma.com.pl lub firma.pl:

mysql> select imie,nazwisko,email from dane where email like '%firma%';

```
mysql> select imie,nazwisko,email from dane
    -> where email like '%firma%';
+----------+----------+----------------------+
| imie     | nazwisko | email                |
+----------+----------+----------------------+
| Grzegorz | Marzec   | grzegorz@firma.com.pl |
| Daria    | Kurak    | daria@firma.pl       |
+----------+----------+----------------------+
2 rows in set (0.00 sec)
```

11. Tworzenie podsumowań.

Aby odrzucić w wyniku zapytania wszystkie wiersze, które się powtarzają używamy słowa kluczowego DISTINCT.

Polecenie bez użycia powyższego słowa kluczowego da nam wynik:

mysql> select miasto from dane;

```
mysql> select miasto from dane;
+-----------+
| miasto    |
+-----------+
| Warszawa  |
| Katowice  |
| Warszawa  |
| Sosnowiec |
| Krakow    |
| Krakow    |
+-----------+
6 rows in set (0.00 sec)
```

Gdy użyjemy słowa kluczowego, zapytanie zwróci nam wynik:

mysql> select distinct miasto from dane;

```
mysql> select distinct miasto from dane;
+-----------+
| miasto    |
+-----------+
| Warszawa  |
| Katowice  |
| Sosnowiec |
| Krakow    |
+-----------+
4 rows in set (0.05 sec)
```

czyli odrzuci rekordy, które się powtarzają.

Bardzo przydatna jest funkcja COUNT(), która zlicza liczbę wierszy otrzymanych w wyniku zapytania.

Poniższe polecenie zwróci nam ilość wszystkich rekordów w tabeli dane:

mysql> select count(*) from dane;

```
mysql> select count(*) from dane;
+----------+
| count(*) |
+----------+
|        6 |
+----------+
1 row in set (0.02 sec)
```

Możemy obliczyć ile osób pochodzi z Warszawy:

mysql> select count(*) from dane where miasto = 'warszawa';

```
mysql> select count(*) from dane
    -> where miasto = 'warszawa';
+----------+
| count(*) |
+----------+
|        2 |
+----------+
1 row in set (0.00 sec)
```

I jeszcze jeden przykład, w którym obliczymy ile osób mieszka w danym mieście:

mysql> select miasto, count(*) from dane group by miasto;

```
mysql> select miasto, count(*)
    -> from dane group by miasto;
+-----------+----------+
| miasto    | count(*) |
+-----------+----------+
| Katowice  |        1 |
| Krakow    |        2 |
| Sosnowiec |        1 |
| Warszawa  |        2 |
+-----------+----------+
4 rows in set (0.02 sec)
```

Ważna uwaga. W takim zapytaniu musimy użyć przed zliczeniem miast, grupowania wyników według miasta, gdyż w przeciwnym przypadku zapytanie zwróci nam błąd.

Oprócz funkcji zliczającej COUNT(), mamy również dostępne funkcje MIN(), MAX(), SUM() i AVG().

Wybierzmy teraz najstarszego użytkownika;

mysql> select min(data_ur) as "NAJSTARSZY" from dane;

```
mysql> select min(data_ur) as "NAJSTARSZY" from dane;
+------------+
| NAJSTARSZY |
+------------+
| 1955-01-26 |
+------------+
1 row in set (0.00 sec)
```

12. Pobieranie informacji z wielu tabel.

Na początek musimy stworzyć drugą tabelę. Będzie ona miała nazwę ‘skrzynka’ i powiedzmy, że będziemy w niej przechowywać dane odnoście ilości wiadomości e-mail, jakie dany użytkownik ma w swojej skrzynce.

```
mysql> create table skrzynka
    -> (
    -> email varchar(50) not null,
    -> listy int
    -> );
Query OK, 0 rows affected (0.00 sec)
```

Zobaczmy zatem jak będzie wyglądać nasza nowa tabela:

mysql> select * from skrzynka;

```
mysql> select * from skrzynka;
+----------------------+-------+
| email                | listy |
+----------------------+-------+
| walus@plus.pl        |     5 |
| grzegorz@firma.com.pl |     3 |
| krzak@polsska.pl     |    11 |
| daria@firma.pl       |     1 |
| wiesiek@serwer.pl    |     9 |
| karol@opoka.pl       |     7 |
+----------------------+-------+
6 rows in set (0.02 sec)
```

Przy wybieraniu danych z dwóch tabel nasze zapytanie będzie różniło się od dotychczasowych zapytań pod dwoma względami:
1. w klauzuli FROM musimy wymienić więcej niż jedną tabelę, gdyż dane mają być pobierane z więcej niż jednej tabeli;

FROM dane,skrzynka

2. w klauzuli WHERE określa się, że tabele dane i skrzynka są złączone przez dopasowanie wartości email w każdej tabeli:

WHERE . . . dane.email = skrzynka.email.

Wybierzmy teraz z tabeli dane - imię, nazwisko i email usera oraz z tabeli skrzynka - ilość listów jakie ma dany user:

```
mysql> select dane.imie,dane.nazwisko,dane.email,skrzynka.listy
    -> from dane,skrzynka
    -> where dane.email = skrzynka.email;
```

```
mysql> select dane.imie,dane.nazwisko,dane.email,skrzynka.listy
    -> from dane,skrzynka
    -> where dane.email = skrzynka.email;
+---------+-----------+----------------------+-------+
| imie    | nazwisko  | email                | listy |
+---------+-----------+----------------------+-------+
| Piotr   | Walczak   | walus@plus.pl        |     5 |
| Grzegorz | Marzec   | grzegorz@firma.com.pl |     3 |
| Maria   | Krzak     | krzak@polsska.pl     |    11 |
| Daria   | Kurak     | daria@firma.pl       |     1 |
| Wiesiek | Walc      | wiesiek@serwer.pl    |     9 |
| Karol   | Rutkowski | karol@opoka.pl       |     7 |
+---------+-----------+----------------------+-------+
6 rows in set (0.11 sec)
```

13. Usuwanie i aktualizacja istniejących rekordów.

Do usuwania rekordów stosujemy instrukcję DELETE:

DELETE FROM nazwa_tabeli WHERE które_rekordy_usunąć;

Jeżeli nie podamy klauzuli WHERE wykasowane zostaną wszystkie rekordy z wskazanej tabeli.
Klauzulą WHERE wybieramy rekordy, które chcemy skasować.

Usuńmy z tabeli dane wszystkie dane userów, którzy mieszkają w Krakowie.

```
mysql> delete from dane where miasto = 'Krakow';
Query OK, 2 rows affected (0.05 sec)
```

Usuńmy z tabeli skrzynka wszystkich, którzy mają więcej niż 10 e-maili.

```
mysql> delete from skrzynka where listy > 10;
Query OK, 1 row affected (0.06 sec)
```

Aby zmodyfikować rekordy, należy użyć instrukcji UPDATE.

UPDATE nazwa_tabeli SET które_kolumny_zmienić WHERE które_rekordy_zmienić;

Dodajmy najpierw nowego usera:

Imie: Darek
Nazwisko: Kaszana

```
mysql> insert dane(imie,nazwisko) values('Darek','Kaszana');
Query OK, 1 row affected (0.06 sec)
```

OK. A teraz załóżmy, że chcemy dopisać jeszcze jego miejsce zamieszkania i e-mail:

Miasto: Lódź
e-mail: kaszanka@sklep.kasza.pl

```
mysql> update dane set miasto = 'Lodz', email = 'kaszanka@sklep.kasza.pl'
    -> where imie = 'Darek' and nazwisko = 'Kaszana';
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

14. Zmiana struktury tabel.

Jeżeli nasze tabele są już stworzone, a my dowiadujemy się, że w tabeli dane powinniśmy byli umieścić jeszcze pole numer_id, które by nadawało unikatowy numer każdemu z użytkowników. W takiej sytuacji wcale nie musimy kasować tabeli, tworzyć jej jeszcze raz, teraz już z polem numer_id i wprowadzać do niej dane.
W takim przypadku możemy posłużyć się instrukcją ALTER TABLE, dzięki której możemy dodawać i usuwać kolumny, zmieniać typ kolumn itp.

No więc dodajmy kolumnę numer_id:

```
mysql> alter table dane add numer_id int unsigned not null auto_increment primary key;
Query OK, 5 rows affected (0.05 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

Teraz jeżeli przejrzymy dane w tabeli zauważymy, że kolumna numer_id została dodana i automatycznie zostały stworzone numery dla wszystkich naszych użytkowników.

```
mysql> select imie,nazwisko,email,numer_id from dane;
+---------+-----------+----------------------+----------+
| imie    | nazwisko  | email                | numer_id |
+---------+-----------+----------------------+----------+
| Piotr   | Walczak   | walus@plus.pl        |        1 |
| Grzegorz| Marzec    | grzegorz@firma.com.pl|        2 |
| Maria   | Krzak     | krzak@polsska.pl     |        3 |
| Daria   | Kurak     | daria@firma.pl       |        4 |
| Wiesiek | Walc      | wiesiek@serwer.pl    |        5 |
| Karol   | Rutkowski | karol@opoka.pl       |        6 |
+---------+-----------+----------------------+----------+
6 rows in set (0.00 sec)
```

15. Podsumowanie.

Kurs, przez który przebrnęliśmy to tylko podstawy, dla wszystkich tych, którzy zaczynają dopiero pracę z MySQL. Dla pogłębienia wiedzy odsyłam do artykułów zawartych w serwisie, a pogrupowanych w chwili obecnej na działy: podstawy, administracja, skrypty. Znajdziecie tam mam nadzieję bardziej dogłębne informacje na tematy poruszone w kursie. A wszelkie problemy, kłopoty, niejasności zgłaszajcie na FORUM, a my postaramy się je wyjaśniać.