Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

Bazy Danych

Ćwiczenie 2: Podstawy pracy z bazą MySQL

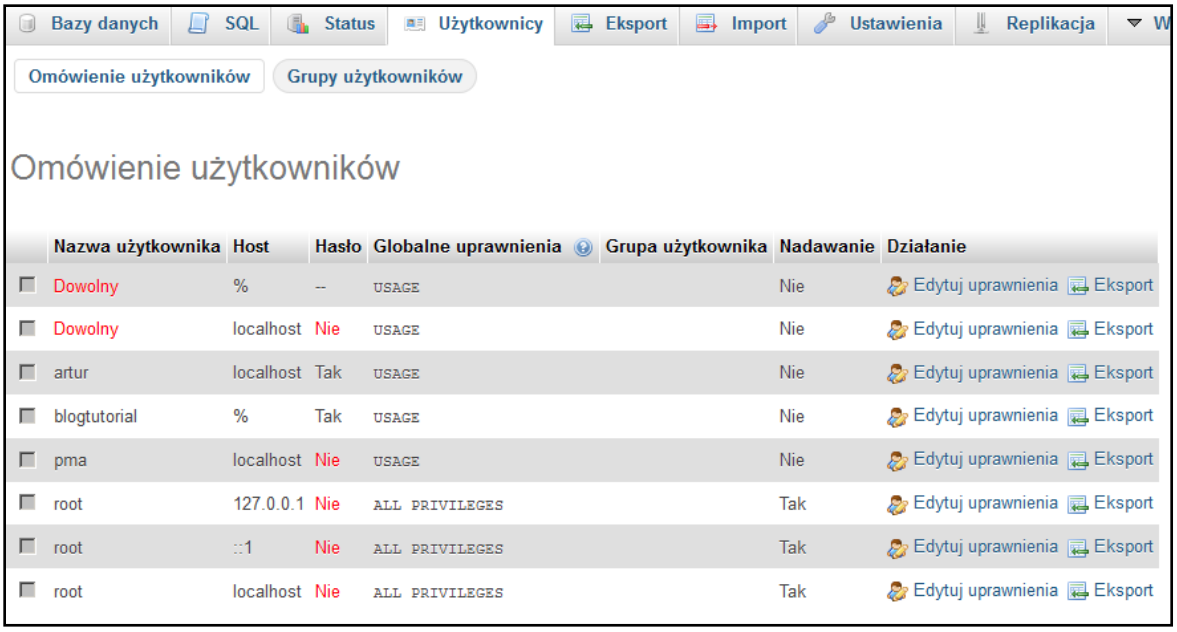
opracował: dr hab. inż. Artur Gramacki (a.gramacki@issi.uz.zgora.pl)

1. Uruchomić system XAMPP. Następnie poprawnie zalogować się na utworzone w poprzednik ćwiczeniu konto robocze (nazwa konta oraz bazy danych: lab) wykorzystując narzędzie phpMyAdmin oraz MySQL-Front (patrz poprzednie ćwiczenie). „Porozglądać" się w obu programach. Pewna biegłość w posługiwaniu się nimi będzie bardzo wskazana, żeby nie powiedzieć niezbędna, w czasie wykonywania wielu zadać podanych w kolejnych instrukcjach.
2. Celem ćwiczeń w tej instrukcji jest nabranie wprawy w pracy z bazą MySQL wykorzystując najbardziej podstawowy (i dostępny w każdej instalacji serwer MySQL) program konsoli tekstowej (nosi on oficjalną nazwę *MySQL monitor*). Jest to program o nazwie *mysql.exe* znajdujący się w katalogu ..*mysql**bin*. Gdy na ekranie otrzymamy komunikat podobny do poniższego oznacza to, że serwer MySQL nie działa i dalsze próby skomunikowania się z nim są bezcelowe:

   ```
   shell> mysql.exe
   ERROR 2003 (HY000): Can't connect to MySQL server on 'localhost' (10061)
   ```

   Należy upewnić się, że serwer MySQL został prawidłowo uruchomiony (patrz poprzednie ćwiczenie).

   Uwaga: na niebiesko zaznaczono polecenia (tu oraz w dalszej części instrukcji) wpisywane przez studenta w konsoli tekstowej. Symbol `shell>` zawsze będzie oznaczać znak zachęty w konsoli tekstowej systemu Windows a symbol `mysql>` znak zachęty w konsoli tekstowej MySQL-a.
3. Sprawdzić, wykorzystując program PhpMyAdmin, czy i jakie konta istnieją bez haseł. Kont takich w produkcyjnej instalacji bazy MySQL nie ma prawa być, gdyż stanowią potężną lukę bezpieczeństwa!



4. Potwierdzić możliwość zalogowania się na wszystkie „otwarte” konta, czyli przykładowo dla użytkownika *root*:

```
shell> mysql.exe
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 30
Server version: 5.6.26 MySQL Community Server (GPL)

Copyright (c) 2000, 2015, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select current_user();
+----------------+
| current_user() |
+----------------+
| @localhost     |
+----------------+
1 row in set (0.00 sec)

mysql>
```

5. Usunąć wszystkie „otwarte” konta a kontom, które nie posiadają założonych haseł utworzyć dowolne hasła.

6. Spróbować zalogować się na użytkownika *root* bez podawania hasła, z podaniem złego hasła i w końcu podając poprawne hasło, czyli:

```
shell> mysql -u root
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password:
NO)
```

```
shell> mysql -u root -pniepoprawnehaslo
Warning: Using a password on the command line interface can be insecure.
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password:
YES)

shell> mysql -u root -p
Enter password: **************
ERROR 1045 (28000): Access denied for user 'root'@'localhost' (using password:
YES)

shell> mysql -u root -p
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql>

shell> mysql -u root -proot
Warning: Using a password on the command line interface can be insecure.
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql>
```

Zrozum dokładnie różnice pomiędzy poszczególnymi wywołaniami programu *MySQL monitor*. **Zwróć uwagę, że po przełączniku `-p` nie ma spacji! Zapominanie o tym jest nagminnym błędem początkujących użytkowników serwera MySQL!**

7. Użyć bardziej rozwlekłej formy parametrów programu *MySQL monitor*, czyli,

```
shell>mysql --host=localhost --user=root --password --database=mysql --port=3306
```

Wyjaśnić, czy i kiedy taka długa forma ma praktyczne zastosowanie.

8. Połączyć się do serwera MySQL z jednoczesnym wyborem bieżącej bazy danych

```
shell> mysql -u root -p mysql
Enter password: ****
Welcome to the MySQL monitor.  Commands end with ; or \g.
...
mysql>
```

Zwróćmy uwagę, że teraz **pojawiła się spacja po przełączniku `-p`.** Tym razem oznacza to, że zamierzamy, po poprawnej autoryzacji, wybrać jako bieżącą bazę danych o nazwie *mysql*. Można też użyć przełącznika `-D`, aby zapis był bardziej wyrazisty, czyli:

```
shell> mysql -u root -p -D mysql
```

9. Będąc już podłączonym do serwera MySQL możemy wydać kilka poleceń, które po pierwsze upewnią nas, że całość działa poprawnie a po drugie pozwolą nieco dokładniej poznać system MySQL. Postaraj się zrozumieć co wykonują poszczególne polecenia.

Zwróć uwagę, że każde z nich jest zakończone średnikiem (za wyjątkiem dwóch pierwszych, gdyż są to polecenia programu *MySQL monitor*). Wielkość liter nie ma znaczenia, jednak sugerujemy, aby wszystkie słowa kluczowe języka SQL oraz polecenia programu *MySQL monitor* wpisywać

dużymi literami a pozostałe małymi (lub też odwrotnie. Najważniejsze jest w sumie konsekwentne trzymanie się cały czas tej samej konwencji).

W ostatnim poleceniu widać znaki `->`. Program *MySQL monitor* wykona wpisane polecenie dopiero po napotkaniu średnika (średnik pełni rolę separatora instrukcji). Każde więc wciśnięcie klawisza *Enter* przenosi nas do linii poniżej i umożliwia kontynuowanie wpisywania polecenia. Znaki `->` wyświetlane są automatycznie i mają one przypominać nam, że wpisywanie polecenie jeszcze się nie zakończyło. Taka funkcjonalność przydaje się wówczas, gdy polecenia są długie i nie mieszczą się w jednej linii.

```
mysql> help
mysql> status
mysql> show databases;
mysql> SELECT database();
mysql> use mysql;
mysql> show tables;
mysql> SELECT current_user();
mysql> SELECT user, host, password FROM user;
mysql> SELECT current_date();
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
mysql> SELECT
    -> user()
    -> ,
    -> current_date;
```

10. Jak już wspomniano wcześniej większość ćwiczeń wykonywana będzie na lokalnych instalacjach MySQL-a, gdzie będziesz posiadać pełne uprawnienia do bazy danych (będziesz znał hasło użytkownika *root*). Jednak na początku poznawania MySQL-a zdecydowanie bezpieczniej będzie pracować na koncie użytkownika, który będzie posiadał zdecydowanie mniej uprawnień i dzięki temu trudniej będzie przypadkowo uszkodzić serwer MySQL. Dlatego też zakładamy, że wszystkie dalsze ćwiczenia będą wykonywane po zalogowaniu się na konto użytkownika *lab*, hasło *lab*. Podczas ćwiczeń będziemy używali bazy o nazwie *lab*[1]. Gdy baza i użytkownik nie istnieją trzeba je utworzyć. Jak to zrobić pokazano niżej.

    Rozpoczynając wykonywanie dalszych ćwiczeń powinniśmy więc albo od razu podczas łączenia się do bazy MySQL wybrać bazę *lab* (przypominamy, że po przełączniku `-p` jest spacja. Gdyby jej nie było, to fraza *lab* zostałaby potraktowana jako hasło):

    ```
    shell> mysql -u lab -p lab
    Enter password: ****
    ...
    mysql>
    ```

    Bazę danych można wybrać też zaraz po udanym logowaniu (polecenie `use lab`):

[1] Uwaga: w systemie MySQL użytkownik oraz baza danych to dwie różne rzeczy. Zwykle administrator tworząc nowego użytkownika, od razu tworzy dla niego nową bazę danych (użytkownik ma pełne uprawnienia do pracy z tą bazą) o takiej samej nazwie. W ogólności jednak nazwy te nie muszą być takie same.
Z kontekstu wypowiedzi należy też wywnioskować czy mówiąc o bazie danych chodzi o system MySQL, czy też bazę danych w rozumieniu obiektu w systemie MySQL.

```
shell> mysql -u lab -p
Enter password: ****
...
mysql> use lab
Database changed
```

Aby upewnić się, jaki użytkownik jest zalogowany oraz do jakiej bazy jest on podłączony wykonaj dwa poniższe polecenia. Powinieneś na ekranie zobaczyć taki wynik:

```
mysql> SELECT DATABASE();
+------------+
| database() |
+------------+
| lab        |
+------------+
1 row in set (0.00 sec)
```

oraz

```
mysql> SELECT CURRENT_USER();
+----------------+
| current_user() |
+----------------+
| lab@localhost  |
+----------------+
1 row in set (0.00 sec)
```

Użytkownik *lab* ma taki zestaw uprawnień, który daje mu nieograniczony dostęp tylko do obiektów (głównie chodzi o tabele) w bazie *lab*. Próba przełączenia się na inną bazę danych zakończy się niepowodzeniem:

```
mysql> use mysql
ERROR 1044 (42000): Access denied for user 'lab'@'localhost' to database 'mysql'
```

11. W tym miejscu pokażemy jak utworzyć nową bazę *lab* oraz nowego użytkownika *lab* z hasłem *lab*. Polecenia te podajemy bez szczegółowego ich objaśniania. Czynimy tak, abyś mógł na swoim domowym komputerze wykonać analogiczne ćwiczenia jak te powyżej. Tworzenie oraz kasowanie baz danych oraz użytkowników to czynność zarezerwowana *wyłącznie* dla administratora MySQL-a. Trzeba to robić *bardzo ostrożnie*, gdyż bardzo łatwo jest spowodować w bazie danych ogromne spustoszenia! Aby wykonać poniższe ćwiczenia musisz zalogować się na użytkownika *root*, czyli:

```
shell> mysql -h localhost -u root -p
```

Dla pewności najpierw wydamy polecenia kasujące istniejącą bazę *lab* oraz użytkownika *lab* (gdyby okazało się, że w Twojej instalacji MySQL-a taka baza i taki użytkownik już istnieją). Zauważmy, że nazwa użytkownika oraz nazwa komputera (*lab*, *localhost* oraz *%*) są umieszczone w apostrofach. Apostrofy są konieczne w zasadzie tylko wtedy, gdy nazwy zawierają znaki specjalne, które są w szczególny sposób traktowane przez system operacyjny. W naszym przypadku można by apostrofy pominąć, jednak ich obecność w niczym nie przeszkadza.

```
mysql> DROP DATABASE lab;
```

```
Query OK, 0 rows affected (0.00 sec)

mysql> DROP USER 'lab'@'localhost';
Query OK, 0 rows affected (0.01 sec)

mysql> DROP USER 'lab'@'%';
Query OK, 0 rows affected (0.00 sec)

mysql> CREATE DATABASE lab;
Query OK, 1 row affected (0.03 sec)
```

Gdybyś w czasie wykonywania powyższych poleceń zobaczył poniższe komunikaty o błędach, będzie to oznaczało, że baza danych *lab* oraz użytkownik *lab* jeszcze nie istnieją w systemie (więc operacja kasowanie czegoś nieistniejącego nie może się udać). Z taką sytuacją spotkasz się, gdy polecenia uruchamiasz po raz pierwszy:

```
mysql> DROP DATABASE lab;
ERROR 1008 (HY000): Can't drop database 'lab'; database doesn't exist

mysql> DROP USER 'lab'@'localhost';
ERROR 1396 (HY000): Operation DROP USER failed for 'lab'@'localhost'

mysql> DROP USER 'lab'@'%';
ERROR 1396 (HY000): Operation DROP USER failed for 'lab'@'%'
```

Teraz możemy wpisać polecenia tworzące nową bazę danych oraz konto użytkownika. Uwaga: w zasadzie musimy utworzyć dwa konta. Jedno (`'lab'@'localhost'` będzie używane *tylko* do łączenia się lokalnego komputera. Drugie (`'lab'@'\%'`) można używać do łączenia się z dowolnego komputera. Nie wnikając na razie w szczegóły zapamiętaj, że musisz utworzyć oba konta.

```
mysql> GRANT ALL PRIVILEGES ON lab.* TO 'lab'@'localhost'
    -> IDENTIFIED BY 'lab';
Query OK, 0 rows affected (0.00 sec)

mysql> GRANT ALL PRIVILEGES ON lab.* TO 'lab'@'%'
    -> IDENTIFIED BY 'lab';
Query OK, 0 rows affected (0.00 sec)
```

Po utworzeniu kont warto sprawdzić, czy wszystko jest w porządku. Wydajemy wiec polecenia „sprawdzające”. Najpierw musimy jednak przełączyć na bazę „systemową” (polecenie `use mysql`). Powinniśmy zobaczyć wynik taki jak poniżej:

```
mysql> use mysql;
Database changed
mysql> SELECT host, user, password FROM user WHERE user LIKE 'lab';
+-----------+------+-------------------------------------------+
| host      | user | password                                  |
+-----------+------+-------------------------------------------+
| %         | lab  | *014CCBA08201296BAB648CAD12A48F7C93D7913D |
| localhost | lab  | *014CCBA08201296BAB648CAD12A48F7C93D7913D |
+-----------+------+-------------------------------------------+
2 rows in set (0.00 sec)
```

oraz

```
mysql> SELECT host, db, user FROM db WHERE db LIKE 'lab';
+-----------+------+------+
| host      | db   | user |
+-----------+------+------+
| %         | lab  | lab  |
| localhost | lab  | lab  |
+-----------+------+------+
2 rows in set (0.00 sec)
```

Aby za każdym razem nie wpisywać mozolnie dość długich poleceń można wszystkie je zapisać w dowolnym pliku tekstowym i uruchomić w tzw. trybie wsadowym. Niech więc nasz plik wygląda jak poniżej:

```
DROP DATABASE lab;
DROP USER 'lab'@'localhost';
DROP USER 'lab'@'%';
CREATE DATABASE lab;
GRANT ALL PRIVILEGES ON lab.* TO 'lab'@'localhost' IDENTIFIED BY 'lab';
GRANT ALL PRIVILEGES ON lab.* TO 'lab'@'%' IDENTIFIED BY 'lab';
USE mysql;
SELECT host, user, password FROM user WHERE user LIKE 'lab';
SELECT host, db, user FROM db WHERE db LIKE 'lab';
```

Następnie z poziomu programu klienckiego *MySQL monitor* wydajemy polecenie `source` (lub jego skróconą wersję `\.` – ukośnik oraz kropka), które odczyta ten pliku i wykona zapisane tam polecenia (ścieżkę dostępu oraz nazwę pliku musisz sam ustalić). Gdy wszystkie polecenia wykonają się poprawnie, na ekranie zobaczysz wynik podobny do poniższego:

```
mysql> source c:\temp\mysql_01.sql
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 1 row affected (0.00 sec)

Query OK, 0 rows affected (0.01 sec)

Query OK, 0 rows affected (0.00 sec)

Database changed
+-----------+------+-------------------------------------------+
| host      | user | password                                  |
+-----------+------+-------------------------------------------+
| %         | lab  | *014CCBA08201296BAB648CAD12A48F7C93D7913D |
| localhost | lab  | *014CCBA08201296BAB648CAD12A48F7C93D7913D |
+-----------+------+-------------------------------------------+
2 rows in set (0.01 sec)

+-----------+-----+------+
| host      | db  | user |
+-----------+-----+------+
```

```
| %         | lab | lab   |
| localhost | lab | lab   |
+-----------+-----+-------+
2 rows in set (0.00 sec)
```

12. Hasła użytkowników oraz adekwatny do wykonywanych zadań zestaw uprawnień są podstawą bezpiecznej pracy z systemem MySQL (oraz z każdą inną bazą danych). W ramach bieżącej instrukcji nie będziemy poznawali zagadnień (dość trudnych) dotyczących systemu uprawnień w MySQL-u. Poznamy natomiast polecenia służące do zmiany hasła użytkownika.

    Na początek zobaczmy na jakiego użytkownika jesteśmy zalogowani:

```
mysql> SELECT USER();
+----------------+
| user()         |
+----------------+
| lab@localhost  |
+----------------+
1 row in set (0.00 sec)
```

    Następnie zmieńmy hasło temu użytkownikowi:

```
mysql> SET PASSWORD = PASSWORD('nowe-haslo');
Query OK, 0 rows affected (0.00 sec)
```

    lub też używając nieco ,,nadmiarowej'' składni:

```
mysql> SET PASSWORD FOR 'lab'@'localhost'= PASSWORD('nowe-haslo');
Query OK, 0 rows affected (0.00 sec)
```

    Od tego momentu logując się do bazy MySQL na konto *lab* musimy podawać hasło *nowe-haslo*. Zwykły użytkownik (nie administrator) może zmienić hasło tylko sobie. Administrator może zmieniać natomiast hasła dowolnych użytkowników. Robi to w następujący sposób:

```
mysql> SET PASSWORD FOR 'lab'@'localhost'= PASSWORD('nowe-haslo');
Query OK, 0 rows affected (0.00 sec)

mysql> SET PASSWORD FOR 'lab'@'%'= PASSWORD('mysql');
Query OK, 0 rows affected (0.00 sec)
```

    Jeżeli zwykły użytkownik (nie administrator) będzie chciał zmienić nie swoje hasło otrzyma następujący komunikat o błędzie:

```
mysql> SET PASSWORD FOR 'root'@'localhost'= PASSWORD('nowe-haslo-root');
ERROR 1044 (42000): Access denied for user 'lab'@'localhost' to database 'mysql
```