Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

Bazy Danych

Ćwiczenie 16: System przywilejów oraz zarządzanie użytkownikami

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

1. Uwagi wstępne

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.

Poniżej podano tylko część wiadomości niezbędnych do *świadomego* wykonania zamieszczonych w następnym rozdziale poleceń. Resztę musisz samodzielnie doczytać w dostępnej literaturze. W szczególności chodzi tutaj o składnię poleceń `GRANT`, `REVOKE` oraz `SET PASSWORD`.

2. Ogólna zasada działania systemu przywilejów

System przywilejów w MySQL (jak i w każdym innym systemie zarządzania bazami danych) gwarantuje, że każdy użytkownik może wykonywać tylko te operacje, na które mu zezwolił administrator. W MySQL przyjęto zasadę, że tożsamość każdego użytkownika łączącego się do serwera jest ustalana wg:

- komputera, z którego nawiązano połączenie,
- podanej nazwy użytkownika.

Postąpiono tak, gdyż serwer MySQL od początku projektowany był do pracy w środowisku internetowym. Uwzględnianie nazwy komputera jest więc konieczne, gdyż trudno zakładać, że dana nazwa użytkownika będzie unikalna w całym internecie. Przykładowo użytkownik *artur* pracujący na komputerze www.*komputer1.pl* nie musi być tą samą osobą, co użytkownik *artur* pracujący na komputerze *www.komputer2.pl*.

Kontrola dostępu w MySQL składa się z dwóch etapów:

- **Etap 1**: serwer MySQL sprawdza, czy użytkownikowi w ogóle wolno się połączyć,
- **Etap 2**: jeżeli użytkownik może się połączyć, serwer kontroluje każde wydane przez niego plecenie, aby sprawdzić, czy użytkownik ma wystarczające przywileje, aby je wykonać.

Serwer przechowuje przywileje w tzw. *tabelach przywilejów* (ang. *grant tables*) w bazie *mysql*. Do tej bazy danych dostęp ma zwykle tylko administrator i w związku z tym tylko on może nadawać i odbierać uprawnienia. Serwer wczytuje zawartość tych tabel do pamięci, kiedy się uruchamia, i potem korzysta z ich zawartości do podejmowania decyzji o zezwoleniu lub zabronieniu wykonywania określonych czynności. Tabele przywilejów to:

- `user`,
- `db`,
- `host` (w praktyce dość rzadko wykorzystywana),
- `tables_priv`,
- `columns_priv`.

Zawartością tabel przywilejów można manipulować bezpośrednio. Są to normalne tabele, do których mamy dostęp za pomocą poleceń takich jak `SELECT` czy też `UPDATE`. Jednak wygodniej i bezpieczniej robić to za pomocą poleceń `GRANT` oraz `REVOKE`.

Strukturę tabel przywilejów łatwo jest poznać wydając polecenie `DESC`. Tabela *user* wygląda następująco:

```
mysql> desc user;
+------------------------+---------------------------------+------+-----+---------+-------+
| Field                  | Type                            | Null | Key | Default | Extra |
+------------------------+---------------------------------+------+-----+---------+-------+
| Host                   | char(60)                        | NO   | PRI |         |       |
| User                   | char(16)                        | NO   | PRI |         |       |
| Password               | char(41)                        | NO   |     |         |       |
| Select_priv            | enum('N','Y')                   | NO   |     | N       |       |
| Insert_priv            | enum('N','Y')                   | NO   |     | N       |       |
| Update_priv            | enum('N','Y')                   | NO   |     | N       |       |
| Delete_priv            | enum('N','Y')                   | NO   |     | N       |       |
| Create_priv            | enum('N','Y')                   | NO   |     | N       |       |
| Drop_priv              | enum('N','Y')                   | NO   |     | N       |       |
| Reload_priv            | enum('N','Y')                   | NO   |     | N       |       |
| Shutdown_priv          | enum('N','Y')                   | NO   |     | N       |       |
| Process_priv           | enum('N','Y')                   | NO   |     | N       |       |
| File_priv              | enum('N','Y')                   | NO   |     | N       |       |
| Grant_priv             | enum('N','Y')                   | NO   |     | N       |       |
| References_priv        | enum('N','Y')                   | NO   |     | N       |       |
| Index_priv             | enum('N','Y')                   | NO   |     | N       |       |
| Alter_priv             | enum('N','Y')                   | NO   |     | N       |       |
| Show_db_priv           | enum('N','Y')                   | NO   |     | N       |       |
| Super_priv             | enum('N','Y')                   | NO   |     | N       |       |
| Create_tmp_table_priv  | enum('N','Y')                   | NO   |     | N       |       |
| Lock_tables_priv       | enum('N','Y')                   | NO   |     | N       |       |
| Execute_priv           | enum('N','Y')                   | NO   |     | N       |       |
| Repl_slave_priv        | enum('N','Y')                   | NO   |     | N       |       |
| Repl_client_priv       | enum('N','Y')                   | NO   |     | N       |       |
| Create_view_priv       | enum('N','Y')                   | NO   |     | N       |       |
| Show_view_priv         | enum('N','Y')                   | NO   |     | N       |       |
| Create_routine_priv    | enum('N','Y')                   | NO   |     | N       |       |
| Alter_routine_priv     | enum('N','Y')                   | NO   |     | N       |       |
| Create_user_priv       | enum('N','Y')                   | NO   |     | N       |       |
| Event_priv             | enum('N','Y')                   | NO   |     | N       |       |
| Trigger_priv           | enum('N','Y')                   | NO   |     | N       |       |
| Create_tablespace_priv | enum('N','Y')                   | NO   |     | N       |       |
| ssl_type               | enum('','ANY','X509','SPECIFIED') | NO   |     |         |       |
| ssl_cipher             | blob                            | NO   |     | NULL    |       |
| x509_issuer            | blob                            | NO   |     | NULL    |       |
| x509_subject           | blob                            | NO   |     | NULL    |       |
| max_questions          | int(11) unsigned                | NO   |     | 0       |       |
| max_updates            | int(11) unsigned                | NO   |     | 0       |       |
| max_connections        | int(11) unsigned                | NO   |     | 0       |       |
| max_user_connections   | int(11) unsigned                | NO   |     | 0       |       |
| plugin                 | char(64)                        | YES  |     |         |       |
| authentication_string  | text                            | YES  |     | NULL    |       |
| password_expired       | enum('N','Y')                   | NO   |     | N       |       |
+------------------------+---------------------------------+------+-----+---------+-------+
43 rows in set (0.16 sec)
```

Kolumny *Host*, U*ser* oraz *Password* noszą nazwę *kolumn zasięgu* a pozostałe *kolumn przywilejów*.

Kolumny zasięgu, jak sama nazwa wskazuje, określają zasięg każdego wpisu. Na przykład wpis w tabeli *user* z wartościami *Host* oraz *User* równymi *artur* oraz *komputer1.pl* będzie używany do uwierzytelnienia połączeń nawiązywanych z komputera o podanej nazwie oraz przez klienta o nazwie *artur*.

Kolumny przywilejów określają przywileje przyznane przez dany wpis, to znaczy dozwolone operacje, które może wykonywać klient (np. prawo do kasowania zawartości określonej tabeli).

3. Weryfikacja połączenia

W pierwszym etapie, gdy następuje weryfikacja połączenia, sprawdzane są wpisy w tabeli *user* (kolumny *Host*, U*ser* oraz *Password*). Serwer akceptuje połączenie tylko wtedy, gdy kolumny *Host* oraz *User* w którymś z rekordów w tabeli *user* pasują do nazwy komputera i nazwy użytkownika a klient poda hasło określone w tym rekordzie. Wartości w kolumnach zasięgu tabeli *user* mogą przybierać następujące wartości:

- w kolumnie *Host* można podać dokładną nazwę lub numer IP komputera (np. *komputer1.pl* lub *192.168.21.130*) lub też użyć symboli wieloznacznych (np. *%.pl*}. Czy wiesz, co będzie oznaczał ten wpis?. Nazwa *localhost* wskazuje na komputer lokalny,
- w kolumnie *User* symbole wieloznaczne są niedozwolone, ale można podać wartość pustą, która pasuje do każdej nazwy. Jeżeli wpis w tabeli *User* dopasowany do przychodzącego połączenia zawiera pustą nazwę użytkownika, użytkownika uznaje się za anonimowego, pozbawionego nazwy, a nie za użytkownika o nazwie podanej przez klienta. Oznacza to, że do dalszej kontroli dostępu przez cały czas trwania połączenia używa się pustej nazwy użytkownika,
- kolumna *Password* też może być pusta. Oznacza to, że klient może połączyć się bez podawania hasła (oczywiście w praktyce należy unikać takich „otwartych furtek").

Z powyższego wynika, że w tabeli *user* może istnieć wiele wpisów, które „pasują" do danego klienta (bo można wpisywać symbole wieloznaczne). Serwer MySQL musi więc za każdym razem ustalić, którego z nich powinien użyć. Robi to w następujący sposób:

- kiedy serwer wczytuje tabelę *user* do pamięci *sortuje* wpisy,
- kiedy klient próbuje nawiązać połączenie, serwer przegląda wpisy w *posortowanej* kolejności,
- serwer używa *pierwszego* wpisu, który pasuje do nazwy komputera i nazwy użytkownika.

Przykładowo niech tabela *user* wygląda tak:

```
+-----------+------+
| Host      | User |
+-----------+------+
| %         | root |
| %         | lab  |
| localhost | root |
| localhost |      |
+-----------+------+
```

Kiedy serwer wczytuje tą tabelę porządkuje wpisy w taki sposób, że *najbardziej specyficzne* wartości kolumny *Host* trafiają na początek listy. Najbardziej specyficzne są dosłowne nazwy komputerów i adresy IP a najmniej specyficzny wpis to *%*, który oznacza „dowolny komputer". Następnie wpisy z taką samą wartością *Host* są porządkowane wg. najbardziej specyficznej wartości w kolumnie *User* (pusta wartość *User* oznacza „dowolnego użytkownika" i jest najmniej specyficzna).

Gdy więc klient próbuje się połączyć, serwer przegląda posortowaną listę i używa pierwszego pasującego wpisu. Powyższa tabela po posortowaniu wygląda więc następująco:

```
+-----------+------+
| Host      | User |
+-----------+------+
| localhost | root |
| localhost |      |
| %         | root |
| %         | ulab |
+-----------+------+
```

Zauważmy, że powyższy mechanizm autoryzacji może być dla początkującego użytkownika nieco mylący. Parę samodzielnie wykonanych ćwiczeń (patrz dalej) powinno wyjaśnić istotę zagadnienia.

Pozostałe kolumny w tabeli *user* (tzw. *kolumny przywilejów*) określają przywileje przyznane przez dany wpis (np. prawo do kasowania tabel).

4. Weryfikacja żądań

Tabela *user*

Po nawiązaniu połączenia serwer przechodzi do drugiego etapu kontroli dostępu. Sprawdza wpisy w pozostałych kolumnach w tabeli *user*, gdzie określone są przywileje na poziomie globalnym (obowiązujące dla każdej używanej bazy danych). Przykładowo, gdy w tabeli *user* jakiś użytkownik ma przyznane uprawnienie *Delete_priv*, może on usuwać rekordy z każdej tabeli w każdej bazie danych! Generalna zasada jest więc taka, aby przywileje w tej tabeli nadawać wyłącznie administratorom i nikomu innemu! Zwykli użytkownicy powinni mieć w tabeli *user* wszystkie wpisy ustawione na *N*. Fragment tabeli *user* pokazano niżej:

```
+-----------+------+-------------+-------------+-------------+-------------+-------------+-----------+----
| Host      | User | Select_priv | Insert_priv | Update_priv | Delete_priv | Create_priv | Drop_priv | ...
+-----------+------+-------------+-------------+-------------+-------------+-------------+-----------+-...
| localhost | root | Y           | Y           | Y           | Y           | Y           | Y         | ...
| %         | root | Y           | Y           | Y           | Y           | Y           | Y         | ...
| localhost | ulab | N           | N           | N           | N           | N           | N         | ...
| %         | ulab | N           | N           | N           | N           | N           | N         | ...
+-----------+------+-------------+-------------+-------------+-------------+-------------+-----------+----
```

Tabela *db*

Budowa tabeli *db* jest następująca:

```
mysql> desc db;
+-----------------------+---------------+------+-----+---------+-------+
| Field                 | Type          | Null | Key | Default | Extra |
+-----------------------+---------------+------+-----+---------+-------+
| Host                  | char(60)      | NO   | PRI |         |       |
```

```
| Db                    | char(64)       | NO   | PRI |         |       |
| User                  | char(16)       | NO   | PRI |         |       |
| Select_priv           | enum('N','Y') | NO   |     | N       |       |
| Insert_priv           | enum('N','Y') | NO   |     | N       |       |
| Update_priv           | enum('N','Y') | NO   |     | N       |       |
| Delete_priv           | enum('N','Y') | NO   |     | N       |       |
| Create_priv           | enum('N','Y') | NO   |     | N       |       |
| Drop_priv             | enum('N','Y') | NO   |     | N       |       |
| Grant_priv            | enum('N','Y') | NO   |     | N       |       |
| References_priv       | enum('N','Y') | NO   |     | N       |       |
| Index_priv            | enum('N','Y') | NO   |     | N       |       |
| Alter_priv            | enum('N','Y') | NO   |     | N       |       |
| Create_tmp_table_priv | enum('N','Y') | NO   |     | N       |       |
| Lock_tables_priv      | enum('N','Y') | NO   |     | N       |       |
| Create_view_priv      | enum('N','Y') | NO   |     | N       |       |
| Show_view_priv        | enum('N','Y') | NO   |     | N       |       |
| Create_routine_priv   | enum('N','Y') | NO   |     | N       |       |
| Alter_routine_priv    | enum('N','Y') | NO   |     | N       |       |
| Execute_priv          | enum('N','Y') | NO   |     | N       |       |
| Event_priv            | enum('N','Y') | NO   |     | N       |       |
| Trigger_priv          | enum('N','Y') | NO   |     | N       |       |
+-----------------------+---------------+------+-----+---------+-------+
22 rows in set (0.16 sec)
```

W tabeli *db* wpisane są przywileje specyficzne dla bazy danych. Zasada działania tej tabeli (dopuszczalne wpisy, sortowanie) jest analogiczna jak dla tabeli *user*. Oprócz kolumn *Host* oraz *user* pojawia się tutaj dodatkowo kolumna *Db*. Brak jest natomiast kolumny *Password*. Kolumny *Host*, *User* oraz *Db* noszą nazwę *kolumn zasięgu* a pozostałe *kolumn przywilejów* (podobnie jak w tabeli *user*). Wartości w kolumnach zasięgu tabeli *db* mogą przybierać następujące wartości:

- w kolumnie *Host* można podać dokładną nazwę lub numer IP komputera lub też użyć symboli wieloznacznych. Nazwa *localhost* wskazuje na komputer lokalny. Wartość *%* lub wartość pusta oznacza „dowolny komputer”,
- w kolumnie *Db* też można używać symboli wieloznacznych. Wartość *%* lub wartość pusta oznacza „dowolna baza danych”,
- wartość pusta w kolumnie *User* oznacza użytkownika anonimowego.

Serwer wczytuje i sortuje tabelę *db* w tym samym czasie co tabelę *user*. Sortowanie odbywa się podług kolumn *Host*, *Db* oraz *User*. Najbardziej specyficzne wartości trafiają na początek listy. Kiedy serwer szuka pasujących wpisów, używa pierwszego znalezionego wpisu.

Tabele *tables_priv* oraz *columns_priv*

Budowa tabel *tables_priv* oraz *columns_priv* jest następująca (opis struktury jest długi, więc pokazano go w dwóch fragmentach):

```
mysql> desc tables_priv;
+-------------+-----------------------------------------------------------------------------
| Field       | Type
+-------------+-----------------------------------------------------------------------------
| Host        | char(60)
| Db          | char(64)
| User        | char(16)
| Table_name  | char(64)
| Grantor     | char(77)
| Timestamp   | timestamp
```

```
| Table_priv  | set('Select','Insert','Update','Delete','Create','Drop','Grant','References
| Column_priv | set('Select','Insert','Update','References')
+-------------+-------------------------------------------------------------------------------
8 rows in set (0.16 sec)


---------------------+------+-----+-------------------+-----------------------------+
                     | Null | Key | Default           | Extra                       |
---------------------+------+-----+-------------------+-----------------------------+
                     | NO   | PRI |                   |                             |
                     | NO   | PRI |                   |                             |
                     | NO   | PRI |                   |                             |
                     | NO   | PRI |                   |                             |
                     | NO   | MUL |                   |                             |
                     | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
how view','Trigger') | NO   |     |                   |                             |
                     | NO   |     |                   |                             |
---------------------+------+-----+-------------------+-----------------------------+
```

oraz

```
mysql> desc columns_priv;
+-------------+-----------------------------------------+------+-----+-------------------+-----------------------------+
| Field       | Type                                    | Null | Key | Default           | Extra                       |
+-------------+-----------------------------------------+------+-----+-------------------+-----------------------------+
| Host        | char(60)                                | NO   | PRI |                   |                             |
| Db          | char(64)                                | NO   | PRI |                   |                             |
| User        | char(16)                                | NO   | PRI |                   |                             |
| Table_name  | char(64)                                | NO   | PRI |                   |                             |
| Column_name | char(64)                                | NO   | PRI |                   |                             |
| Timestamp   | timestamp                               | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
| Column_priv | set('Select','Insert','Update','References') | NO   |     |                   |                             |
+-------------+-----------------------------------------+------+-----+-------------------+-----------------------------+
7 rows in set (0.16 sec)
```

Tabele *tables_priv* oraz *columns_priv* zawierają przywileje specyficzne dla tabel i kolumn. W tych dwóch tabelach tylko w kolumnie *Host* mogą pojawić się symbole wieloznaczne. Sortowanie odbywa się podług kolumn *Host*, *Db* oraz *User*. Przypomina to sortowanie tabel *user* oraz *db* ale jest prostsze, bo tylko kolumna *Host* może zawierać symbole wieloznaczne.

Wartości w kolumnach zasięgu tabel *tables_priv* oraz *columns_priv* mogą przybierać następujące wartości:

- w kolumnie *Host* obu tabel można podać dokładną nazwę lub numer IP komputera lub też użyć symboli wieloznacznych. Nazwa *localhost* wskazuje na komputer lokalny. Wartość *%* lub wartość pusta oznacza „dowolny komputer”,
- w kolumnach *Db*, *Table_name* oraz *Column_name* nie mogą pojawić się symbole wieloznaczne ani wartośći puste.

Procedura określania przywilejów użytkownika

- w przypadku żądań wymagających przywilejów administracyjnych (np. `SHUTDOWN`) sprawdzane są wpisy w tabeli *user*,
- w przypadku żądań związanych z konkretną bazą danych (np. `INSERT`, `UPDATE`) sprawdzane są najpierw przywileje globalne w tabeli *user*. Jeżeli odpowiedni wpis zezwala na żądaną operację, serwer przyznaje dostęp. Jeżeli przywileje w tabeli *user* są niewystarczające, serwer sprawdza wpisy w tabeli *db* (a czasami też w tabeli *host* – jest ona jednak rzadko używana w praktyce, więc nie omawiamy jej),

- serwer szuka w tabeli *db* pasujących wartości kolumn *Host*, *Db* oraz *User*. Kolumny *Host* oraz *User* są dopasowywane do komputera i podanej nazwy użytkownika. Kolumna *Db* jest dopasowywana do bazy danych, z której chce skorzystać użytkownik. Gdy brak jest wpisu z pasującymi wartościami w kolumnach *Host* oraz *User*, serwer odmawia dostępu,
- po ustaleniu przywilejów specyficznych dla bazy danych, określonych w tabeli *db* (i ew. też *Host*), serwer dodaje je do globalnych przywilejów określonych w tabeli *user*. Jeżeli wynik pozwala na wykonanie żądanej operacji, serwer przyznaje dostęp. W przeciwnym wypadku serwer kolejno sprawdza przywileje użytkowników specyficzne dla tabel oraz dla kolumn w tabelach *tables_priv* oraz *columns_priv*, dodaje je do przywilejów użytkownika i na podstawie wyniku przyznaje dostęp lub odmawia dostępu.

Kiedy uwzględniane są zmiany przywilejów

Kiedy serwer MySQL uruchamia się, cała zawartość tabel przywilejów zostaje wczytana do pamięci i od tego momentu jest używana do kontroli dostępu.

Kiedy zmieni się zawartość tabel przywilejów rezydujących w pamięci, przywileje już połączonych klientów zmieniają się w następujący sposób:

- zmiany przywilejów dotyczących tabel i kolumn są uwzględniane przy następnym żądaniu klienta,
- zmiany przywilejów dotyczących baz danych są uwzględniane przy następnym użyciu instrukcji `USE`,
- zmiany przywilejów globalnych i haseł są uwzględniane przy następnym połączeniu klienta.

Jeżeli tabele przywilejów zostaną zmodyfikowane za pomocą instrukcji `GRANT`, `REVOKE` oraz `SET` *Password*, serwer zostanie poinformowany o zmianach natychmiast i automatycznie uaktualni zawartość tabel przywilejów znajdujących się w pamięci.

Jeżeli tabele przywilejów zostaną zmodyfikowane za pomocą instrukcji `INSERT`, `UPDATE` lub `DELETE` zmiany nie będą miały wpływu na sprawdzanie przywilejów, dopóki użytkownik nie uruchomi ponownie serwera albo nie nakaże mu ponownie wczytać tabele przywilejów poprzez wydanie polecenia `FLUSH PRIVILEGES` lub `mysqladmin flush-privileges` lub `mysqladmin reload`.

Oglądanie aktualnie przyznanych przywilejów

Oglądanie przyznanych przywilejów poprzez bezpośrednie zaglądanie do tabel systemowych jest dość niewygodne i na dodatek łatwo o pomyłkę. Dużo wygodniejsze jest posługiwanie się poleceniem `SHOW GRANTS`, które w przejrzysty sposób pokazuje aktualnie przyznane przywileje dla wybranego użytkownika, przykładowo:

```
mysql> SHOW GRANTS FOR  'lab'@'%';
+-----------------------------------------------------------------------------------------------------------+
| Grants for lab@%                                                                                          |
+-----------------------------------------------------------------------------------------------------------+
| GRANT USAGE ON *.* TO 'lab'@'%' IDENTIFIED BY PASSWORD '*014CCBA08201296BAB648CAD12A48F7C93D7913D' |
| GRANT ALL PRIVILEGES ON `lab`.* TO 'lab'@'%'                                                              |
+-----------------------------------------------------------------------------------------------------------+
2 rows in set (0.00 sec)
```

5. Polecenia do wykonania

Rozwiązania wszystkich poniższych zadań należy umieścić w *skrypcie* (pliku tekstowym). Aby skrypt można uruchamiać wielokrotnie pamiętaj o tym, że na początku skryptu muszą znaleźć się polecenia kasujące odpowiednie wpisy w tabelach przywilejów! W przeciwnym wypadku będą pojawiać się błędy naruszania kluczy głównych (patrz budowa tabel przywilejów).

Zakładamy, że ćwiczenia rozpoczynamy wykonywać, gdy w systemie istnieje *tylko* konto *root*. (jeden lub dwa wpisy w tabeli *user*. Czy potrafisz wyjaśnić dlaczego mówimy o jednym lub dwóch wpisach?). W tabelach *db*, *tables_priv* oraz *columns_priv* nie ma żadnych wpisów.

Zadanie 1

Załóżmy, że w tabeli *user* mamy następujące wpisy:

```
+-----------+------+
| Host      | User |
+-----------+------+
| %         | lab  |
| komp.com  |      |
+-----------+------+
```

Jaki użytkownik zostanie faktycznie uwierzytelniony, gdy połączenie zostanie nawiązywane przez użytkownika *lab* z komputera *komputer1.com*? Jak sprawdzić, jakiego konta użył serwer do uwierzytelnienia użytkownika? Wyciągnij praktyczne wnioski.

Zadanie 2

Sprawdzić, czy w lokalnej instalacji MySQL są jakieś konta, które pozwalają połączyć się do serwera bez podawania hasła lub też istnieją konta anonimowe.

Zadanie 3

Utworzyć 4 konta użytkowników o nazwach *user1*, *user2*, *user3*, *root2*.

Użytkownicy *user1* oraz *user2* nie powinni mieć nadanych żadnych uprawnień natomiast użytkownik *user3* powinien mieć nadane pełne prawa do bazy *db3* (baza o tej nazwie nie musi w tym momencie fizycznie istnieć).

Ostatnie konto (*root2*) ma mieć uprawnienia administratora (takie same jakie posiada użytkownik *root*).

Użyć poleceń `CREATE USER ...`, `GRANT USAGE ...` oraz `GRANT ALL PRIVILEGES ...`

Zadanie 4

Utworzyć dwie nowe bazy danych o nazwach *db1* oraz *db2*.

Zadanie 5

Nadać pełne prawa do bazy *db1* dla użytkownika *user1* oraz pełne prawa do bazy *db2* dla użytkownika *user2*.

Zadanie 6

Zmienić hasła użytkownikom user1 oraz *user2* na dowolne inne. Użyć raz polecenia `SET PASSWORD` a raz polecenia `GRANT USAGE`.

Zadanie 7

Odebrać użytkownikowi *user3* wszystkie uprawnienia do bazy *db3*.

Zadanie 8

Utworzyć 4 tabele (struktura tabel może być zupełnie dowolna). Dwie w bazie *db1* oraz dwie w bazie *db2*. Użytkownikowi *user1* nadać prawa *SELECT, INSERT, UPDATE, DELETE* do pierwszej tabeli użytkownika *user2* oraz prawo *SELECT* do jednej wybranej kolumny w drugiej tabeli użytkownika *user2*. Użytkownikowi *user2* nadać pełne prawa do wszystkich obiektów w bazie *db1*.

Zadanie 9

Za pomocą polecenia `SHOW GRANTS` wyświetlić uprawnienia wszystkich użytkowników.

Zadanie 10

Zapoznać się z aktualną zawartością tabel przywilejów (*user*, *db*, *tables_priv*, *columns_priv*). Postarać się dokładnie zrozumieć poszczególne wpisy. Skonfrontować zawartość tych tabel z wynikami zwracanymi przez polecenie `SHOW GRANTS`.

Rozwiązania

Do instrukcji dołączono rozwiązania do ćwiczeń z poprzedniego punktu. Celowo podajemy je wyłącznie w postaci plików graficznych, abyś nie mógł po prostu skopiować poleceń ☺. Postaraj się najpierw samodzielnie rozwiązać zadania, a gdy już naprawdę nic nie będzie Ci wychodziło, posłuż się poniższą „ściągawką”.