30.11.2014
Pytania teoretyczne do programowania w MySQL

Uwaga wstępna 1.
Odpowiedzi udzielaj w Twojej pracy semestralnej.
Przepisz pytania (teoretyczne i praktyczne) i udziel odpowiedzi. Na pytania praktyczne odpowiedzi udziel poprzez kopiowanie instrukcji.
Np.
mysql>show databases;
lub czytelne zrzuty ekranu.

Uwaga wstępna 2.
Możesz posiłkować się stroną z kursem w necie.

Zadania

Zadanie 1.
Co to jest i do czego służy mysql?

Zadanie 2.
Jakie czynności musisz wykonać, aby móc korzystać z mysql, (jaki program należy zainstalować i inne czynności)?

Zadanie 3.
Uruchom serwer mysql poprzez:
Sposób 1
Uruchamiamy krasnal i klikamy prawym na zielone pióro i wybieramy MySQL CMD
Sposób 2
Wchodzimy do folderu:
usr\mysql\bin i uruchamiamy plik mysql

Zadanie 4.
Na końcu instrukcji jest średnik **„ ; ”**

Wyświetl bazy danych znajdujące się dysku poprzez:
show databases;

Uwaga
Jeśli po napisaniu instrukcji otrzymasz
-> (strzałkę) to oznacza to, że nie wpisałeś średnika po zakończeniu instrukcji.

Zadanie 5.
1)Wyloguj się z mysql poprzez Quit;
2)Zaloguj się na root mysql.

Wejdź w DOS do folderu c:\usr\mysql\bin i wydaj komendę:
mysql –u root –p enter
podaj hasło krasnal

Możesz do uruchomienia serwera mysql użyć następującego BATa

```
c:
cd c:\
cd C:\usr\mysql\bin
mysql -u root -pkrasnal
pause
```

nagraj go pod nazwą np. uruch_sql.bat

Zadanie 6.
Uaktywnij bazę danych o nazwie mysql poprzez użycie instrukcji:
use nazwa bazy danych;

Zadanie 7.
Wyświetl tabele wchodzące w skład bazy danych uaktywnionej w pytaniu poprzednim:
show tables;

Zadanie 8.
Wyświetl strukturę tabeli **db** poprzez:
describe nazwa_tabeli;

Zadanie 9.
Wyświetl dane z tabeli **db** uaktywnionej w pytaniu poprzednim.
Select * from nazwa_tabeli;

W celu polepszenia wyświetlania wyświetl okienko DOS na pełen ekran poprzez Alt(lewy)+Enter

Algorytm gdy chcesz wyświetlić z bazy dane to:
1) Zaloguj się do mysql,
2) Uaktywnij bazę z, której chcesz wyświetlić dane: *use baza;*
3) Wyświetl dane z tabeli znajdującej się w tej bazie czyli
   select * from nazwa_tabeli;

Zadanie 10.
Temat: Założenie bazy danych oraz wpisanie danych do niej.

1)Przeczytaj o typach pól w mysql.

2)Sposoby automatyzacji pracy w Mysql

Z poziomu DOS:
Przy wykonywaniu złożonych poleceń mysql możesz polecenia (instrukcje mysql) **zapisać w pliku tekstowym** np.o nazwie poleceni10.txt, najlepiej aby plik ten znajdował się tam gdzie znajduje się mysql.exe wydać komendę w dos.

Uruchomienie w DOS pliku tekstowego poprzez wydanie komendy:
mysql –u root –p<polecenie10.txt enter i teraz hasło.

Z poziomu Mysql:
Mamy:
a) plik tekstowy pod nazwa np. z13_zak_trzy_litery nazwiska_ucznia.sql
np. z13_zak_bar.sql
b)nagrany do folderu bin (miejsce c:\usr\mysql\bin)
c)bedąc zalogowana w mysql (czyli masz kursor mysql>)
d)uruchom plik instrukcją:
source *z13_zak_bar.sql*

3)Plik tekstowy powinien zawierać:

- Kasowanie bazy danych, (gdy zakładałeś bazę i tabelę a coś było źle to jest już baza i ponowne założenie będzie powodować błędy)→drop database….;
- założenie bazy →create database……;
- aktywacja bazy →use …….;
- utworzenie tabeli →create table ……..;
- wpisanie danych do tabeli→ insert into…….
- wyjść z mysql poprzez instrukcje→ exit;

4)Załóż plik tekstowy posługując się informacjami poniżej:
a) drop database….;→kasowanie bazy
b)Załóż bazę danych o nazwie **nazwisko słuchacza** (bez polskich liter małymi literami) poprzez instrukcję **create database nazwa_bazy;**
c)aktywacja bazy →use …….;
d)utworzenie tabeli →create table ……..;
Utwórz tabelę o nazwie tabeli: **kontakty_nazwisko_słuchacza** np. kontakty_kowalski (bez polskich liter małymi literami).czterech polach:
id_ trzy_litery_nazwiska_ucznia → typ pola (int(10) unsigned NOT NULL auto_increment),
numer_trzy_litery_nazwiska_ucznia → typ pola (tinyint(4)),
nazwisko_trzy_litery_nazwiska_ucznia → typ pola (text),
waga_trzy_litery_nazwiska_ucznia → typ pola (int(11))
e)Wpisz do niej pięć rekordów rekordów z sensowymi danymi.
(Wzoruj się na przykładzie poniżej oraz przykłady w opisie teoretycznym).
wpisanie danych do tabeli→ insert into…….
f)wyświetlenie zawartości tabeli :
select * from nazwa_tabeli;

Przykład:
Uwaga1 napis **mysq**l w przykładzie poniżej to znak zachęty, przy pisaniu plikowego tekstowego możesz go pominąć.
Uwaga2 przykład poniżej to zakładanie tabeli. Konieczne będzie najpierw wydanie komend:

- Zakładanie bazy danych
- Aktywacja złożonej bazy i teraz tworzenie tabeli(patrz poniżej).

Przykład→założenie tabeli

```
mysql> CREATE TABLE  arkusz  (
   id  int(10) unsigned NOT NULL auto_increment,
   pole_a  varchar(10) NOT NULL,
   pole_b  int(1) NOT NULL,
   pole_c  int(1) NOT NULL,
   pole_d  int(1) NOT NULL,
   pole_e  int(1) NOT NULL,
   pole_f  int(1) NOT NULL,
   pole_g  varchar(3) NOT NULL,
  PRIMARY KEY  ( id )
);
```

Opis do instrukcji powyżej:

Atrybut **UNSIGNED** przy tworzeniu tabeli w bazie informuje bazę, iż będziemy przechowywać w niej tylko wartości dodatnie, czyli 0, 1, 2, 3, … itp.
NOT NULL, że pole nie może być puste.
Jak dodajesz dane do bazy, to w tym polu, któremu dałeś właściwość **AUTO_INCREMENT**, nie wprowadzasz żadnych danych, tylko pustą wartość NULL. Mimo tego, serwer MySQL i tak wprowadzi odpowiednią wartość: będzie to kolejny numerek w kolejności (np. przy pierwszym rekordzie da 1, potem 2 i tak dalej). W ten sposób można łatwo odróżnić każdy rekord, gdyż dostał swój unikalny identyfikator, który się nie powtórzy.
PRIMARY KEY - powoduje, że dane w kolumnie nie mogą sie powtarzać, służy do identyfikacji rekordu, jest klucz podstawowy.

Przykład→wpisanie danych

```
INSERT INTO uzytkownicy (imie, nazwisko, email, data_dodania) VALUES
('Bogdan', 'Przykładowski', 'b.example@domena.pl', NOW() );
```

Zadanie 11.
a)Zaloguj się na root'a
b)Wyświetl bazy danych znajdujące się dysku. →zrzut ekranu(z nazwiskiem)
c)Uaktywnij bazę, którą utworzyłeś w poleceniu poprzednim. →zrzut ekranu(z nazwiskiem).
d)Wyświetl tabele wchodzące w skład bazy danych, którą utworzyłeś w porzednim pytaniu. →zrzut ekranu(z nazwiskiem)
e)Wyświetl strukturę tabeli, którą utworzyłeś w porzednim pytaniu. →zrzut ekranu(z nazwiskiem)
f)Wyświetl dane z tabeli do, której wpisywałeś dane w pytaniu powyżej. →zrzut ekranu(z nazwiskiem)

Zadanie 12.
Temat: Wpisywanie danych tekstowych do bazy mysql.

- Utwórz nową bazę danych o nazwie *baza_nazwisko_ucznia* pod zbiór importowany z pliku tekstowego. Użyj instrukcji create.

- Utwórz użytkownika o nazwie *nazwisko_ucznia* (bez polskich liter, małymi literami) i haśle *plik_tekstowy*, który będzie miał wszystkie prawa do tabel tworzonych w bazie danych. (Wzoruj się na przykładzie poniżej).

Uwaga:
Założenie konta jest w mysql jest utoższamiane z nadanie uprawnień do bazy danych.

Przykład:
Założenie konta:
do bazy → arkusz_kalkulacyjny
nazwa użytkownika → arkusz
hasło → arkuszhaslo

grant all on arkusz_kalkulacyjny.* to arkusz@localhost identified by 'arkuszhaslo';

- Wyloguj się z konta administratora i zaloguję na nowo utworzone konto użytkownika o nazwie *nazwisko_ucznia* (bez polskich liter, małymi literami) i haśle *plik_tekstowy*.
(Wzoruj się na przykładzie poniżej).

Przykład:
mysql -u arkusz –parkuszhaslo arkusz_kalkulacyjny ENTER
będziesz proszony o podanie hasła.

Instrukcję logowania wpisz w DOS.

- Załóż tabelę w bazie o nazwie *baza_nazwisko_uczni.* Nazwa tabeli to tabela_nazwisko_ucznia.

Poniżej masz jeden wiersz bazy danych. Są to wyniki Totolotka:
-data zakładu,
-sześć wylosowanych liczb,
-gdzie padła wygrana,
-wielkość wygranej,
-numer kolektury.
Uwagi
a)Nazwy pól powinny zawierać trzy litery Twojego nazwiska.
b)Pamiętaj o polu identyfikatora(klucz)
c)Pamiętaj o odpowiednich typach pól.

1996-04-27;48;36;17;8;3;1;Poznań;1000000;6/18

- Dokonaj wpisania danych z pliku tekstowego otrzymanego od nauczyciela do tabeli tabela_nazwisko_ucznia. (Wzoruj się na przykładzie poniżej).

Przykład(nie zwiaany ztrescią zadania):
Ładowanie danych.
Załadowanie danych ze zbioru CSV, zbiór 101.csv został zapisany w katalogu /tmp.
Przykład jednego rekordu
"15cc","3","0","1","1","1","A4"
Jesteśmy cały czas w linii komend MySQL-a:

mysql> LOAD DATA INFILE '/tmp/101.csv' INTO TABLE `arkusz` FIELDS TERMINATED BY ',' ENCLOSED BY "" ESCAPED BY '\\' LINES TERMINATED BY '\n' (pole_a, pole_b, pole_c, pole_d, pole_e, pole_f, pole_g);

Zadanie 13.

Treść polecenia:
W pliku losowania.txt znajduje się zestawienie 513 losowań Dużego Lotka, w których wystąpiły najwyższe wygrane.
Każdy wiersz zawiera następujące informacje:

- Data (data losowania Dużego Lotka),
- L_1, L_2, L_3, L_4, L_5, L_6 (6 kolejno wylosowanych liczb z 49),
- Miejscowość (miejscowość, w której padła wygrana),
- Wygrana (wartość najwyższej wygranej),
- Kolektura (nr kolektury, w której padła wygrana),
- Dane w wierszach rozdzielone są znakiem tabulacji.

Przykład danych w pliku losowania.txt:

Data	L_1	L_2	L_3	L_4	L_5	L_6	Miejscowość	Wygrana	Kolektura
1996-04-27	48	36	17	8	3	1	Poznań	1000000	6/18
1997-03-08	41	40	32	21	10	8	Police	1060754,5	15/99
1997-03-19	43	35	31	18	8	7	Szczecin	1471617,8	15/51

Korzystając z danych zawartych w pliku losowania.txt wykonaj poniższe polecenia. Odpowiedzi do poniższych poleceń umieść w pliku tekstowym zadanie1.txt poprzedzając odpowiedź do danego polecenia jego oznaczeniem literowym.

a) Podaj, w której miejscowości, kiedy oraz w jakiej wysokości padła najwyższa wygrana.
b) Podaj zestawienie zawierające miasta, w których wygrana padła więcej niż dziesięć razy.
c) Która z liczb w podanych losowaniach wystąpiła najczęściej i ile razy?
d) Jaką kwotę wypłacono łącznie na najwyższe wygrane w latach 2000-2003?
e) Podaj zestawienie zawierające dwie kolumny: rok oraz liczbę wygranych w każdym roku.

Rozwiązanie:
1)Użyj Mysql→ uruchom krasnala→ uruchom mysql z logowaniem na konto root i hasłem krasnal. Wejdź do folderu c:\usr\mysql\bin i wydaj komendę:
mysql –u root –p enter
podaj hasło krasnal

Baza nazwa →baza_nazwisko słuchacza np. baza_baranski

załóż bazę poprzez
create database baza_nazwisko słuchacza np. baza_baranski;

2)Załóż strukturę bazę danych korzystając z pliku tekstowego (treść pliku tekstowego poniżej.

pola bazy→ nazwa z trzema literkami nazwiska słuchacza np. liczba_1_bar
Tabela→nazwa tabela_trzy_pierwsze_litery_nazwiska np. tabela_bar

Treść pliku tekstowego

```
use baza_baranski;
create table tabela_baranski(
id_bar int(10) unsigned NOT NULL auto_increment,
data_bar date,
liczba_1_bar float(2),
liczba_2_bar float(2),
liczba_3_bar float(2),
liczba_4_bar float(2),
liczba_5_bar float(2),
liczba_6_bar float(2),
miasto_bar text,
wygrana_bar float(9,1),
kolektura_bar text,
PRIMARY KEY (id_bar)
);
```

Nagraj plik tekstowy pod nazwa z13_zak_trzy_litery nazwiska_ucznia.sql
np. z13_zak_bar.sql do folderu bin (miejsce c:\usr\mysql\bin)

bedąc zalogowana w mysql uruchom plik instrukcją:
source *z13_zak_bar.sql*

uwaga:
musisz być zalogowany na serwerze sql oraz na koncie, czyli masz kursor mysql>

Wstaw plik z13_zak_bar.sql do treści pracy.

3)-Sprawdź, jakie są bazy→zrzut ekranu z nazwiskiem.
-Sprawdź, jakie są tabele w bazie, które wykonałeś przed chwilą→zrzut ekranu z nazwiskiem.
-Sprawdź, jakie są pola w tabeli z tego zadania→zrzut ekranu z nazwiskiem.

4)Wczytaj dane do tabeli

a)Wpisz plik tekstowy (zmień nazwę tabeli oraz nazwy pól tak aby nie było „bar" a twoje trzy litery).

Treść pliku tekstowego

```
use baranski;
LOAD DATA INFILE 'c:\\usr\\mysql\\bin\\losowania3.txt' INTO TABLE
tabela_baranski FIELDS TERMINATED BY '\t' ESCAPED BY '\\' LINES
TERMINATED BY '\n'
(data_bar,liczba_1_bar,liczba_2_bar,liczba_3_bar,liczba_4_bar,liczba_5_bar,liczba_
6_bar,miasto_bar,wygrana_bar,kolektura_bar);
```

b)Nagraj plik tekstowy pod nazwa z13_wcz_trzy_litery nazwiska_ucznia.sql
np. z13_wcz_bar.sql do folderu bin (miejsce c:\usr\mysql\bin)

c)uruchom plik instrukcją:

mysql>source z13_wcz_bar.sql

d)Wstaw plik z13_wcz_bar.sql do treści pracy.

5)Dla sprawdzenia wyświetl dane z tabeli

select * from nazwa_tabeli →(jaką masz nazwę tabelę)

6)Rozwiazanie dla 13a

7)Rozwiazanie dla 13b
Etap1
zliczanie ilości rekordów w bazie
plik → p13_b1.txt
Etap2
wyświetlenie wszystkich miast z ilościa ile razy nastapiła wygrana
plik → p13_b2.txt

Etap3
wyświetlenie ile razy nastapiła wygrana w ‘Warszawie’
plik → p13_b3.txt

Etap4
wyświetlenie ile razy nastapiła wygrana w miastach gdzie było wiecej niż 10 wygranych
plik → p13_b4.txt

Zadanie 14.
Etap1
Wyświetl zawartość bazy z zadania 10 (USE i SELECT)
Etap2
Do bazy danych z zadania 10 wstaw nowe pole wzrost_trzy_litery_nazwiska np. wzrost_bar typu całkowitego o 11 cyfrach. Pole wstaw za polem numer_trzy_litery_nazwiska. Użyj **ALTER TABLE.**
Etap3
Do utworzonej kolumny wpisz dane instrukcją **UPDATE.**
Etap4
Wyświetl zawartość bazy (SELECT)
Etap5
Skasuj 2 i 5 rekord bazy (DELETE)
Etap6
Wyświetl zawartość bazy (SELECT)
Etap7
Skasuj dane z obecnego drugiego rekordu z pola wzrost i waga instrukcją **UPDATE**.
Etap8
Wyświetl zawartość bazy (SELECT)

Zmiana struktury tabeli ALTER TABLE

Zastosowanie:

Możesz używać MySQL ALTER TABLE:

- dodać lub usunąć kolumny,
- zmienić typ danych kolumny,
- dodać klucz podstawowy,
- zmienić nazwę tabeli i wiele więcej.

Składnia ALTER TABLE MySQL:

ALTER TABLE nazwa_tabeli akcja1[, akcja2, ...]

Następnym po ALTER TABLE słowem kluczowym jest nazwa tabeli, do której chcesz wprowadzić zmiany. Po nazwie tabeli jest działanie, ktróre chcesz zastosować do tabeli. Akcja to może być dodanie nowej kolumny, dodanie klucza podstawowego .... czy zmienić nazwę tabeli. MySQL pozwala na wykonywanie wielu czynności na raz, oddzielając je przecinkami.

Przykład:
Etap1→stworzenie tabeli

```
CREATE TABLE zadania (
id_zadania INT NOT NULL,
tytul VARCHAR(45) NULL,
data_poczatkowa DATETIME NULL,
data_koncowa DATETIME NULL,
opis VARCHAR(200) NULL,
PRIMARY KEY (id_zadania),
UNIQUE INDEX id_zadania_UNIQUE (id_zadania ASC)
);
```

Etap2→Używając polecenia ALTER TABLE dodaj auto-increment do kolumny

Załóżmy, że chcemy aby kolumna id_zadania była zwiększana automatycznie o jeden, gdy wstawiamy nowe zadanie, musimy dodać auto-increment do kolumny. Aby to osiągnąć, musimy użyć instrukcji MySQL ALTER TABLE aby zmienić identyfikator zadania kolumny, aby uczynić go aby przyrost automatycznie był powiększany o jeden w następujący sposób:

```
ALTER TABLE zadania
CHANGE COLUMN id_zadania id_zadania INT(11) NOT NULL
AUTO_INCREMENT;
```

Etap3→Używając polecenia ALTER TABLE dodaj nową kolumnę do tabeli

Z powodu nowego wymogu, musimy dodać nową kolumnę o nazwie zadanie_kompletne, w której będzie przechowywany procent wykonania dla każdego zadania w tabeli zadania. W tym przypadku możemy wykorzystać polecenie ALTER TABLE aby dodać nową kolumnę w następujący sposób:

```
ALTER TABLE zadania ADD COLUMN zadanie_kompletne DECIMAL(2,1) NULL
AFTER opis;
```

Etap4→Używaj polecenia ALTER TABLE usuń kolumnę z tabeli

Powiedzmy, że nie chcemy już kolumny z opisem zadania, więc możemy tę kolumnę usunąć. Poniżej przykład jak usunąć kolumnę z tabeli:

```
ALTER TABLE zadania
DROP COLUMN opis;
```

Etap5 →Zmiana nazwy tabeli za pomocą instrukcji ALTER TABLE

Możemy użyć POLECENIA MySQL ALTER TABLE aby zmienić nazwę tabeli. Należy pamiętać, że przed zmianą nazwy tabeli należy wziąć poważnie pod uwagę, aby zobaczyć jej zależności od danych na poziomie aplikacji. Możemy zmienić

nazwę naszej tabeli zadania w następujący sposób:

ALTER TABLE zadania
RENAME TO praca;

Usuwanie rekordów DELETE

Polecenie DELETE pozwala na usunięcie określonych rekordów prze klauzule WHERE lub wszystkich rekordów.

Składnia:
DELETE FROM table_name WHERE search_condition

Np.
DELETE FROM dane WHERE miesiac = 'styczeń'

DELETE FROM dane
usunięcie całej zawartości tabeli w bazie danych

Usuwanie wszystkich wierszy w danej tablicy MySQL - TRUNCATE

Składnia polecenia TRUNCATE:

TRUNCATE TABLE nazwa_tablicy;

Polecenie TRUNCATE TABLE usuwa zupełnie wszystkie dane z tablicy o nazwie nazwa_tablicy. Logicznie, jest to ekwiwalent polecenia DELETE usuwającego wszystkie rekordy, jednak są praktyczne różnice. W przypadku tablic typu InnoDB polecenie TRUNCATE TABLE jest zamieniane na odpowiednie polecenie DELETE, więc w tym przypadku nie ma zupełnie różnic, jednak w innych przypadkach są różnice:

- Operacja TRUNCATE usuwa zupełnie i tworzy od nowa tablicę co jest znacznie szybsze niż usuwanie rekordu po rekordzie.
- Operacje TRUNCATE nie są bezpieczne w przypadku transakcji; baza na pewno zgłosi błąd jeśli mamy aktywną transakcję lub zamknięty dostęp do tablicy.
- Nie jest zwracana liczba usuniętych rekordów.
- Tak długo jak plik z definicją tablicy (plik 'nazwa_tablicy.frm') jest nieuszkodzony, tak długo można wykonać polecenie TRUNCATE TABLE i w ten sposób odtworzyć pustą tabelę nawet w przypadku kiedy dane z tabeli lub indeksy są uszkodzone.
- Uchwyt talicy, którym posługuje się wewnętrznie MyQL nie pamięta ostatnio użytej wartości AUTO_INCREMENT ale zaczynać odliczać od początku.

Polecenie TRUNCATE TABLE pochodzi od rozszerzenia języka SQL, które wprowadziła firma Oracle.

Zadanie 15.

Treść polecenia: Baza danych, która będzie obsługiwać transakcje.

transakcje (kliknij aby przenieść się do teorii)

Wykonaj skrypt mysql;
1)założenie bazy o nazwie ***hurtownia_nazwisko*** następujących polach:
id_nazwisko (klucz podstawowy)
imie_nazwisko
nazwisko_nazwisko
wiek_nazwisko
Aby widoczne były polskie znaki użyj DEFAULT CHARSET=utf8.
wymuszeniu silnika InnoDB (tzw składowanie tabel)
2)Wpisz dane

id	imie	nazwisko	wiek
1	Jan	Nowak	24
2	Ewa	Werner	30
3	Marcin	Tracz	33
4	Łucja	Kowalska	26
5	Paweł	Nowak	23

3)wyświetl wszystkie dane z bazy
4)Zmodyfikuj dane w tabeli tak, aby wiek wszystkich pracowników o id większym od 2 zmienić na 18. Obejmij tą instrukcję transakcją.

START TRANSACTION;
UPDATE *pracownicy* SET wiek = 18 WHERE id > 2;

5)Wyświetl wszystkie dane z bazy
6)Wycofaj zmianę, stosując słowo kluczowe ROLLBACK.
7)Wyświetl wszystkie dane z bazy. Czy dane powróciły do
8) Obejmij transakcją instrukcję dopisywania nowego pracownika. Dopisz siebie i zatwierdź transakcję.
9)Wyświetl wszystkie dane z bazy

Zadanie 16.

Treść polecenia: Baza danych, która będzie obsługiwać transakcje wraz z punktami przywracania.

Obejmij transakcją dwie instrukcje: zmień imiona wszystkich pracowników o id>1 na Marcin i nazwiska na Zagłoba. Po każdej z nich zdefiniuj punkt przywracania.
1. Definiujemy pierwszy punkt przywracania:

START TRANSACTION;
UPDATE *pracownicy* SET imie = 'Marcin' WHERE id > 1;
SAVEPOINT *punkt_pierwszy;*
2. Wyświetlamy zawartość tabeli:

3. Definiujemy drugi punkt przywracania:

UPDATE *pracownicy* SET nazwisko = 'Zagłoba' WHERE id > 1;

SAVEPOINT *punkt_drugi;*
4. Wyświetlamy zawartość tabeli:

5. Cofamy zmiany do poprzednio zdefiniowanego punktu przywracania:

ROLLBACK TO SAVEPOINT *punkt_pierwszy*;
6. Wyświetlamy zawartość tabeli:

7. Całość transakcji można nadal wycofać używając słowa kluczowego **ROLLBACK**. Zrób to. Przekonasz się, że zawartość tabeli wróci do stanu przed rozpoczęciem transakcji.

Część V
Pytania teoretyczne do programowania w PhpMyAdmin

Zadanie 17.
1)Użyj phpMyAdmin. Uruchom go poprzez: uruchomienia krasnala, następnie kliknij prawym zielone piór w zasobniku na dole po prawej stronie na pasku zadań i teraz wybierz WWW i phpMyAdmin.
użytkownik→root
hasło→krasnal

Załóż bazę danych o nazwie nazwisko_słuchacza_php (bez polskich liter małymi literami) o trzech polach:
numer (tinyint(4)),
nazwisko(text),
waga (int(11))
o nazwie tabeli kontakty_nazwisko_słuchacza np. kontakty_kowalski (bez polskich liter małymi literami).

Wpisz do niej pięć rekordów rekordów z sensowymi danymi.

Po założeniu bazy wpisz następujący skrypt (treść patrz poniżej). Będziesz musiał zmienić nazwę bazy oraz nazwę tabeli w treści skryptu poniżej.

Nagraj ten skrypt pod nazwą zad14_twoje_nazwiski.php w folderze c:\usr\krasnal\www\~twoje_konto

Skrypt uruchom poprzez:
http:\\127.0.0.1\~nazwa_konta\zad14_twoje_nazwiski.php

Po sprawdzeniu działania skryptu wykonaj:
1)umieść listing w pracy semestralnej.

2)Wykonaj wytłumaczenia wszystkich instrukcji użytych w skrypcie w formie tabeli:
.

	komenda PHP	wytłumaczenie
1	<?	początek skryptu
2	$user= 'root';	przypisanie do zmiennej $user wartości root→użytkownik, który loguje się do bazy to root
3	$pass= 'krasnal';	przypisanie do zmiennej $pass wartości krasnal→użytkowniem, który loguje się do bazy ma hasło krasnal

Skrypt do wpisania. Obsługuje on bazę zapisaną powyżej.

```
<?
$user= 'root';
$pass= 'krasnal';
$host = '192.168.0.29';
$base = 'baza';
$con=mysql_connect($host,$user,$pass);
mysql_select_db($base);
if($con) echo 'ok';
$ret = mysql_query("SELECT * FROM `kontakty`",$con);
echo "<br>";
while ($row = mysql_fetch_array($ret))
{
$y= count($row);
for($i=0;$i<$y;$i++ )
{
echo( $row[$i]." ");
}
echo "<br>";
}
if($_GET['sub']=='ok')
echo "dodaje";
mysql_query("INSERT INTO `kontakty` ( `numer` , `nazwisko` , `waga` ) VALUES
($_GET['num'], $_GET['nazw'], $_GET['waga');",$con);
?>

<form action=conn.php method=get>
<input name=num><br>
<input name=nazw><br>
<input name=waga><br>
<input name=sub type=submit value=ok>
</form>
```

Teoria

http://www.uz.zgora.pl/~agramack/files/BazyDanych/MySQL/mysql_01.pdf

Obsługa serwera Krasnal

Poniżej przedstawiam wam wszystko opcje (z krótkim opisem) które są dostępne po kliknięciu prawym przyciskiem myszy na ikonę naszego serwera (ta ikona znajduje się na pasku obok zegara i przypomina „pióro”):

-**Stop** – zatrzymanie pracy serwera,
-**Restart** – restart naszego serwera (restartuj swój serwer po każdej zmianie ustawień),
-**Ustawienia** – konfiguracja naszego serwera, dalsze opcje:
- httpd.conf, access.conf i srm.conf to pliki konfiguracyjne
- MySQL dalsze opcje:
  - *my.ini* – plik konfiguracyjny dla bazy danych,
  - *Hasło MySQL* – ustalamy sobie login i hasło,
  - *phpMyAdmin* – plik konfiguracji,
- PHP dalsze opcje:
  - *php.ini* – plik konfiguracyjny dla php,
  - *serwer SMTP* – ustawienia naszego serwera poczty wychodzącej (SMTP),
- *PHP 4 i PHP 5* – możesz wybrać PHP 4 lub 5 dla naszego serwera, przy aktualnie włączonej wersji będzie znaczek (dziubek),
- Konta – tworzenie i usuwanie kont użytkowników,
- Katalog z hasłem – tutaj możesz ustawić i zabezpieczyć jakiś katalog hasłem,
- Deinstalator – usunięcie krasnala z naszego PC-ta,
- Zaawansowane opcje – tutaj możesz troszkę bardzie skonfigurować serwerek,
-**Połączenia** – nie będę wymieniał opcji ale tutaj możesz zobaczyć dane dotyczące transferu i zsanować jakieś IP,
-**WWW** – opcje dotyczące kont WWW, dalsze opcje:
- localhost – strona główna naszego serwera,
- phpMyAdmin – edytor bazy danych,
- Konta – spis wszystkich kont www jakie są na naszym serwerze,
- WebAlizer – statystyki naszego serwera,
-**SMTP Serv** – serwer SMTP,
-**MySQL CMD** – monitor MySQL, wiersz poleceń,
-**MySQL Admin** – administracja bazą danych,
-**Aktualizacje** – tutaj sprawdzisz czy są aktualizacje dla naszego serwera,
-**Koniec** – zamkniecie pracy serwera.

Jak widać interfejs programu jest prosty, a sam program nie wymaga dużej konfiguracji ze strony użytkownika.

Jeśli chcemy aby nasz serwer był widoczny dla całej sieci, musimy zmienić jedną rzecz. Mianowicie wchodzimy do pliku httpd.conf (dostępny po kliknięciu prawym przyciskiem myszy na „pióro” i wejście w USTAWIENIA). Teraz szukamy linijki **ServerName 127.0.0.1**, i w miejsce: 127.0.0.1 wpisujemy swój adres IP. Zapisujemy plik i restartujemy nasz serwer. Teraz jak ktoś chce wejść na Twoją stronę musi w wyszukiwarce wpisać twój IP (oczywiście Twój serwer musi być włączony).

Gdy chcemy aby ludzie widzieli naszą stronę po wpisaniu Twojego IP, a nie stronę krasnala, należy wejść do katalogu c:\usr\apache\httpd\html, wykasować pliki które tam są z wyjątkiem .htaccess (jeśli go skasujesz przestanie działaś blokowanie IP) i wkleić tam swoją stronę.
Aby się dostać do phpMyAdmin należy w wyszukiwarce wpisać localhost/phpmyadmin. Login to: **root** a hasło: **krasnal**. Oczywiście, należy zmienić hasło na swoje, aby to zrobić zaloguj się na w/w dane i na stronie głównej zobaczysz „Zmiana hasło" kliknij i zmień hasło na swoje. Zaleca się zostawienie tego loginu, ponieważ wymagany jest przy niektórych skryptach (np. zpanel – panel administracyjny, podobny do DirectAdmin lecz darmowy).

Z tego powodu, że znaleziono błędy zaleca się zainstalowanie Patcha 1.0 (kliknij tutaj aby go pobrać z naszego downloadu)!!

Co to jest język zapytań SQL, do jakich baz danych się odnosi? Jakie są trzy podstawowe części?

Język **SQL** (*Structured Query Language*) służy do manipulowania danymi umieszczonymi w relacyjnych bazach danych. Jest językiem **uniwersalnym**, dzięki czemu praca na różnych systemach baz danych sprowadza się do wydawania tych samych lub podobnych komend tzw. **zapytań SQL**. Język SQL został zaimplementowany w większości relacyjnych systemów baz danych takich jak: **DB2**, **Oracle**, **InterBase**, **MySQL**, **dBase**, **Paradox**.
Składnię języka SQL można podzielić na trzy części:

- **język definiowania struktur danych** - DDL (*Data Definition Language*) - jest wykorzystywany do wszelkiego rodzaju operacji na tabelach, takich jak: tworzenie, modyfikacja oraz usuwanie,
- **język do wybierania i manipulowania danymi** - DML (*Data Manipulation Language*) - służy do manipulowania danymi umieszczonymi w tabelach, pozwala na wstawienie danych, ich prezentację, modyfikowanie oraz usuwanie,
- **język do zapewnienia bezpieczeństwa dostępu do danych** - DCL (*Data Control Language*) - jest używany głównie przez administratorów systemu baz danych do nadawania odpowiednich uprawnień do korzystania z bazy danych.

Co to jest Mysql?

Jest system zarządzania relacyjnymi bazami danych rozpowrzechiniany na podstawie licencji wolnej licencji.

Popularne typy danych MySQL

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

Funkcje operujące na datach

Funkcja	Sposób użycia	Przeznaczenie
HOUR()	HOUR(kolumna)	Zwraca samą godzinę ze wskazanej daty
MINUTE()	MINUTE(kolumna)	Zwraca same minuty ze wskazanej daty.
SECOND()	SECOND(kolumna)	Zwraca same sekundy ze wskazanej daty.
DAYNAME()	DAYNAME(kolumna)	Zwraca nazwę dnia tygodnia.
DAYOFMONTH()	DAYOFMONTH(kolumna)	Zwraca sam dzień miesiąca ze wskazanej daty (wyrażone liczbą).
MONTHNAME()	MONTHNAME(kolumna)	Zwraca nazwę miesiąca występującego we wskazanej dacie.
MONTH()	MONTH(kolumna)	Zwraca sam miesiąc ze wskazanej daty (wyrażony liczbą).
YEAR()	YEAR(kolumna)	Zwraca sam rok ze wskazanej daty.
ADDDATE()	ADDDATE(kolumna INTERVAL x typ)	Dodaje do daty przechowywanej w kolumnie x jednostek i zwraca wynik.
SUBDATE()	SUBDATE(kolumna INTERVAL x typ)	Odejmuje od daty przechowywanej w kolumnie x jednostek i zwraca wynik.
CURDATE()	CURDATE()	Zwraca bieżącą datę.
CURTIME()	CURTIME()	Zwraca bieżący czas.
NOW()	NOW()	Zwraca bieżącą datę i czas.
UNIX_TIMESTAMP()	UNIX_TIMESTAMP(data)	Zwraca liczbę sekund jaka upłynęła od początku tzw. epoki unixa lub od wskazanej daty.

Obliczenia na datach

Funkcja ADDDATE() i SUBDATE(), będące synonimami DATE_ADD() i DATE_SUB(), przeprowadzają obliczenia na datach. Ich składnia jest następująca:

ADDDATE(data, INTERVAL x typ)

Data może pochodzić z kolumny tabeli lub Może być wprowadzona ręcznie. Wartość x jest różna dla każdego typu. Dostępne typy to: SECOND, MINUTE, HOUR, DAY, MONTH i YEAR.

Mogą one tez występować w nast. następujących kombinacjach:

MINUTE_SECOND, HOUR_MINUTE, DAY_HOUR, oraz YEARS_MONTH.

Aby dodać do wskazanej daty dwie godziny, wpisz:

ADDATE(data, INTERVAL 2 HOUR)

Aby dodać dwa tygodnie do 31 grudnia 2002, wpisz:

ADDATE(`2002-12-31`, INTERVAL 14 DAY)

Aby odjąć od daty 15 miesięcy, wpisz:

SUBDATE(data, INTERVAL `1-3` YEAR_MONTH)

Ostatnie zapytanie informuje system, że chcesz odjąć od wartości przechowywanej w kolumnie jeden rok i trzy miesiące.

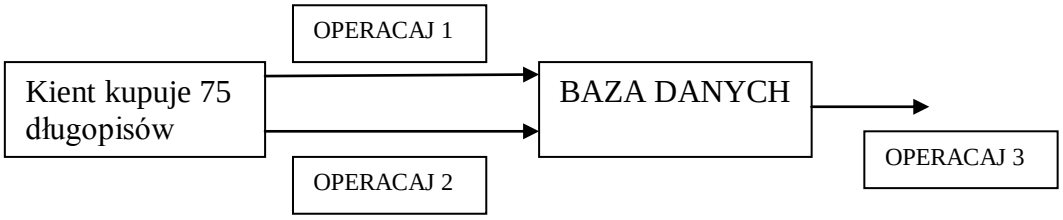
Opracowano na podstawie L. Welling, L. Thomson: PHP i MySQL. Tworzenie stron WWW. Helion 2003.

Metody składowania tabel

MyIsam - podstawowy, domyślny mechanizm składowania MySQL. Jest bardzo dobry i uniwersalny. Nie wiesz co wybrać? - wybieraj to. Ponadto posiada najwięcej narzędzi do naprawiania i sprawdzania tabel oraz tabele moga być kompresowane i obłsugują wyszukiwanie pełnotekstowe.
Memory (HEAP) - mechanizm składowania, który zamiast na dysku twardym przechowuje dane w pamięci RAM, w związku z czym po ew. wyłączeniu serwera dane te kasują się bezpowrotnie. Dane są ponadto hashowane. Idealny do tymczasowego przechowywania danych, np. sesji.
InnoDB - obsługują transakcje, co jest najistotniejszą różnicą charakteryzującą ten typ mechanizmu składowania. Ponadto działają szybciej niż MyISAM i obsługują klucze obce.

Transakcje w bazach danych

Czym w zasadzie są transakcje?
Przykład:
System zarządzający inwentarzem w hurtowni.
Gdy klient kupuje 75 sztuk długopisów, do bazy danych idą zapytania aktualizujące stan kasy, w której kupował oraz zmieniające liczbę długopisów w inwentarzu.
W przypadku gdy długopisów jest już bardzo mało, system automatycznie wysyła zamówienie do producenta długopisów na kolejne 2000 długopisów.
Gdyby w połowie wykonywania tych operacji system straciłby zasilanie to np. stan kasy zostałby poprawiony, ale stan długopisów nie zgadzałby się już z rzeczywistością. Mogłoby dojść potem do sytuacji, że kolejny klient chce zakupić 200 długopisów, a w inwentażu brakuje 50, ponieważ nie było dostawy długopisów. Dlaczego? Ponieważ przez awarię system nie wykrył, że długopisów jest już mało i zamówienie nie zostało wysłane do producenta.



OPERACJA 1→zmienia stan kasy o wartośc zakupionych długopisów (zmiana wartości pola w bazie danych np. kasa_posiada)
OPERACJA 2→zmienia stan długopisów w magazynie (zmiana wartości pola w bazie danych np. stan_długopisow)
OPERACJA 3→przygotowanie formularza zamówienia do producenta (hurtownia kupuje długopisy) , zmiana wartości pól w bazie danych w których zapisane jest jakie zamówienie zostało wystawiane.

Te trzy operacje muszą się wykonać jedna po drugiej. Zblokowanie operacji na bazie nazywamy transakcją

Definicja
Transakcje grupują wyrażenia (instrukcje) SQL, traktując ta grupę jako jedno.

Transakcja może składać się z jednego zapytania lub z wielu, to zależy od charakteru zadania do wykonania.

Transakcje składają się z etapów:

* **Rozpoczęcie transakcji, inaczej inicjalizacja** - po rozpoczęciu transakcji WSZYSTKIE kolejne zapytania SQL są traktowane jako część instrukcji transakcji, następuje ich zgrupowanie do momentu zamknięcia lub anulowania transakcji.
* **Wykonanie**
* **Zatwierdzenie transakcji** - zapisanie na trwałe wszystkich zmian dokonanych przez instrukcję transakcji. Jeśli nie będziesz kończył transakcji odpowiednio szybko, będzie dochodziło do blokowania całej aplikacji. Poprzez odpowiednio szybko rozumiem sytuację kiedy transakcja jest otwarta TYLKO do momentu, kiedy nie będzie pewne, że jej wynik będzie pozytywny. Nie należy zwlekać z zatwierdzeniem transakcji.
* **Anulowanie transakcji**, lub też wycofanie - kończy transakcję i porzuca wszystkie zmiany jakich miały dokonać instrukcje. Przydaje się w instrukcjach warunkowych aplikacji internetowej. Np. transakcja jest zatwierdzana jeśli przyjdzie potwierdzenie wpłaty pieniędzy na konto bankowe, w przeciwnym wypadku jest anulowana.

Transakcje mają następujące zasady:

ACID,
Atomicity,
Consistency,
Isolation,
Durability - Atomowość,
Spójność,
Izolacja,
Trwałość.
Oznacza to, że każda transakcja wykona się w całości, lub wcale. Podczas jej wykonywania dwie równoległe transakcje nie widzą wprowadzanych przez siebie zmian (zależy od poziomu izolacji) do momentu zatwierdzenia transakcji. Zmiany nie są też widoczne dla pozostałych użytkowników tej bazy danych. Po jej wykonaniu nie zostaną naruszane zasady integralności bazy. A cały system udostępni nam nienaruszone dane po nagłej awarii systemu.

Transakcje można używać samodzielnie, ale ich zastosowanie najczęściej idzie w parze z zastosowaniem kursorów w procedurach składowych.
Procedury składowe są udoskonaleniem struktury języka SQL, są osobnymi programami przetwarzającymi dane na poziomie bazy danych.

W teorii transakcji istnieje też pojęcie **punkt pośredni lub punkt zapisu** (save point), działające na tej samej zasadzie. Chodzi i zapisanie stanu transakcji do danego punktu i możliwość cofania się do tego momentu, wewnątrz transakcji.
To jak zapisywanie stanu gry w czasie wykonywania konkretnej misji w grze komputerowej.

5. Transakcje w MySQL.

W MySQL 6.0 mamy do dyspozycji standardowe procedury czyli:
START TRANSACTION (lub BEGIN) - rozpoczynanie transakcji

COMMIT (lub END) - zatwierdzanie
ROLLBACK - anulowanie
I dla InnoDB oraz Falcon
SAVEPOINT - punkty pośrednie
ROLLBACK TO SAVEPOINT - cofnięcie do punktu pośredniego
RELEASE SAVEPOINT - usuwa punkt pośredni o podanej nazwie z obecnej transakcji

8. Typy blokad
Blokady dzielą się na poziomy.
Są blokady na poziomie tabel
na poziomie wierszy.
I tutaj wynika pierwsze zagrożenie lub raczej problem w transakcjach.
Bo choć blokady na poziomie tabel można spotkać w większości (we wszystkich?) silników bazodanowych, to blokady wierszy tylko w tych bardziej zaawansowanych technologicznie.
Blokowanie CAŁEJ TABELI na czas wykonywania transakcji jest okropnie niewydajne. A co jeśli z tabeli korzysta kilku, albo o zgrozo, kilkunastu użytkowników, to nie będą mogli oni dokonać nic w całej tabeli do czasu, aż nie zakończy się tamta pierwsza transakcja. Teraz już widzisz jak ważne jest zamykanie transakcji kiedy już tylko można tego dokonać.

Blokowanie wierszy jest o wiele lepsze. Ponieważ blokuje tylko jeden rekord i jest mniejsze prawdopodobieństwa zastoju, lub w przypadku źle napisanych aplikacji internetowych, blokady całej strony.

9. Cofanie transakcji.
Jak już dobrze wiesz transakcje umożliwiają cofanie się do punktu pośredniego lub do stanu przed rozpoczęciem grupy zapytań.
Jak ta część mechanizmu współgra? Otóż na potrzeby tego projektu wynaleziono takie pojęcie jak dziennik transakcji. Nie jest on niczym innym jak dziennikiem logów aka zmian. Dzięki temu dziennikowi możliwe jest cofanie transakcji JUŻ ZATWIERDZONYCH.
Dzienniki taki przechowują zapis wszystkich transakcji zazwyczaj od momentu ostatniego backupu bazy. Jak to działa?

Załóżmy że mamy backup bazy z przed 3 dni i nastąpiła awaria, dajmy na to dysku. W takim wypadku musimy przywrócić stan z przed 3 dni.
Przywrócenie stanu z przed awarii jest banalnie proste po przywróceniu kopii bazy. Musimy potem tylko przywrócić zapisane transakcji w dzienniku transakcji, który na nowo wykona zapisane zapytania SQL.

Jeśli awaria nastąpiła z naszej winy, to następnym krokiem powinno być czynność zapobiegająca powtórzeniu się sytuacji.

PUNKTY PRZYWRACANIA
Podczas wykonywania transakcji możemy określić również punkty zapisu, tzw. **SAVEPOINT**. Po słowie kluczowym SAVENPOINT używamy identyfikatora.
SAVEPOINT ***nazwa_punktu_przywracania;***
Dzięki zdefiniowaniu identyfikatorów punktów zapisu możemy cofnąć zmiany przeprowadzane przez transakcję do określonego za pomocą identyfikatora punktu. Aby cofnąć zmiany do zdefiniowanego wcześniej punktu SAVENPOINT, używamy polecenia:

ROLLBACK TO SAVEPOINT ***nazwa_punktu_przywracania;***
Aby usunąć punkt przywracania używamy polecenia:
RELEASE SAVEPOINT ***nazwa_punktu_przywracania*****;**

KONTROLA TRANSAKCJI

Podczas wykonywania transakcji możliwy jest podział ich na mniejsze części za pomocą słowa kluczowego **SAVEPOINT**. Całość transakcji można nadal wycofać używając słowa kluczowego **ROLLBACK**.

Transakcje na przykładzie PHP

```
<?php
//Łączenie z bazą danych.
$pdo = new PDO('mysql:Host=host;dbname=nazwaBazy', 'Uzytkownik', 'Haslo');
//Ustawienie poziomu raportowania błędów.
$pdo ->setAttribute( PDO::ATTR_ERRMODE, PDO::ERRMODE_WARNING );
//rozpoczynanie transakcjii
try
{
$pdo -> query('start transaction');
//kto i co kupuje
$klient = 2;
$przedmiot = 1;
//pierwsze zapytanie zabierające pieniądze z konta.
$stmt = $pdo -> prepare('UPDATE users.money SET users.money=users.money-shop.cost
FROM users, shop WHERE shop.id='.$przedmiot.' AND users.id='.$klient.')');
$stmt -> execute();
$stmt -> closeCursor();
unset($stmt);
//drugie zapytanie dodające przedmiot.
$stmt = $pdo -> prepare('INSERT INTO items(what, owner) VALUES('.$przedmiot.',
'.$klient.')');
$stmt -> execute();
$stmt -> closeCursor();
//bez błędów - zatwierdzenie.
$pdo -> query('commit');
die('powiodło się');
}
catch(PDOException $e)
{
//W razie jakiegokolwiek błędu, wyskoczy wyjątek i od razu razem z nim transakcja zostanie
wycofana.
$pdo -> query('rollback');
//Wyświetlenie błędu.
die('Nie powiodła się! Błąd:'.$e->getMessage());
}
?>
```

Wejście w MySQLa na root czyli administartora (mysqlowego)
w DOS wpisujemy w folderze (c:\usr\mysql\bin) gdzie zianstalowany jest mysql:
mysql -u root –p Enter
podajemy hasło; Enter

Dla pakietu Krasnal jest:
krasnal

wpisując komendę mysql lub mysql -p ten drugi przypadek to jest wtedy gdy jest haselko, ktore nie musi byc takie samo jak na roota do serwera.
Jeżeli chcesz wejśc na konkretnego usera to musisz skorzystać jeszcze z opcji -u, np. mysql -u tcz -p

Zakładanie baz danych oraz uaktywnienie do bazy

CREATE DATABASE moja_baza;
USE moja_baza;

Polecenie CREATE DATABASE powoduje utworzenie nowej bazy danych o nazwie "moja_baza", następnie polecenie USE powoduje wybranie tej bazy danych jako aktywnej.

Usunięcie bazy

drop database nazwa_bazy;

Tworzenie tabeli.

np.

```
CREATE TABLE uzytkownicy(
    uid MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT,
    imie VARCHAR(30) NOT NULL,
    nazwisko VARCHAR(30) NOT NULL,
    email VARCHAR(50) NOT NULL,
    data_dodania DATETIME NOT NULL,
    PRIMARY KEY (uid)
);
```

```
CREATE TABLE newsletter_adresses (
id INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
email VARCHAR(40) NULL,
category SMALLINT NULL,
is_active TINYINT NOT NULL DEFAULT '0'
PRIMARY KEY (id)
)  ENGINE=MyISAM  DEFAULT CHARSET=utf8 COLLATE=utf8_polish_ci;
```

Powyższy układ można też zapisać w jednej linii - średnik informuje o końcu polecenia.

Za pomocą polecenia CREATE tworzymy tabelę uzytkownicy.
Następnie określamy kolumny i ich rodzaje,

czyli np. kolumna **"uid"** typu MEDIUMINT, bez znaku - UNSIGNED (typ określa liczby z zakresu od -n do n, a w przypadku typu unsigned wartość liczona jest od 0 w górę),
NOT NULL (czyli musi mieć jakąś wartość), AUTO_INCREMENT - inkrementuj (zwiększaj o 1) przy dodawaniu nowego rekordu.
Kolejne polecenia są analogiczne: **imie** o długości 30 znaków, **nazwisko** o długości 30 znaków,
email o długości 50 znaków oraz data dodania (format daty DATETIME, czyli rrrr-mm-dd gg-mm-ss, np. 2006-03-12 15:23:01) - i wszystkie kolumny muszą mieć jakąś wartość (NOT NULL).
PRIMARY KEY określa, która z kolumn jest kluczem głównym (warto wspomnieć, że pole tego typu powinno mieć unikatową wartość - czyli w przypadku tabeli "uzytkownicy" każdy rekord musi mieć inny "uid" - ID użytkownika).
Na końcu zamykamy nawias i stawiamy średnik. Jeżeli nie wystąpiły żadne błędy powinna pojawić się w bazie nowa tabela o podanych wyżej kolumnach.

Sprawdzenia tabel znajdujących w konkretnej bazie danych można użyć polecenia:
SHOW TABLES;

Sprawdzenia, jakie kolumny znajdują się w danej tabeli wystarczy wpisać:
SHOW COLUMNS FROM nazwa_tabeli;

Sprawdzenia jakie bazy danych istnieją na koncie zalogowanego użytkownika używa się:
SHOW DATABASES;
Polecenie SHOW wypisuje nazwy tabel, baz danych czy też kolumn danej tabeli wraz typami i określonymi parametrami.

INSERT - wstaw danych do tabeli
Zapytanie INSERT odnosi się do wstawiania nowych rekordów. INSERT można zastosować na co najmniej dwa sposoby:

sposób 1
INSERT INTO nazwa_tabeli VALUES ('wartosc_kol_1', 'wartosc_kol_2', NULL, 'wartosc_kol_n');

W tym wypadku podaje się kolejno wartości dla kolumn. Jeśli kolumna ma zawierać puste pole trzeba wpisać NULL (w powyższym przykładzie jest to kolumna trzecia).
Sposób ten powinno się wykorzystywać, jeśli wszystkie kolumny mają zostać wypełnione. Dla niewielkich tabel o kilku kolumnach może okazać się to przydatne i szybsze, ale również bardzo podatne na błędy (kolejność musi być zachowana).

sposób 2:
INSERT INTO nazwa_tabeli (kolumna_1, kolumna_7) VALUES ('wartość kol. pierwszej', 'wartość kol. siódmej');

Tutaj sam decydujesz jakie wartości i gdzie chcesz wpisać. Nie musisz ich wstawiać po kolei. Jeżeli większość kolumn ma mieć wartości domyślne to zaleca się właśnie takie użycie tego polecenia.

np. jak wstawić przykładowy rekord do tabeli, którą utworzyłem wcześniej:
INSERT INTO uzytkownicy (imie, nazwisko, email, data_dodania) VALUES ('Bogdan', 'Przykładowski', 'b.example@domena.pl', NOW());
Do określenia daty dodania użyłem funkcji NOW(), której używa się do określenia aktualnego czasu. Funkcja NOW() nie jest wpisana w apostrofy.

Manipulacja uprawnieniami.

Wszelkie zmiany dotyczące tabel w bazie mysql można przeprowadzać za pomocą poznanych wcześniej poleceń, select, insert, update, delete. Można również w łatwy sposób zarządzać uprawnieniami za pomocą poleceń GRANT (dodawanie uprawnień) i REVOKE (odbieranie uprawnień).

Ustawienie hasła dla użytkownika "root"
mysql> UPDATE user SET Password = PASSWORD('hasło') WHERE user = 'root';
Za pomocą polecenia UPDATE zmieniliśmy wartość pola Password na nowe hasło, tam gdzie użytkownik jest zdefiniowany jako root (dwa rekordy). Hasła w MySQL kodowane są za pomocą funkcji PASSWORD(), dlatego musimy jej użyć, aby zapisać hasło w tabeli.
Po zmiania hasła należy jeszcze przeładować uprawnienia poleceniem:
mysql> FLUSH PRIVILEGES;

Hasło można też zmienić za pomocą prostszego polecenia:
mysql> SET PASSWORD FOR root = PASSWORD('hasło');
należy niezapomnieć o przeładowaniu uprawnień.

Dodawanie użytkowników i uprawnień dla nich
- służy do tego polecenie GRANT:
mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON nazwa_bazy.nazwa_tabeli TO nazwa_użytkownika@localhost;
Uwaga. jeżeli zamiast nazwa_bazy lub nazwa_tabeli użyjemy znaku * co oznacza to dotęp do wszystkich baz lub tabel, a znak *.* dostęp do dowolnej bazy wraz t wszystkimi tabelami.
W tym wypadku nowy użytkownik ma prawa do wykonywania instrukcji SELECT, INSERT, UPDATE i DELETE ale musi logować się z hosta lokalnego - odpowiedzialny jest za to ten fragment polecenia - nazwa_użytkownika@localhost. Jeżeli nie chcemy dawać tak wielkich uprawniwń możemy dać polecenie np.
mysql> GRANT SELECT ON nazwa_bazy.nazwa_tabeli TO nazwa_użytkownika@%;
w tym przypadku nowy użytkownik ma prawo tylko do przeglądania udostępnionych tabel lub baz ale może logować się ze wszystkich hostów - @%.

Można też sprecyzować i ograniczyć dostęp tylko do konkretnych pól tabeli:
mysql> GRANT SELECT(nazwa_pola1,nazwa_pola2),UPDATE(nazwa_pola1, nazwa_pola2) ON nazwa_bazy.nazwa_tabeli TO nazwa_użytkownika@localhost;

Aby użytkownikowi nadać komplet uprawnień stosujemy polecenie ALL PRIVILEGES:
mysql> GRANT ALL PRIVILEGES ON nazwa_bazy.nazwa_tabeli TO nazwa_użytkownika@localhost;
Możemy utworzyć też użytkownika który będzie miał wszystkie uprawnienia oraz będzie mógł innym te uprawniwnia (granty) nadawać i odbierać - polecenie WITH GRANT OPTION:
mysql> GRANT ALL PRIVILEGES ON nazwa_bazy.nazwa_tabeli TO nazwa_użytkownika@localhost WITH GRANT OPTION;
Narazie tworzyliśmy użytkowników którzy nie musieli podawać hasła przy logowaniu.
Można też utworzyć użytkownika z dostępem do wszystkich baz danych i tabel, ale z wymaganym hasłem 'haslo':
mysql> GRANT ALL PRIVILEGES ON *.* TO nazwa_uzytkownika@localhost IDENTIFIED BY 'haslo';

Usuwanie uprawnień użytkownika
- służy do tego polecenie REVOKE: mysql> REVOKE SELECT ON nazwa_bazy.nazwa_tabeli FROM nazwa_uzytkownika@localhost;
Jak widać z powyższego przykładu, aby usunąć uprawnienia nadane użytkownikowi, posługujemy się podobną składnią jak przy dodawaniu uprawnień. Różnice są dwie: zamiast słowa GRANT występuje REVOKE, a zamiast TO - FROM.

Można też usunąć dostęp tylko do konkretnych pól tabeli:
mysql> REVOKE SELECT(nazwa_pola1,nazwa_pola2),UPDATE(nazwa_pola1, nazwa_pola2) ON nazwa_bazy.nazwa_tabeli FROM nazwa_użytkownika@localhost;

Tworzenie bazy z użyciem PHP

```
<?php

mysql_connect("127.0.0.1","admin","haslo");

$zapytanie = "CREATE DATABASE daneosobowe";
$wynik = mysql_query($zapytanie);
if($wynik) echo "Baza została założona !<br>";
else {
echo "Baza nie została założona !<br>";
exit;
}

mysql_select_db(daneosobowe);

$zapytanie = "CREATE TABLE dane ( ".
"id int(11) DEFAULT '0' NOT NULL auto_increment, ".
"imie char(30), ".
"nazwisko char (30), ".
"dataur date DEFAULT '0000-00-00' NOT NULL, ".
"adres blob, ".
"telefon int(15), ".
"UNIQUE id (id), ".
"PRIMARY KEY (id) ".
")";
$wynik = mysql_query($zapytanie);
if($wynik) echo "Tabela założona prawidłowo !<br>";
else {
echo "Blad ! Tabela nie została założona !<br>";
exit;
}

$zapytanie = "INSERT INTO dane VALUES(1,'Jan','Kowalski','1981-12-28','ul.Długa 6 m 23
Toruń','7632325')";
$wynik = mysql_query($zapytanie);
if($wynik) echo "Pierwszy rekord dodany prawidłowo !<br>";
else {
echo "Błąd !!! Pierwszy rekord nie został dodany !<br>";
}

?>
```

Po uruchomieniu skryptu na serwerze powstanie nasza baza z tabelką dane i pierwszym rekordem. By każdy wiedział czy jego skrypt wykonał się prawidłowo dodałem warunki jeżeli w którymś momencie wystąpi błąd poinformuje nas o tym odpowiedni komunikat. Zawartość skryptu jest podobna do pliku zawierającego komendy dla serwera SQL. Tu jednak plik interpretuje PHP więc musimy użyć jego funkcji mysql_query(zapytanie SQL). Analizując powyższy kod widać, że jest podobny on do pliku z metody pierwszej.

W PHP do połączenia się z serwerem MySQL służy komenda mysql_connect("127.0.0.1","admin","haslo"); jako jego wartości podajemy adres hosta, nasz login i hasło. Do zmiennych przypisujemy zapytania, a następnie poprzez funkcję mysql_query wykonujemy je. Po utworzeniu bazy musimy ją wybrać by dalej można było z nią pracować, służy do tego funkcja mysql_select_db(nazwa-bazy); . Dalej skrypt wykonuje te same polecenia które mieliśmy w sposobie pierwszym.

Jak założyć nowego użytkownika MySQL przez konsole?
Umieszczam małą ściągę pt. Jak założyć nową bazę mysql i użytkownika, który będzie miał pełne prawa dostępu do tej bazy (create table, drop table, delete, update etc). Użytkownik ten jednak nie może mieć prawa nawet odczytu z innych baz na serwerze MySQLa.

Załóżmy, że projekt nazywa się owidiusz. Najpierw zakładam bazę:
create database owidiusz;

Jeżeli ta baza będzie przechowywać dane w kodowaniu UTF8, dodajemy:
create database owidiusz character set utf8 collate utf8_polish_ci;

(collate pozwala na poprawne sortowanie poslkich znaków!)

Następnie uzytkownika, nadajemy mu od razu hasło:
create user owidiusz identified by 'owidiusz123';

Nadajemy uprawnienia do bazy:
GRANT ALL PRIVILEGES ON owidiusz.* TO 'owidiusz'@'localhost';

I odświeżamy uprawnienia:
flush privileges;

I to wszystko :), można korzystać z bazy.

Dodawanie nowego użytkownika
CREATE USER user [IDENTIFIED BY [PASSWORD] 'password']

set password for ziutek=password('legia01');

Uruchomianie phpMyAdmin
Sposób1
na pióro zielone (klik prawym)□ WWW□phpMyAdmin
Sposób2
http://localhost/phpmyadmin/

login: root
hasło: krasnal

SELECT - wybierz
Służy do wybierania rekordów z bazy danych. Składnia jest bardzo prosta:
SELECT * FROM nazwa_tabeli;
Powyższa instrukcja spowoduje wybranie wszystkich rekordów z tabeli o danej nazwie. Natomiast, aby wybrać określone kolumny należy użyć następującej składni:
SELECT kolumna1, kolumna2 FROM nazwa_tabeli;

Jeżeli nie użyjemy żadnych instrukcji warunkowych zostaną wybrane wszystkie wypisane kolumny i np. wyświetlone na ekranie.

Instrukcje warunkowe i dodatkowe

Niezwykle ważnymi elementami, w operowaniu baz danych, są instrukcje "sterujące" czy "warunkowe". Są szczególnie ważne podczas budowy bardziej skomplikowanych projektów. Nie oznacza to, że są one niezbędne - wręcz przeciwnie - możesz wcale ich nie używać.

Warunek WHERE

Możesz jej użyć do określenia, do których rekordów chcesz się odwołać - czyli sprecyzować zapytanie. Określasz wartość danej kolumny rekordu, np.:

SELECT * FROM nazwa_tabeli WHERE imie='Marian';

Powyższe zapytanie powinno wyświetlić wszystkie rekordy (i wszystkie kolumny), których wartość kolumny "imie" odpowiada ciągowi znaków "Marian". Wartość może być dowolna.

Do określenia warunków możemy stosować operatory takie jak: =, <, >, <=, >= czy != (różny) - czyli te znane z języków programowania. Oprócz tego istnieją jeszcze inne:

Operator Znaczenie
IS NULL - nie ma wartości
IS NOT NULL - ma jakąś wartość
BETWEEN - mieści się w zakresie (jest pomiędzy)
NOT BETWEEN - nie mieści się w zakresie (jest poza)

Rys. 2. Polecenie insert, select oraz instrukcja warunkowa - efekty

Oprócz tego możemy łączyć ze sobą warunki poprzez OR (także ||) - oznacza logiczne "lub" czyli, że co najmniej jeden warunek został spełniony, oraz AND (także &&) - oznacza logiczne "i" czyli, że wszystkie warunki zostały spełnione. Jeżeli użyjesz samego NOT (lub !) - będzie to oznaczać, że do wykonania instrukcji warunek nie może być spełniony.

LIMIT

Jest to ograniczenie wyników zapytania. Jeśli posłużysz się powyższą instrukcją WHERE, wyświetlą się wszystkie rekordy, których kolumna "imie" ma wartość "Marian". Jeżeli chcesz wyświetlić jeden rekord na końcu zapytania wpisz LIMIT 1, np.:

SELECT * FROM nazwa_tabeli LIMIT 1;

Zostanie wyświetlony pierwszy rekord spełniający określone kryteria.

ORDER BY

Za pomocą tej instrukcji możemy uporządkować (posortować) wyniki zapytania. Mamy tutaj dwie możliwości: ASC - alfabetycznie a-Z lub liczbowo rosnąco (jeśli sortujesz wg. dat to pierwsza, która się wyświetli będzie tą najstarszą) oraz DESC - odwrotnie do ASC.

SELECT imie, nazwisko, data_dodania ORDER BY data_dodania DESC;

Instrukcje niebezpieczne

Istnieją też polecenia, na które należy uważać. Jeden mały błąd może spowodować, że baza danych zostanie zepsuta (a konkretnie zapisane w niej wartości). Przed testowaniem tych poleceń zaleca się zrobienie kopii zapasowej bazy.

UPDATE - aktualizuj

Jeżeli chcesz ponownie zapisać coś do jakiegoś rekordu użyj UPDATE. Jednak uważaj - jeżeli nie zastosujesz odpowiedniej konstrukcji zapytania z wyraźnym warunkiem określającym jeden rekord (lub więcej) - zostaną zaktualizowane wszystkie rekordy.

Ogólna postać zapytania wygląda następująco:
UPDATE nazwa_tabeli SET kolumna='wartosc';
Jednak zwykle używa się instrukcji WHERE do określenia, który rekord (lub rekordy) ma być uaktualniony.
Przykładem uaktualnienia jedynego możliwego rekordu w tabeli jest następująca składnia:
UPDATE uzytkownicy SET imie='Romek' WHERE uid = 2;

Rys. 3. Polecenie update - efekty
DELETE - usuń
Czasami przyjdzie potrzeba usunąć jakiś rekord - do tego służy polecenie DELETE:
DELETE FROM nazwa_tabeli WHERE kolumna='wartosc';
Najbezpieczniejszym sposobem usuwania tylko jednego rekordu, jest określenie unikalnego identyfikatora, który nie występuje w innych rekordach, czyli np.
DELETE FROM uzytkownicy WHERE uid = 1;
Jeżeli nie zastosujesz polecenie WHERE stracisz nieodwracalnie wszystkie rekordy w tej tabeli!
DROP - wyrzucać
Za pomocą tego polecenia możesz usunąć np. całą tabelę - DROP nazwa_tabeli; lub nawet całą bazę danych - DROP nazwa_bazy;. Pamiętaj jednak, że wszystkie dane zostaną utracone bezpowrotnie, dlatego najlepiej zrobić kopię bazy danych.
Podsumowanie
Powyżej przedstawiono przykładowe zapytania MySQL, które pomogą postawić pierwsze kroki. Znajomość poleceń INSERT, SELECT, UPDATE oraz LIMIT, ORDER BY oraz WHERE jest wystarczająca do wykorzystania bazy danych w dosyć funkcjonalny sposób.
Podczas uaktualniania / kasowania rekordów należy postępować ostrożnie. Zawsze należy używać instrukcji warunkowych - chyba, że wszystkie rekordy spełniają dany warunek.
Wartości tabel, np. w instrukcjach warunkowych, określa się co najmniej na trzy sposoby: operatorem pisanym (WHERE kolumna IS NULL), dla wartości znakowych - wartość wpisywana jest w nawiasie (WHERE kolumna='wartosc') oraz dla wartości liczbowych - można podać liczbę bez nawiasów (WHERE kolumna != 2).

Pytanie składnia instrukcji select

Aby pobrać dane zapisane w tabeli należy użyć zapytania SELECT. Jego postać ogólna prezentuje się następująco:
SELECT co_zaprezentować FROM nazwa_tabeli
  [WHERE warunki_wyszukiwania]
  [ORDER BY sortowanie [ASC | DESC], ...]
  [LIMIT [ofset,] ilość_wierszy];
W miejscu co_zaprezentować podaj (po przecinku) listę kolumn, które chcesz zawrzeć w zestawieniu. W miejscu nazwa_tabeli podaj nazwę tabeli, z której pobierzesz dane. Wybierając trzy kolumny do zestawienia z tabeli pracownicy napiszesz następująco:
SELECT imie, nazwisko, placa FROM pracownicy;
Spowoduje to wyświetlenie wszystkich rekordów, jednak w zestawieniu zostaną zaprezentowane jedynie wartości trzech pól: imie, nazwisko i placa:
+------------+--------------+---------+

```
| imie       | nazwisko     | placa   |
+------------+--------------+---------+
| Jan        | Kowalski     | 1200.00 |
| Izabela    | Kwiatkowska  |    NULL |
| Aleksander | Borowiecki   | 1500.34 |
| Aniela     | Michałkowska |  854.29 |
| Katarzyna  | Kowalska     | 1200.00 |
+------------+--------------+---------+
```

Aby w zestawieniu umieścić wszystkie pola można użyć w miejscu co_zaprezentować znaku '*' (gwiazdki):

```
SELECT * FROM pracownicy;
```

Zostaną wówczas wyświetlone wszystkie rekordy znajdujące się w tabeli:

```
+------------+--------------+---------------+---------+
| imie       | nazwisko     | data_urodzenia | placa   |
+------------+--------------+---------------+---------+
| Jan        | Kowalski     | 2002-07-20     | 1200.00 |
| Izabela    | Kwiatkowska  | NULL           |    NULL |
| Aleksander | Borowiecki   | 1952-08-06     | 1500.34 |
| Aniela     | Michałkowska | 1970-05-23     |  854.29 |
| Katarzyna  | Kowalska     | 2002-07-02     | 1200.00 |
+------------+--------------+---------------+---------+
```

Dzięki klauzuli WHERE jesteś w stanie wpłynąć na zakres prezentowanych danych. Dzięki niej możesz dokładnie definiować co chcesz uzyskać swoim zapytaniem. Specyfikując dokładne warunki wyszukiwania można z tabeli zawierającej setki tysięcy rekordów wybrać tylko kilka interesujących w danym momencie informacji.

Klauzulę WHERE stosuje się najczęściej w poleceniu SELECT. Ma ona jednak zastosowanie także w innych poleceniach operujących na danych takich jak UPDATE, DELETE itp.

Stosując operatory przyrównania możesz dokładnie określić, jakie informacje chcesz pobrać. Dozwolone w MySQL operatory przyrównania to:

=	Równe
>	Większe
>=	większe równe
<	Mniejsze
<=	mniejsze równe
<> lub !=	Różne
LIKE	służy głównie do porównywania danych łańcuchowych

Przykłady zastosowania:

```
SELECT * FROM pracownicy WHERE placa >= 1000;
```

Spowoduje wyświetlenie listy pracowników, których płaca jest większa lub równa 1000:

```
+------------+------------+---------------+---------+
| imie       | nazwisko   | data_urodzenia | placa   |
+------------+------------+---------------+---------+
| Jan        | Kowalski   | 2002-07-20     | 1200.00 |
| Aleksander | Borowiecki | 1952-08-06     | 1500.34 |
+------------+------------+---------------+---------+
```

SELECT imie, nazwisko, placa FROM pracownicy WHERE nazwisko = 'Kowalski';
Spowoduje wyświetlenie danych (tylko imię, nazwisko i placa) wszystkich pracowników, których nazwisko brzmi dokładnie Kowalski:

```
+------+----------+-------------------+
| imie | nazwisko | placa             |
+------+----------+-------------------+
| Jan  | Kowalski | 1200.00           |
+------+----------+-------------------+
```

SELECT * FROM pracownicy WHERE nazwisko LIKE 'K%';
Spowoduje wyświetlenie wszystkich pracowników, których nazwisko rozpoczyna się na literę 'K':

```
+-----------+-------------+---------------+-------------------+---------+
| imie      | nazwisko    | data_urodzenia | adres             | placa   |
+-----------+-------------+---------------+-------------------+---------+
| Jan       | Kowalski    | 2002-07-20     | Kwiatowa 8, Poznań | 1200.00 |
| Izabela   | Kwiatkowska | NULL           | NULL              |    NULL |
| Katarzyna | Kowalska    | 2002-07-02     | NULL              | 1200.00 |
+-----------+-------------+---------------+-------------------+---------+
```

Dzięki zastosowaniu znaków globalnych (% i _) istnieje możliwość przyrównania do dowolnego ciągu znaków. Znak '%' (procent) zastępuje dowolną ilość znaków. Znak '_' (podkreślenie) zastępuje dokładnie jeden znak. Zapytanie:
SELECT imie, nazwisko FROM pracownicy WHERE nazwisko LIKE 'Kowalsk_';
spowoduje wyświetlenie wszystkich pracowników, których nazwisko zaczyna się ciągiem znaków 'Kowalsk' i zaraz po nim występuje jeden dowolny znak:

```
+-----------+----------+
| imie      | nazwisko |
+-----------+----------+
| Jan       | Kowalski |
| Katarzyna | Kowalska |
+-----------+----------+
```

Warunki wyboru podawane za WHERE można łączyć ze sobą stosując operatory AND oraz OR. Dzięki temu istnieje możliwość zbudowania zapytania bardziej złożonego, a co za tym idzie bardziej dokładnego. W momencie zastosowania operatora AND wszystkie połączone tak warunki muszą zostać spełnione, aby w wyniku pojawił się dany rekord. Jeśli zastosujesz do połączenia warunków operator OR - wówczas może zostać spełniony tylko jeden z warunków wchodzących w skład zapytania. Wydając zapytanie:
SELECT * FROM pracownicy WHERE (placa > 500 AND placa < 1000) OR nazwisko = 'Kowalski';
spowodujesz wyświetlenie w wyniku wszystkich pracowników, których płaca mieści się w zakresie 500-1000 oraz pracowników o nazwisku 'Kowalski':

```
+--------+-------------+---------------+---------+
| imie   | nazwisko    | data_urodzenia | placa   |
+--------+-------------+---------------+---------+
| Jan    | Kowalski    | 2002-07-20     | 1200.00 |
| Aniela | Michałkowska | 1970-05-23     |  854.29 |
+--------+-------------+---------------+---------+
```

Dane w tabeli mogą być przechowywane w dowolnej kolejności. Możesz jednak spowodować ich pobranie w ściśle określonym porządku. Kolumny, według których MySQL ma posortować dane podaje się po klauzuli ORDER BY oddzielone

przecinkami. Chcąc więc uszeregować listę pracowników rosnąco według nazwiska i malejąco według płacy wpiszesz następujące polecenie:
SELECT * FROM pracownicy ORDER BY nazwisko ASC, placa DESC;
ASC oznacza sortowanie rosnąco według podanego pola, DESC natomiast oznacza sortowanie malejące:

```
+------------+--------------+----------------+---------+
| imie       | nazwisko     | data_urodzenia | placa   |
+------------+--------------+----------------+---------+
| Aleksander | Borowiecki   | 1952-08-06     | 1500.34 |
| Katarzyna  | Kowalska     | 2002-07-02     | 1200.00 |
| Jan        | Kowalski     | 2002-07-20     | 1200.00 |
| Izabela    | Kwiatkowska  | NULL           |    NULL |
| Aniela     | Michałkowska | 1970-05-23     |  854.29 |
+------------+--------------+----------------+---------+
```

Używając klauzuli LIMIT spowodujesz wyświetlenie jedynie części rekordów. Aby pobrać dwa pierwsze rekordy napisz:
SELECT * FROM pracownicy LIMIT 2;
W wyniku otrzymasz:

```
+---------+-------------+----------------+---------+
| imie    | nazwisko    | data_urodzenia | placa   |
+---------+-------------+----------------+---------+
| Jan     | Kowalski    | 2002-07-20     | 1200.00 |
| Izabela | Kwiatkowska | NULL           |    NULL |
+---------+-------------+----------------+---------+
```

Wypełniając pole ofset wyświetlisz podaną ilość rekordów od pewnego miejsca w tabeli. Chcąc pobrać rekordy od 3 do 7 napisz następująco:
SELECT * FROM pracownicy LIMIT 2, 5;
Spowoduje to wyświetlenie pięciu rekordów (o ile tyle istnieje w bazie) poczynając od rekordu trzeciego:

```
+------------+--------------+----------------+---------+
| imie       | nazwisko     | data_urodzenia | placa   |
+------------+--------------+----------------+---------+
| Aleksander | Borowiecki   | 1952-08-06     | 1500.34 |
| Aniela     | Michałkowska | 1970-05-23     |  854.29 |
| Katarzyna  | Kowalska     | 2002-07-02     | 1200.00 |
+------------+--------------+----------------+---------+
```

MySQL pozwala na połączenie ze sobą wielu opcji, dzięki którym można bardzo dokładnie zawęzić poszukiwaną ilość informacji. Przykład bardziej złożonego zapytania można przedstawić następująco:

```
SELECT imie, nazwisko, placa FROM pracownicy
  WHERE placa >= 500 AND placa < 1200
  ORDER BY nazwisko
  LIMIT 5;
```

Powyższe zapytanie spowoduje wygenerowanie następującego zestawienia:

```
+--------+--------------+--------+
| imie   | nazwisko     | placa  |
+--------+--------------+--------+
| Aniela | Michałkowska | 854.29 |
+--------+--------------+--------+
```

Jak widać połączono w tym momencie warunek wyboru według płacy pomiędzy 500 a 1200 z sortowaniem danych według nazwiska i ograniczeniem wyniku tylko do pięciu pierwszych rekordów.

Najlepiej to zrobić już w bashu, a pozniej stosując komende
mysql -p nazwa_bazy < plik_tabeli
wrzuca sie tabele do bazy. Tutaj taki pliczek powinien mieć albo rozszerzenie *.sql lub nazywa się dump. Tak więc całość może wyglądać tak:
mysql tcz -p < dump.

Teraz słówko o tworzeniu użytkowników
Żeby z baz danych mogły korzystać skrypty PHP3 zakładamy uzytkownika operator nadajemy mu jakieś hasło np. 123 tworzymy pliczek wg wzoru:
grant select,insert,update,delete on nazwa_bazy.* to operator@localhost identified by '123';
grant select,insert,update,delete on nazwa_bazy.* to operator@'%' identified by '123';

Nastepnie robimy mysql -p mysql < pliczek

Jeżeli chcemy dać dostęp koledze do jego baz danych to sprawa wygląda następująco (to przerobiony pliczek):
grant all on nazwa_bazy.* to user@localhost identified by 'haslo';
grant all on nazwa_bazy.* to user@'%' identified by 'haslo';

mysqladmin
mysqladmin jest narzędziem, za pomocą którego możemy tworzyć i usuwać bazy oraz przeładowywać konfigurację. Nie będziemy go używać do wyłączania serwera, bo od tego jest skrypt omówiony powyżej. Narzędzie wywołuje się poleceniem mysqladmin, a flagi i argumenty (opcje) polecenia służą do wykonywania określonych zadań. Ponieważ wcześniej zmieniliśmy hasło dla użytkownika mysql, winniśmy przy każdym poleceniu dodać -u mysql i -p na końcu (aby klient zapytał nas o hasło), np tak dla komendy status:
mysqladmin -u mysql status -p
Enter password:
Uptime: 6720 Threads: 1 Questions: 1 Slow queries: 0 Opens: 6 \
Flush tables: 1 Open tables: 0 Queries per second avg: 0.000
Kilka przydatnych komend:
- create nazwabazy - tworzy nową bazę danych o nazwie nazwabazy.
- drop nazwabazy - usuwa bazę danych o nazwie nazwabazy
- flush-privileges albo reload - obie opcje robią to samo - przeładowują tablice uprawnień. Powinniśmy wykonać przeładowanie zawsze, gdy dodamy np nowego

użytkownika do jakiejś bazy, ponieważ do czasu przeładowania takie konto jest nieaktywne.

mysqldump

mysqldump to program, służący do "zrzucania" danych - czyli do robienia kopii zapasowych. Polecenie to jest przydatne w przypadku wykonywania kopii zapasowych podczas aktualizacji MySQL czy też przenoszenia danych na inny serwer.

Kilka przydatnych opcji:

- --databases baza1 baza2... - zrzuca dane z baz, których nazwy podaliśmy w liście oddzielonymi spacjami.
- --all-databases - to samo co wyżej ale dla wszystkich bazy.
- --add-drop-table - dodaje DROP TABLE do skryptów tworzących tabele z backup-u (jak będziemy przywracać dane). Przydatne, kiedy np chcemy przywrócić już istniejącą tabelę. Spowoduje to najpierw jej usunięcie, a następnie utworzenie od nowa z danych, które mieliśmy w kopii zapasowej. Ogólnie, żeby uniknąć ewentualnych problemów z duplikatami wierszy, itp. warto tą opcję włączyć.
- --no-create-info - w kopii nie zostanie zawarta informacja o tworzeniu tabel (nazwy tabel, typy pól, indeksy itp.). Przydatne, jeżeli chcemy zrobić kopię tylko danych, a nie całej struktury bazy.
- --no-data - Ta opcja pozwala zapisać samą informację o strukturze bazy i tabel, nie zapisuje natomiast danych.
- --opt nazwabazy - tworzy kopię bazy nazwabazy wraz z rozszerzonymi informacjami MySQL, blokowaniem tabel, itd. Chyba najczęściej stosowana opcja przy robieniu kopii zapasowych.

Należy pamiętać, że wynik polecenia mysqldump należy przekierować (>) do jakiegoś pliku. Podam może kilka przykładów użycia tego programu. Zrzucenie zawartości bazy baza1 do pliku baza1_bkp.sql:

```
# mysqldump -u mysql --databases baza1 > baza1_bkp.sql -p
```

Stworzenie kopii zapasowej struktury bazy (bez danych) baza2 i zapisanie jej w pliku baza2_str.sql:

```
# mysqldump -u mysql --databases --no-data baza2 > baza2_str.sql -p
```

Niezwykle przydatną opcją jest --opt omówiona wcześniej. Polecam jej stosowanie do wykonywania kopii całej bazy, włącznie ze strukturą tabel i samymi danymi. Aby stworzyć pełną kopię zapasową bazy baza1 i zapisać ją w pliku baza1_kopia.sql:

```
# mysqldump -u mysql --opt baza1 > baza1_kopia.sql -p
```

Przywracanie baz wykonanych poleceniem mysqldump wygląda tak:

```
# mysql -u mysql baza1 < baza1_kopia.sql -p
```

Inna metoda (w sumie różniąca się zapisem):

```
# mysql -u mysql -e 'source /sciezka_do_pliku/baza1_kopia.sql' baza1 -p
```

Przykład pierwszej sesji z mysql:

```
volt% mysql -p -u ziutek
Enter password: gienek
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 40515 to server version: 3.23.36-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> set password for ziutek=password('legia01');
```

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

mysql> quit
Bye
volt% mysql -p -u ziutek
Enter password: legia01
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 40520 to server version: 3.23.36-log

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql> use test_ziutek;
Database changed
mysql> create table my_test ( imie varchar(25) );
Query OK, 0 rows affected (0.00 sec)

mysql> select * from my_test;
Empty set (0.00 sec)

mysql> insert into my_test (imie) values ('Franek');
Query OK, 1 row affected (0.00 sec)

mysql> select * from my_test;
+--------+
| imie   |
+--------+
| Franek |
+--------+
1 row in set (0.00 sec)

mysql>
```

W razie pytań proszę kontaktować się z: szmurlor@iem.pw.edu.pl
Przykład połączenia z php

```
<html>
<HEAD>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-2">
<meta name="Author" content="Robert Szmurło">
<title>Przykładowa strona prezentujaca połaczenie z PHP do mysql'a</title>
</HEAD>
<body>
<h1>Zawartość tabeli my_test w bazie danych Ziutka:</h1>
<?php
  $link = mysql_connect("localhost:3306", "ziutek", "legia01")
        or die ("Could not connect");
  mysql_select_db ("test_ziutek")
        or die ("Could not select database");

  $query = "SELECT * from my_test";
```

```
  $i = 1;
  $result = mysql_query($query) or die ("Insert Query failt");
  while($line = mysql_fetch_array($result)){
    $imie = $line["imie"];
    print "$i: $imie<br>";
    $i = $i + 1;
  }


?>
</body>
</html>
```

Snort + MySQL + phpMyAdmin + ACID
Najpierw zacznę, że opis instalacaji dotyczy serwerka z postawionym apaczem, php i localnym userem marcin, czyli w tym przypadku /home/marcin i z niego będziemy kompilować itp. Należy również pamiętać, że wszystkie polecenia które poprzedza znak:
$ wykomuje user (marcin)
# wykonuje root
Będąc w swoim $HOME zaczniemy do utworzenia nowego katalogu żeby nie zaśmiecić sobie głównego $HOME programami które będziemy ściągać.

```
$ mkdir kombajn
$ cd kombajn
```

Zaczniemy od zassania mysql'a i kolejno rozpakowania go i skompilowania. Opcja przy konfiguracji --with-unix-socket-path=/var/run/mysqld/mysqld.sock określa nam położenie socketu mysql'a. A dlaczego w ten sposób? Dlatego, ponieważ snort wykorzystując mysql'a właśnie w tym miejsu szuka socket'u mysql'a

NOTE:
Sprawdz ustawienia swojego php.ini odnośnie socket'u mysql'a. Proponuje ustawić go na:

```
mysql.default_socket = "/var/run/mysqld/mysqld.sock"
```

--- mysql ---

```
$ wget ftp://sunsite.icm.edu.pl/pub/unix/mysql/Downloads/MySQL-4.1/mysql-4.1.9.tar.gz
$ tar zxvf mysql-4.1.9.tar.gz
$ cd mysql-4.1.9
$ ./configure --with-unix-socket-path=/var/run/mysqld/mysqld.sock
$ make
# make install
# cp support-files/my-medium.cnf /etc/my.cnf
```

$ cd ..

I postępujemy zgodnie z dokumentacją. Jeżeli dostaniesz mysql_install_db command not found podaj pełną ścieżkę /usr/local/bin/mysql_install_db

mysql_install_db --user=mysql

Jeżleli nie masz usera i grupy mysql to dodaj

adduser mysql
groupadd mysql

Kolejny krok to utworzenie katalogu, pod który kompilowaliśmy socket i nadanie mu odpowiednich praw

mkdir /var/run/mysqld
chown mysql.mysql /var/run/mysqld/

Skoro mam już mysqla to warto zaopatrzyć się w phpMyAdmin'a Czasami się przydaje gdy chcesz coś szybko zrobić.
Zakładam, że masz skonfigurowanego swojego apacza i dodanie kolejnego virtualhosta nie jest dla ciebie problemem. Proponuje zrobić virtualhost'a na /home/marcin/kombajn/ utworzyć tam plik .htaccess o zawartości Options Indexes. Warto dać to też po SSL'u i dostęp tylko z LAN'u

--- phpMyAdmin ---

$ cd /home/marcin/kombajn
$ wget http://aleron.dl.sourceforge.net/sourceforge/phpmyadmin/phpMyAdmin-2.6.0-pl3.tar.bz2
$ bzip2 -dc phpMyAdmin-2.6.0-pl3.tar.bz2 |tar xf -
$ cd phpMyAdmin-2.6.0-pl3

I tutaj musimy troszeczkę podłubać w pliku config.inc.php. Edytujemy go i szukamy lini $cfg['PmaAbsoluteUri'] = ''; i zastępujemy ją adresem pod jakim będzie widniał nasz phpadmin

$cfg['PmaAbsoluteUri'] = 'http://xydomena.pl/phpadmin';

Następnie zmieniamy $cfg['Servers'][$i]['auth_type'] = 'config'; na

$cfg['Servers'][$i]['auth_type'] = 'cookie';

Koleny krok to wpisanie jakiś pierdół w $cfg['blowfish_secret'] = '';

$cfg['blowfish_secret'] = 'fqn9rqvqwerg54iysekhsdfbsfqw(&3r-I)454';

I spróbujemy się zalogować

http://xydomena.pl/phpadmin login root, bez hasła, jeżeli się zalogujesz pierwszym krokiem będzie zmiana hasła i wylogowanie sie z phpadmina.
Przy kolejnej próbie zalogowania dostaniesz errora:

"Client does not support authentication protocol requested by server; consider upgrading MySQL client"

Czyli przechodzimy na konsle i klepiemy

```
# mysql -u root -p
Enter password:
```

I podajemy hasło które przed chwilą ustawiliśmy w phpadminie, następnie klepiemy magiczną regułkę:

```
mysql> set password for 'root'@'localhost' = OLD_PASSWORD('tajnehaslo');
mysql>exit
```

I ponowna próba zalogowania się przez phpadmian zakończy sie powodzeniem ;)

Teraz pora na utworzenie 2-uch baz dla snorta, jeżeli wolisz klepać z kosoli to mozesz to zrobić w następujący sposób:

```
# mysql -u root -p
Enter password:
mysql> create database snort_log;
mysql> create database snort_archive;
mysql> exit
```

Narazie to wystarczy.

--- SNORT ---

Zasysamy, konfigurujemy dając opcje pod mysql'a

```
$ cd /home/marcin/kombajn/
$ wget http://www.snort.org/dl/snort-2.3.0.tar.gz
$ tar zxvf snort-2.3.0.tar.gz
$ cd snort-2.3.0
$./configure --with-mysql --with-mysql-include=/usr/local/include/mysql/ --with-mysql-lib=/usr/local/lib/mysql
```

hmm.. i tu chwilkę się zastanaowimy ;) jeżli konfiguracaj przeszła ci bez problem znaczy to, że masz wszyskie wymagane pakiety. Jeżeli natomiast czegoś ci brakuje to patrz na czym się wywali ale prawdopodobnie będzie to libpcap0 libpcap-dev lub libpcre3 , libpcre3-dev

```
$ make
# make install
# mkdir /etc/snort
```

```
# cp etc/* /etc/snort/
# cp -r rules schemas /etc/snort/
```

Spróbujemy odpalić snorta

```
# snort -c /etc/snort/snort.conf
```

Jeżeli przy próbie odpalenia snorta wywali nam błąd w stylu:

snort: error while loading shared libraries: libmysqlclient.so.14: cannot open shared object file: No such file or directory

Należy wyedytować /etc/ld.so.conf i dopisać linię:

/usr/local/lib/mysql

I oczywiście

```
# ldconfig
```

Pewnie będzie kolejny error ;)

ERROR: Unable to open rules file: ../rules/local.rules or /etc/snort/../rules/local.rules Fatal Error, Quitting..

W tym przypadku edytujemy /etc/snort/snort.conf i szukamy vat RULE_PATH, zmieniamy go na :

var RULE_PATH /etc/snort/rules

I kolejny error :P

ERROR: [!] ERROR: Can not get write access to logging directory "/var/log/snort". (directory doesn't exist or permissions are set incorrectly
or it is not a directory at all)

Czyli:

```
# mkdir /var/log/snort
# snort -c /etc/snort/snort.conf
```

I działa :P

Teraz zajmiemy się konfiguracją snorta i mysql'a. Zaczniemy od mysql'a i dadania usera snort, a następnie dodamy table do baz.

```
# mysql -u root -p
Enter password:
```

```
mysql> GRANT USAGE ON * . * TO "snort"@"localhost" IDENTIFIED BY "tajnos"
WITH MAX_QUERIES_PER_HOUR 0 MAX_CONNECTIONS_PER_HOUR 0
MAX_UPDATES_PER_HOUR 0 ;
mysql> GRANT ALL PRIVILEGES ON `snort\_archive` . * TO "snort"@"localhost"
WITH GRANT OPTION ;
mysql> GRANT ALL PRIVILEGES ON `snort\_log` . * TO "snort"@"localhost" WITH
GRANT OPTION ;
```

Następnie powinniśmy dodać tabele do obu baz ale nim to zrobimy to zassamy ACID żeby móc stworzyć tablele dla baz snort_log i snort_log. Czyli

```
## --- ACID --- ##

$ cd /home/marcin/kombajn
$ wget http://acidlab.sourceforge.net/acid-0.9.6b23.tar.gz
$ tar zxvf acid-0.9.6b23.tar.gz
```

I dodajemy table do baz

```
# mysql -D snort_log -u snort -p < /home/marcin/kombajn/snort-
2.3.0/schemas/create_mysql
# mysql -D snort_log -u snort -p <
/home/marcin/kombajn/acid/create_acid_tbls_mysql.sql
# mysql -D snort_archive -u snort -p < /home/marcin/kombajn/snort-
2.3.0/schemas/create_mysql
# mysql -D snort_archive -u snort -p <
/home/marcin/kombajn/acid/create_acid_tbls_mysql.sql
```

Czyli mysql'a mamy gotowego do pracy więc pora dokończyć konfigracje snorta. W tym celu edytujemy /etc/snort/snort.conf i szukamy Configure output plugins i lini

```
# output database: log, oracle, dbname=snort user=snort password=test
```

I dopisujemy

```
output database: alert, mysql, user=snort password=tajnos dbname=snort_log
host=localhost
```

Kolej na ACID i edycję pliku /home/marcin/kombajn/acid/acid_conf.php w którym dodajemy parametry dla połączenia się z bazami.

```
$alert_dbname = "snort_log";
$alert_host = "localhost";
$alert_port = "3306";
$alert_user = "snort";
$alert_password = "tajnos";

/* Archive DB connection parameters */
$archive_dbname = "snort_archive";
$archive_host = "localhost";
```

$archive_port = "3306";
$archive_user = "snort";
$archive_password = "tajnos";

Jak zrobiłeś wcześniej virtualhosta to klepnij sobie w przeglądarkę acid.xydomena.pl
I pewnie dostaniesz kolejnego errora od ACID :P

Error loading the DB Abstraction library: from "/adodb.inc.php"
Check the DB abstraction library variable $DBlib_path in acid_conf.php
The underlying database library currently used is ADODB, that can be downloaded at http://php.weblogs.com/adodb;

Czyli robimy dokładnie to co napisane

$ cd /home/marcin/kombajn
$ wget http://heanet.dl.sourceforge.net/sourceforge/adodb/adodb460.tgz
$ tar zxvf adodb460.tgz

I w pliku //home/marcin/kombajn/acid/acid_conf.php podajemy lokalizecje adodb.inc.php czyli szukamy DBlib_path i dopisujemy:

$DBlib_path = "/home/marcin/kombajn/adodb/";

Nie pozostaje nam nic innego jak odpalić mysqla i snorta ;)

Możemy napotkać się jeszcz na problem z połączeniem do bazy mysq za pomocą ACID otrzymując na stronie komunikat:

Client does not support authentication protocol requested by server; consider upgrading MySQL client

Należy wtedy zalgować sie do mysqla na usera snort i zmieniec haslo:

$ mysql -u snort -p
Enter password:
mysql> set password for 'snort'@'localhost' = old_password ('agentos');

I odświeżyć stronę ;)
Na tym koniec, ponieważ konfiguracja samego snorta +ACID to inna bajka

Tworzenie kopii zapasowych w Mysql'u

Marcin Federowicz () • 2004-12-30 18:52:59 • wersja do wydruku

Czyli jak mieć więcej włosów na głowie po utracie danych w bazie :)

Jeśli kiedykolwiek z powodu utraty ważnych danych wyrywałeś sobie włosy, bez wątpienia określenie 'kopia zapasowa' - potocznie backup nabiera specjalnego znaczenia w twoim życiu. Bazy danych mają spore możliwości katalogowania danych, ale biorąc pod uwagę ilość informacji powierzanych do przechowywania przez serwery MySql'a skutki niewłaściwego użycia komendy DROP DATABASE, nieoczekiwanej awarii systemu, ręcznej edycji struktury tabeli zakończonej niepowodzeniem są katastroficzne i nie do przywrócenia -- chyba że posiadamy kopię zapasową z której możemy odtworzyć utracone dane.
MySQL posiada wbudowane narzędzie nazywające się mysqldump które służy do tworzenia niezależnych od platformy plików tekstowych które zawierają pełną kopię tego co znajdowało się w bazie, która przez przypadek została utracona.
Aby utworzyć kopię zapasową bazy db_przykładowa, należy wykonać poniższą komendę:

mysqldump -u <uzytkownik> -p <haslo> -h <nazwahosta> db_przykladowa > /usr/backups/mysql/db_przykladowa.2004-12-27

MySQL użyje username, password, oraz host aby sprawdzić czy posiadamy uprawnienia dostępu do bazy. Po udanej autentykacji, przekierowujemy cały potok wyjścia wytworzony przez polecenie mysqldump do wskazanego przez nas pliku.
Dobrą praktyką jest dodawanie do nazwy plików z backupem daty. W niektórych przypadkach gdy kopie robione są kilka razy na dzień przydatne jest również wstawianie informacji z czasem ich utworzenia.
Dopisek: W dalszych przykładach dla łatwiejszego zobrazowania działania polecenia mysqldump pominę używanie opcji -u, -p, oraz -h, ty powinieneś ich używać w przeciwnym wypadku MySQL będzie się skarżył że ich nie podałeś :)
Jeśli mamy do czynienia z bazą w której niektóre z tabel są używane znacznie częściej niż inne wtedy mysqldump pozwala nam wykonać tylko ich kopię:

mysqldump db_przykladowa artykuły komentarze linki > /usr/backups/mysql/db_przykladowa.art_kom_lin.2004-12-27

Powyższe polecenie zrobi zrzut tabel artykuly, komentarze, oraz linki naszej bazy. Oczywiście plik wyjściowy nazwaliśmy tak aby było wiadomo co w nim sie znajduje. Ta funkcja jest pomocna gdy mamy do czynienia z dużymi bazami (np. systemy zarządzania treścią), tam również spotykane jest kilka typów tabel które razem mogą być pogrupowane w jedną sekcję dzięki czemu uzyskamy mniejsze łatwiejsze do zarządzania pliki z kopiami bezpieczeństwa. Przykładowo nazwa sekcji może być wykorzystana jak poniżej:

mysqldump db_przykladowa gracz punkty sezon > /usr/backups/mysql/db_przykladowa.graczinfo.2004-12-27
Aby zaoszczędzić na powierzchni dyskowej możemy w locie kompresować pliki za pomocą gzip'a:
mysqldump db_przykladowa | gzip > /usr/backups/mysql/db_przykladowa.2004-12-27.gz
Dumpy po sieci
Przechowywanie kopii zapasowych na tej samej maszynie co oryginalna baza jest jak pokazywanie swojej swej sekretnej broni swojemu największemu wrogowi.
Jednym z najprostszych sposobów przechowywania backupów na innej maszynie w

sieci jest przekopiowanie ich ręcznie. Ludzie żyjemy chyba w erze cyfrowej -- od czego jest automatyzacja !
W tym przykładzie posłużymy sie dwoma maszynami główna będzie posiadała adres IP 192.168.1.11, natomiast maszyna na którą będziemy kopiować to 192.168.1.22.
No dobrze, więc zacznijmy od tego iż dobrze jest stworzyć oddzielną partycję (w tym przykładzie będziemy ją nazywać 'archive') na zdalnej maszynie i zamontować ją. W tym wypadku nawet gdy system na zdalnej maszynie będzie nie zdatny do użycia to możemy zainstalować nowy system bez obawy o pliki naszych kopii zapasowe.
Do wykonywania kopii zapasowych na zdalnej maszynie linuksowej musisz posiadać skonfigurowaną obsługę NFS (Network File System). Lektura 'Understanding NFS' [1] oraz 'Implementing NFS' [2] będzie pomocna aby uruchomić NFS. Gdy do kopii używamy natomiast maszyny z systemem z Redmond na pokładzie, będziemy potrzebować zainstalowaną i skonfigurowaną usługę Samby. Jeśli mamy z tym problemy pomocne będą dokumenty znajdujące się tutaj [3]
Załóżmy że posiadasz NFS na swojej zdalnej maszynie, otwórz więc /etc/exports w swoim ulubionym edytorze tekstowym i dodaj tą linijkę:
/archive 192.168.1.11 (rw, no_root_squash)
W ten oto sposób informujemy NFS aby współdzielił katalog /archive wraz z systemem który posiada bazę. Katalog ten na maszynie lokalnej jak i zdalnej posiada prawa do odczytu i do zapisu oraz użytkownik łączący się z maszyny bazodanowej posiada uprawnienia roota na maszynie zdalnej. Po zachowaniu efektów naszej pracy wydajemy następującą komendę:
exportfs -a -r
Dzięki czemu przeeksportujemy wszystkie katalogi tak jak to mamy zapisane w pliku /etc/exports. Po więcej informacji na temat exportfs odsyłam do man exportfs.
Po czym restartujemy usługę NFS:
/etc/rc.d/init.d/nfs start
Powyższe polecenie ustawi naszą zdalną maszynę. W katalogu /mnt na maszynie z MySQL'em tworzymy katalog backup_share za pomocą polecenia:
mkdir /mnt/backup_share
Następnie montujemy katalog archive z maszyny zdalnej w tym katalogu:
mount \u2013o soft 192.168.1.22:/archive /mnt/backup_share
Mamy więc podmontowany folder, czas więc na wykonanie kopii zapasowej. Wracając do naszej bazy będzie to wyglądało mniej więcej tak:
mysqldump db_przykladowa > /mnt/backup_share/db_przykladowa.2004-12-27
Na maszynie wyposażonej w system Windows ze skonfigurowaną usługą Samby, musimy utworzyć katalog archive oraz udostępnić go z prawami do odczytu i zapisu. Następnie tworzymy katalog backup_share w /mnt na naszym serwerze bazodanowym tak jak w poprzedniej sekcji i montujemy go :
mount -t smbfs -o username=<uzytkownik>, password=<haslo> //192.168.1.22/archive /mnt/backup_share
Oczywiście należy zastąpić <uzytkownik> oraz <haslo> informacjami niezbędnymi do właściwego dostępu do naszej udostępnianej zdalnej maszyny. W końcu tworzymy kopie zapasową (db_przykladowa) w katalogu share:
mysqldump db_przykladowa > /mnt/backup_share/db_przykladowa.2004-12-27
Automatyzacja procesu
Teraz w momencie gdy wiemy już w jaki sposób tworzyć kopie baz i ich tabel oraz wiemy jak przechowywać je na zdalnych lokalizacjach nadszedł czas aby Linux pokazał swoją siłę i wykonywał te zadania za nas. Do tego celu użyjemy demona o nazwie cron. cron jest linuksowym programem który uruchamia w tle zadania w

określonych przez nas interwałach czasowych. Demon crona budzi się raz na minutę i sprawdza plik crontab czy zawiera on jakieś zadania i jeśli jakieś znajdzie to je wykonuje czyż to nie proste ?:)
Napiszemy więc sobie prosty skrypt który będzie wykonywał kopie zapasowe i do odpalania go zaprzęgniemy cron'a . Otwórzmy więc nasz ulubiony edytor tekstowy i przekopiujmy to co znajduje się poniżej do nowego pliku:

```
## Jeśli robisz kopię na maszynie linuksowej, odkomentuj linię poniżej
# mount \u2013o soft 192.168.1.22:/archive /mnt/backup_share

## W przeciwnym wypadku jeśli masz maszynkę z Windowsami, odkomentuj poniższą linię i uzupełnij username oraz password
# mount -t smbfs -o username=<uzytkownik>, password=<haslo> //192.168.1.22/archive /mnt/backup_share

## Na koniec komenda $(date +%F) poda aktualne informacje na temat daty
mysqldump -u <uzytkownik> -p <haslo> -h <nazwahosta> db_przykladowa > /mnt/backup_share/db_przykladowa.$(date +%F)

#odmontowujemy system plików
umount /mnt/backup_share
```

Zapisujemy efekty naszej pracy jako db_przykladowa_backup.sh
Należy zwrócić uwagę na to iż jest to skrypt napisany szybko i trochę 'na kolanie'. W momencie konkretnej implementacji powinniśmy sprawdzić czy zdalna partycja jest zamontowana, czy plik z naszą kopią bezpiecznie został przesłany do zdalnego systemu i wiele innych rzeczy.
Nadajemy prawo do wykonywania dla naszego skryptu:
chmod +x ./db_przykladowa_backup.sh
Teraz przyszła kolej na cron'a. Jeśli należymy do grupy ludzi leniwych to możemy wykorzystać jedną z 4 opcji do uruchomienia naszego skryptu: raz na godzinę, raz na dzień, tydzień, oraz miesiąc. W zależności od tego którą opcję wybierzemy musimy przekopiować nasz skrypt do właściwego katalogu /etc/cron.przedzialczasu, gdzie przedzialczasu jest interwałem czasowym co który chcemy wykonywać kopie zapasowe. Ja przekopiowałem skrypt do /etc/cron.daily gdyż chce aby backupy były robione raz dziennie. Gdy będziesz gotowy ponownie uruchom demon cron'a :
/etc/rc.d/init.d/crond restart
No i to by było na tyle -- twój backup będzie wykonywany o 4:02 a.m. każdego dnia, dlaczego tak ano dlatego iż jest to domyślne ustawienie dla katalogu /etc/daily które znajduje się w pliku /etc/crontab. Oczywiście to nie oznacza iż nie możemy ustawić własnej godziny czy innego interwału czasowego po więcej informacji odsyłam na strony Wikipedii dla crontab'a [4].
Używanie dumpów
Pliki backup'owe generowane przez mysqldump'a są zwykłymi plikami tekstowymi w których znajduje się wiele poleceń CREATE TABLE oraz INSERT które przywracają bazę danych. Oto zawartości przykładowego pliku:

```
-- MySQL dump 8.23
--
-- Host: localhost Database: geeklog
---------------------------------------------------------
-- Server version 3.23.58
```

```
--
-- Table structure for table `gl_commentcodes`
--

CREATE TABLE gl_commentcodes (
code tinyint(4) NOT NULL default '0',
name varchar(32) default NULL,
PRIMARY KEY (code)
) TYPE=MyISAM;

--
-- Dumping data for table `gl_commentcodes`
--


INSERT INTO gl_commentcodes VALUES (0,'Comments Enabled');
INSERT INTO gl_commentcodes VALUES (-1,'Comments Disabled');
```

Abyśmy mogli przywrócić bazę z tego przykładowego pliku, najpierw musimy stworzyć pustą bazę, Aby wypełnić tą bazę tabelkami a tabelki danymi musimy wybrać właściwy plik z kopią bazy:

mysql -u <uzytkownik> -p <haslo> -h <nazwahosta> db_przykladowa < /mnt/backup_share/db_przykladowa.2004-12-27

W tym momencie musimy podać niezbędne do autoryzacji informacje i gotowe! Teraz nasza baza jest taka jaką widzieliśmy ją ostatnim razem.

Podsumowanie

mysqldump jest niewątpliwie niezmiernie ważnym narzędziem dla administratorów serwerów MySQL, oraz wspaniałym narzędziem utrzymania integralności i 100% dostępności krytycznych danych przetwarzanych przez nasze serwery danych.

Autorem oryginalnego artykułu jest:

Pobierz spakowaną witrynę infolinux.pl
Jeżeli pragniesz poznać Linuksa Mandrake (obecnie Mandriva), to... dobrze trafiłeś. Witryna została zauważona przez twórców magazynu KOMPUTER ŚWIAT 5/2004(141) str. 46 poprzez umieszczenie linku oraz magazynu CHIP 4/2004 (str.114) poprzez umieszczenie artykułu opisującego ten serwis internetowy. Jak na hobbystyczną stronę o Linuksie to miłe ;) Acha... na stronie mandrakelinux.pl/informacje podano też link z opisem cytuję "duży zbiór praktycznych informacji o Mandrake" (mam ją w swoim archiwum - klub.chip.pl/twarogal).
Zapraszam do zadawania pytań na FORUM oraz mailem. Chętnie udzielę (bezpłatnie) odpowiedzi. Oficjalne ceny za jedną poradę na stronie MandrakeSoftPL (mam ją w moim archiwum z dnia 2.05.2004) wahają się od 20 do 350 zł.
KSIĘGA GOŚCI oraz STATYSTYKI ODWIEDZIN

Autor artykułu Kamil Twarogal

*

Spis treści:

*

Baza danych plus PHP (oparte na serwerze www np. Apache) to narzędzia umożliwiające zbieranie i przetwarzanie danych. Na dzień dzisiejszy mamy wersje programów zarówno na Windows, jak i Linuksa oraz (co oczywiste) na Uniksy. Pod Windows problemem (dla nowicjuszy) może być konfiguracja poszczególnych programów (MySQL, Apache z modułem PHP), by ładnie współpracowały. Rozwiązaniem wydaje się być projekt KRASNAL, zawierający instalkę z już skonfigurowanymi programami. Działa na starym Win9x, WinNT i najnowszym Windows. Najmniej stabilnie pracuje na Win9x.

Uwaga: komputer na którym instalujesz MySQL musi posiadać kartę sieciową, by na niej mógł pracować odpowiednio skonfigurowany interfejs sieciowy. Komputery pracujące w sieci (zarówno od dostawcy Internetu jak i pracujące w domowej sieci) mają automatycznie (np. poprzez DHCP) ustawione parametry typu: adres IP, maskę sieci, nazwę komputera itd. Wówczas nie powinnimy mieć problemów z pracą MySQL. Gorzej, gdy mamy komputer odłączony od sieci (karta sieciowa jest w komputerze niezbędna). Rozwiązanie problemu na przykładzie Win98SE umieściłem w tej notatce. Problem dotyczy też Linuksa - wówczas trzeba utworzyć nowy interfejs sieciowy z wymyślonym, fikcyjnym adresem na eth0 (karta sieciowa). Adres IP:

192.168.000.002, Maska 255.255.255.000, Broadcast 192.168.000.255, ewentualna Brama: 192.168.000.001 (w Mandrake/Mandriwa pomoże nam w tym druid drakgw czyli UTWÓRZ NOWY INTERFEJS SIECIOWY).

*

Niniejszy artykuł pisałem opierając się na Linuksie Mandrake 10.1 (obecnie Mandriva), co nie powinno mieć większego znaczenia, gdyż MySQL można instalować na każdym systemie operacyjnym: Windows, Linux, Unix. Przyjąłem, że dane wpisujemy i odczytujemy pracując na kliencie MySQL (wydając odpowiednie polecenia w powłoce tekstowej). Klient i serwer są na tym samym komputerze (to ważne), czyli logujemy się na klienta MySQL klepiąc po klawiaturze podłączonej do serwera (ew. łączymy się tunelem ssh).

Odradzam jednoczesną naukę PHP i MySQL. Polecam w pierwszej kolejności zaznajomić się z teorią baz danych (tu MySQL).
Uwaga: user root używany do obsługi serwera MySQL nie ma nic wspólnego z systemowym userem root poznanym podczas prac na Linuksie - ma inne hasło i inne zadania. Zbieżność nazw (moim zdaniem niezbyt udana) jest związana z określeniem usera o najwyższych prawach dostępu. Na niniejszej stronie mamy więc do czynienia z systemowym userem root oraz z userem root od MySQL.
Zwróć uwagę na plik z historią poleceń .mysql_history , który jest umieszczony w katalogu domowym usera czyli np. /home/antek lub /root . Może tam znaleźć się hasło jako pozostałość po niewłaściwym (ale teoretycznie możliwym) logowaniu.
Przykład logowania niewłaściwego:
mysql -u root -p haslo;
oraz właściwego:
mysql -u root -p (ENTER, haslo). Problem rozwinę poniżej.
Średnik ; kończący wpis oznacza (w MySQL) koniec polecenia i jest bardzo ważnym elementem! Bez niego zlecenie nie zadziała. Uwaga na znaczek ' (na klawiszu ze znakiem cudzysłowia).
W Linuksie Mandrake/Mandriva serwer MySQL uruchamiamy odwołując się do skryptu startowego: /etc/rc.d/init.d/mysqld (z parametrami np. start, stop, restart). Klienta uruchamiamy poprzez plik /etc/rc.d/init.d/mysql (dodając parametry np. -u -p -h). Do tematu powrócę poniżej.

*

Uruchomienie serwera MySQL

Podczas instalacji Linuksa Mandrake/Mandriva wybierz serwer MySQL (zostaną automatycznie dodane inne pakiety np. klient mysql). Nie instaluj (na razie modułów PHP, CGI, itp). Przyjmuję, że interfejsy sieciowe na komputerze są skonfigurowane (patrz notatka na początku tej strony). Zaloguj się w systemie jako systemowy user root i uruchom serwer MySQL zleceniem:
/etc/rc.d/init.d/mysqld start (lub stop, restart) (wciśnij klawisz ENTER)
Uwaga: serwer MySQL można automatycznie włączać podczas startu systemu zaznaczając właściwą opcję po wydaniu zlecenia: drakxservices. Więcej o autostarcie tutaj.

*

Uruchomienie klienta MySQL
(na tym samym komputerze co zainstalowano serwer MySQL)

Jako systemowy zwykły user (nie systemowy root!) np. antek wykonaj zlecenie:
mysql (wciśnij klawisz ENTER)
lub jeżeli nie wyjdzie, to:
cd /etc/rc.d/init.d/ (wciśnij klawisz ENTER)
mysql (wciśnij klawisz ENTER)
Powyższe uruchomienie klienta MySQL uda się wyłącznie, gdy user root MySQL nie ma jeszcze ustawionego hasła, a tak jest np. tuż po zakończeniu instalacji systemu.
Zdołałeś zalogować się do serwera MySQL. Wylistuj dostępne bazy danych (powinno być ich kilka) zleceniem:
mysql>show databases; (wciśnij klawisz ENTER)
Zwróć uwagę na średnik na końcu zlecenia!
Jeżeli zobaczyłeś prostą tabelkę, to znaczy, że możesz zacząć naukę baz danych MySQL.
Pracę klienta MySQL zakończysz zleceniem exit.
Jeżeli zalogujesz się do serwera MySQL, to przez cały czas prac z lewej strony będzie widoczny napis mysql - stąd na niniejszej stronie umieściłem przed każdym zleceniem wpis mysql>

*

LOGOWANIE

Przyjmuję, że serwer MySQL automatycznie włącza się po starcie systemu. Działasz na jednym komputerze jako systemowy user np. antek.
Ze względów bezpieczeństwa w pierwszej kolejności ustaw userowi root (tego od MySQL) jakieś hasło. Wykonaj:
cd /etc/rc.d/init.d/ ENTER
mysql ENTER
Jesteś zalogowany jako root (ten od MySQL). Ustaw hasło rootowi (temu od MySQL) zleceniem:
mysql>set password = password('jakieshaselko'); ENTER
Uwaga na średnik ; na końcu zlecenia. Ważne jest położenie (lub brak) spacji w zleceniu. Znaczek ' znajdziesz na klawiszu ze znakiem cudzysłowia.
Wyloguj się zleceniem exit ENTER
Zaloguj się na nowo, ale tym razem rozbudowanym zleceniem:
mysql -u root -p (ENTER i podaj hasło roota od MySQL)

Nie wolno wpisywać hasła bezpośrednio w polecenie!!! np. mysql -u root -p hasło; , gdyż zostanie odnotowane w pliku ~/mysql_history i będzie można je wykraść.
Gdy znasz nazwę dazy danych nad którą będziesz pracować, to możesz od razu zalogować się i wskazać domyślną bazę danych:
mysql -u root -p nazwabazydanych (ENTER i podaj hasło roota)
Zwróć uwagę, że na końcu zlecenia nie ma średnika ;

Uwaga: jeżeli podczas logowania nie podasz domyślnej bazy danych, to zrób to po zalogowaniu zleceniem:
mysql>use nazwabazydanych; ENTER
Uwaga na średnik ; na końcu zlecenia.
W zależności od konfiguracji serwera MySQL czasami będziesz musiał podczas logowania podać adres hosta (komputera). Jeżeli serwer i klient MySQL są na tej samej maszynie, to wypróbuj poniższe zlecenia. Ponieważ na razie masz tylko usera root (od MySQL), to oczywistym jest, że nie możesz logować się na zwykłego usera (od MySQL) - w naszym przykładzie jest to kaziu. Zwróć uwagę, że na końcu zlecenia nie ma średnika.
mysql -u kaziu -p -h localhost ENTER
lub
mysql -u kaziu -p -h 192.168.1.1 ENTER
lub adres IP i nr portu (tu 3306):
mysql -u kaziu -p -h 217.96.171.101:3306 ENTER
W ostateczności wypróbuj zlecenie:
mysql kaziu@localhost -p ENTER
Parametr -u oznacza, że po spacji podamy nazwę usera (np. root), parametr -p oznacza, że user ma hasło, -h oznacza, że po spacji podamy adres (lub nazwę) hosta. Wylogujesz się zleceniem exit

Stosuj zasadę, że uruchamiasz klienta mysql z komputera na którym działa serwer mysqld. Jeżeli musisz to zrobić z odległego komputera, to uruchom szyfrowany tunel ssh i po przejęciu shella zwykłego usera odpal na linuksowym serwerze klienta mysql. Osobiście próbowałem zdjąć zabezpieczenia w moim Mandrake/Mandriva i logować się na serwer MySQL z klienta w domowej sieci. Dotarłem do odpowiednich opcji w konfigu MySQL, udostępniłem na firewallu. Straciłem kilka dni i nic. Po prostu Mandrake/Mandriva ma szereg dodatkowych zabezpieczeń (co jest dobre na wypadek włamu), których nie publikuje w darmowych poradach. Z drugiej strony może i dobrze, że są zabezpieczenia. Przecież MySQL to jedynie silnik bazodanowy. Do wprowadzania danych lub wyprowadzania kwerend należy używać narzędzi typu php.

*

TWORZENIE BAZ DANYCH

Baza danych zbudowana jest z nazwy oraz tabelek, stąd w pierwszej kolejności utworzymy nazwę dla naszej bazy danych, a dopiero potem tabelki podporządkowane tylko tej bazie danych. Pamiętaj, by w nazwach baz danych (i tabelkach) nie wprowadzać polskich liter typu śćźżąóń oraz spacji (zamiast niej można dać dolną kreskę _). Zwróć uwagę na średnik ; na końcu zlecenia. Jesteś zalogowany jako root (ten od MySQL).
Wylistuj bazy danych (ich nazwy) zleceniem:
mysql>show databases; ENTER
Dodaj nową nazwę bazy danych (utwórz nową bazę danych) zleceniem:
mysql>create database nazwabazydanych; ENTER
Wylistuj ponownie bazy danych, a następnie usuń bazę danych zleceniem:
mysql>drop database nazwabazydanych; ENTER

*

Uwaga: bazy danych oraz tabelki wykonuj pod rootem (tym od MySQL). Dopiero na koniec prac utwórz zwykłego usera (tego od MySQL), nadaj mu hasło, odpowiednie prawa (np. do zapisu, odczytu itd.) - opis tworzenia usera jest poniżej. Od tego momentu (czyli wprowadzania danych i zwykłej pracy w bazie danych) nie należy używać roota (tego od MySQL), działając wyłącznie pod zwykłym userem (tym od MySQL).

*

TWORZENIE TABEL

Będąc zalogowanym jako zwykły systemowy user np. antek zaloguj się jako jako root (ten od MySQL) do serwera MySQL - opis powyżej. Następnie wykonaj zlecenia (zwróć uwagę na średnik ; na końcu zleceń):
mysql>create database klienci; ENTER
mysql>use klienci; ENTER
Utwórz tabelkę nazwiska w bazie danych klienci jednym zleceniem:
mysql>create table nazwiska (ENTER
mysql>id int not null auto_increment, ENTER
mysql>opis text, ENTER
mysql>name varchar(25), ENTER
mysql>primary key(id)); ENTER
Zwróć uwagę na ustawienie nawiasów (), przecinków , oraz położenie spacji.
Została utworzona w bazie danych klienci tabela nazwiska, a w niej trzy kolumny: numeracja id rekordów (z opcjami: int - o tym za chwilę, not null czyli "nigdy puste", auto_increment czyli utomatycznie nadawana numeracja +1), opis (z ustawionym typem danych text), name (z ustawionym typem danych varchar 25 znaków). Klucz ustawiono na kolumnę id.

Zobacz, czy tabelka istnieje zleceniem:
mysql>show tables; ENTER
Wyświetl strukturę (czyli kolumny) tabeli nazwiska:
mysql>describe nazwiska; ENTER
Zlecenie select wyświetla dane. Na razie mamy nową, pustą tabelę, więc nie ma w niej danych.
Usuń tabelkę nazwiska zleceniem:
mysql>drop table nazwiska; ENTER

*

Trochę teorii o bazach danych i tabelkach

Tabelki możemy teoretycznie budować w sposób dowolny, zaznaczając jedynie, by wpisy były widziane jako literki-cyferki (do tekstu) lub liczby (do późniejszych działań matematycznych). Oczywiście zawsze musi być kolumna z numeracją rekordów czyli id (nie mylić z numerem kolejnym, ale o tym później). Ten bardzo prosty schemat ma jedną poważną wadę: mała informacja zajmie dużo miejsca. Dlaczego? Ano każda komórka w danym rekordzie ma zaplanowaną przestrzeń (maksymalną ilość

możliwych do wpisania liter-cyfr). Jeżeli wielkość maksymalna wynosi np. 65538 znaków, to prosty wpis np. Alicja zamiast 6 znaków zajmie 65538 znaków. Jeżeli natomiast zaplanujemy komórkę o typie danych np. 30 znaków, to marnotrawstwo będzie znikome. Minusem ograniczania ilości znaków jest odmowa wpisu danych przekraczających wielkość zaplanowaną, stąd koniecznym będzie zaznajomienie się z poniższymi tabelkami typów danych. Nie polecam kucia na pamięć, ale wpis do notatnika lub wydruk by się przydał.
LICZBY CAŁKOWITE
LICZBY ZMIENNOPRZECINKOWE
DATA I CZAS
TEKST

OPCJE DOTYCZĄCE W/W TYPÓW DANYCH

TYP

ZAKRES WARTOŚCI

auto_increment Autonumeracja +1
BinaryParametr binary wskazuje, że wpisy są traktowane jako wartości binarne i uwzględniona zostanie wielkość znaków (w innych typach wielkość znaków w polach tekstowych jest domyślnie ignorowana).
default wartosc Określa wartość domyślną narzuconą przez programistę. Przykładem ustandaryzowanej wartości domyślnej jest np. opcja not null czyli "nigdy puste".
not null Określa czy baza danych ma zaakceptować brak wpisu. W przypadku nie podania wartości not null, MySQL automatycznie zaakceptuje wpis o wartości null (nic).
primary key Bardzo ważny parametr określający kolumnę, która ma zawierać unikalny klucz (w praktyce najczęściej jest to nr id)
Zerofill Parametr zmuszający bazę danych do automatycznego uzupełniania pola liczbowego zerami do (określonej w innym parametrze) długości tego pola np. 0000021

*

Modyfikacje struktury tabel

ZMIANA NAZWY TABELI
Będąc zalogowanym jako root (ten od MySQL) ustaw domyślną bazę danych np. klienci zleceniem:
use klienci; ENTER
Wylistuj tabelki zleceniem:
show tables; ENTER
Następnie wyświetl strukturę (czyli kolumny) tabelki o nazwie np. nazwiska zleceniem:
describe nazwiska; ENTER
Zmień nazwę tabelki nazwiska na nazw zleceniem:
mysql>alter table nazwiska rename nazw; ENTER

Sprawdź wyniki prac zleceniem: show tables; ENTER

DODAWANIE NOWEJ KOLUMNY
Będąc zalogowanym jako root (ten od MySQL) ustaw domyślną bazę danych np. klienci zleceniem:
use klienci; ENTER
Wylistuj tabelki zleceniem:
show tables; ENTER
Następnie wylistuj tabelkę o nazwie np. nazw zleceniem:
describe nazw; ENTER
Będąc zalogowanym jako root (ten od MySQL) i mając ustawioną domyślną bazę danych np. klienci dodaj w tabeli nazw nową kolumnę panienskienazwisko tak, by było położone po kolumnie opis zleceniem:
mysql>alter table nazw add panienskienazwisko varchar(30) after opis; ENTER
Jeżeli nowa kolumna ma być pierwsza, to zamiast parametru after nazwakolumny daj parametr first
Sprawdź wyniki prac zleceniem: show tables; ENTER

ZMIANA NAZWY KOLUMNY
Zmień (w tabeli nazw) nazwę istniejącej kolumny panienskienazwisko na nową nazwę: pan_nazw zleceniem:
mysql>alter table nazw change panienskienazwisko pan_nazw; ENTER

ZMIANA TYPU DANYCH KOLUMNY
Zmień (w tabeli nazw) typ danych varchar(25) istniejącej kolumny pan_nazw , na varchar(60) zleceniem:
mysql>alter table nazw modify pan_nazw varchar(60); ENTER

*

TWORZENIE UŻYTKOWNIKA

Masz utworzoną pod rootem (tym od MySQL) bazę danych oraz tabelki wraz z ustalonymi typami danych (opis powyżej). Teraz trzeba utworzyć pod rootem (tym od MySQL) zwykłego użytkownika (tego od MySQL), by za jego pomocą wypełnianiać tabele w bazach danych. Jest to związane z pierwszą zasadą bezpieczeństwa: NIGDY NIE NALEŻY UŻYWAĆ KONTA ROOTA (TEGO OD MYSQL) DO WYPEŁNIANIA TABEL DANYMI!

*

Tworzenie użytkownika/usera odbywa się po zalogowaniu pod roota (tego od MySQL) omówionym już powyżej zleceniem: cd /etc/rc.d/init.d/ oraz mysql -u root -p (ewentualnie mysql -u root -p nazwabazydanych). Każdy użytkownik może mieć nadane różne prawa.

Oto lista najczęściej nadawanych praw zwykłym userom MySQL. Praktycznie nie ma potrzeby, by nadawać więcej praw do zwykłej pracy nad bazą danych, a prawo np.

do kasowania tabeli lub np. tworzenia nowej kolumny lepiej zostawić tylko rootowi (od MySQL).

select czyli prawo do odczytu, wyszukiwania i sortowania danych
insert czyli prawo do wstawiania nowych rekordów i danych
update czyli prawo do zmieniania wartości rekordów
delete czyli prawo do kasowania danych

oraz pozostałe prawa:

lock czyli blokowanie tabel
alter czyli dokonywanie zmian w strukturze tabel
create czyli tworzenie nowych baz i tabel
drop czyli usuwanie baz i tabel
grant czyli nadawanie określonych przywilejów
revoke czyli odbieranie przywilejów
repair czyli naprawianie uszkodzonych tabel
check czyli sprawdzanie tabel i aktualizacja statystyk

Tworzymy usera MySQL kaziu z hasłem ofarts3ai. Przypominam, że każde polecenie trzeba kończyć średnikiem ; . Znaczek ' jest na klawiszu ze znakiem cudzysłowia.

Tworzenie użytkownika kaziu z hałsem ofarts3ai. Posiada on nadane (grant) wszystkie (all) uprawnienia na wszystkie (*) tabele we wskazanej bazie 'baza1'
mysql>grant all on baza1.* to kaziu identified by 'ofarts3ai';
a jeżeli będą problemy dopisz @localhost do nazwy usera:
mysql>grant all on baza1.* to kaziu@localhost identified by 'ofarts3ai';

Tworzenie użytkownika kaziu z hasłem ofarts3ai. Posiada on nadane wszystkie uprawnienia na wszystkich bazach i tabelach .
mysql>grant all on * to kaziu identified by 'ofarts3ai';

Tworzenie użytkownika kaziu z hałsem ofarts3ai. Posiada on nadane wszystkie uprawnienia na tabelę 'tabela1' w bazie 'baza1'
mysql>grant all on baza1.tabela1 to kaziu identified by 'ofarts3ai';

Można też zamiast all (wszystkie) dać tylko wybrane uprawnienia np:
mysql>grant select, insert, delete, create on * to kaziu identified by 'ofarts3ai';

By dodać istniejącemu użytkownikowi kaziu uprawnienia delete, update:
mysql>grant delete, update on * to kaziu;

By usunąć uprawnienia delete, update:
mysql>revoke delete, update on * from kaziu;

Przydatnym jest, by do każdej bazy był skonfigurowany użytkownik z ograniczonymi prawami (np. tylko select). Ze względów bezpieczeństwa należy do jednej bazy danych utworzyć jednego usera z większymi uprawnieniami (select, insert, update, delete) i jednego z uprawnieniami ograniczonymi (select).

Można też stworzyć uniwersalnego usera zosia z hasłem czytajx, który będzie miał tylko prawo select (pobieranie, sortowanie danych) we wszystkich bazach:
mysql>grant select on * to zosia identified by 'czytajx';

Przykładowe zlecenie tworzące zwykłego usera azat (działającym na tym samym komputerze co zainstalowano serwer MySQL - stąd parametr @localhost), z uprawnieniami do czytania, wprowadzania, korygowania i kasowania danych w bazie danych o nazwie klienci, w tabeli nazwiska. Hasło usera: qworpa. Zlecenie powinno być wpisane w jednej linii.
mysql>grant select, insert, update, delete on klienci.nazwiska to azat@localhost identified by 'qworpa';

Druga zasada bezpieczeństwa: NIE NADAWAJ ZWYKŁEMU UŻYTKOWNIKOWI (OD MySQL) PRAW ALL (WSZYSTKICH) BEZ WYRAŹNEJ POTRZEBY, GDYŻ PRZEJĘCIE GO PRZEZ WŁAMYWACZA DA MU PEŁNIĘ PRAW NAD MySQL (BAZY DANYCH, TABELE, UŻYTKOWNICY, ROOT).

Trzecia zasada bezpieczeństwa: ZAWSZE TWÓRZ MINIMUM DWÓCH USERÓW (OD MySQL) Z RÓŻNYM STOPNIEM UPRZYWILEJOWANIA. W PRZYSZŁOŚCI SKRYPTY PHP OBSŁUGUJ USEREM O OGRANICZONYCH PRAWACH, TAK BY PRZEJĘCIE GO PRZEZ INTRUZA NIE UMOŻLIWIŁO ZNISZCZENIE BAZY. W wielu moich skryptach formularze PHP umożliwiające kasowanie zawartości baz danych są usunięte z konta www i wkładane jedynie na czas prac administracyjnych. Utrudnia to nieprawne przejęcie uprzywilejowanego usera (od MySQL).

*

Jak sprawdzić nazwy użytkowników w MySQL
oraz
jak sprawdzić prawa pojedynczego użytkownika?

W Linuksie Mandrake/Mandriva jako zwykły systemowy user np. antek zaloguj się pod roota (od MySQL) zleceniem: cd /etc/rc.d/init.d/ ENTER
mysql -u root -p ENTER i podaj hasło roota od MySQL

Teraz jako root (od MySQL) wylistuj dostępne bazy danych (powinno być ich kilka) zleceniem:
mysql>show databases;

Jest tam baza danych mysql. Wejdź do bazy danych mysql zleceniem:
mysql>use mysql;

Wylistuj tabele w bazie danych mysql
mysql>show tables; ENTER

Mamy kilka tabel (ilość może być różna zależnie od wersji serwera MySQL):
- column_priv czyli uprawnienia do kolumn
- db czyli uprawnienia do baz danych

tables_priv czyli uprawnienia do tabel
host czyli uprawnienia hostów do baz danych
user czyli opis userów (użytkowników MySQL)

Jesteśmy zainteresowani tabelką opisującą użytkowników. Tabelka ma nazwę user.
Wyświetl strukturę (czyli listę kolumn) tej tabeli zleceniem:
mysql>describe user;
Pojawią się wszystkie kolumny zawierające parametry userów: przynależność do hosta, nazwa usera, hasło, przywileje (priv czyli prawa).
Nieszczęśliwie nazwa tabeli user ma taką samą nazwę jak jedna z jej kolumn: user.
Dodatkową komplikacją jest to, że KOLUMNY mają tutaj położenie w POZIOMIE.
Przy okazji warto zapamiętać listę przywilejów: select, insert, update, delete, create, drop, reload, shutdown, process, file, grant, references, index, alter.
Parametr TYPE przy user, password informuje nas, że nazwa usera oraz hasło usera może mieć maksymalnie 16 znaków (a przynajmniej tak jest na moim komputerze).

Aby wylistować wszystkich userów MySQL i ich wybrane prawa należy w tabeli user wyedytować kolumnę user oraz po kolei kolumny, które w nazwie mają "_priv" (można też sprawdzić parametr host) za pomocą zlecenia:
mysql>select user, host, select_priv, insert_priv, delete_priv, grant_priv from user;

Aby sprawdzić, czy dany user MySQL: kaziu oraz zosia ma wybrane prawa:
mysql>select user, select_priv, insert_priv from user where user = kaziu and user = zosia;

Oczywiście w powyższym przykładzie wpisałem zapytanie dotyczące kilku praw, ale można wpisać wszystkie. W odpowiedzi uzyskasz ładny wykaz praw (literka Y-yes, literka N-no)

Aby wybrać tylko nazwy userów z tabeli user wpisz jako root (od MySQL):
mysql>select user from user;

*

Jeżeli w przyszłości nie będziesz mógł się zalogować klientem MySQL, to pamiętaj, że po korekcie w bazie danych mysql i modyfikacji tabeli np. user należy zleceniem flush privileges odświeżyć bufor serwera MySQL.

*

Jeżeli zapomnimy hasło użytkownika

Jeżeli zapomnimy hasło użytkownika (tego od MySQL) np. kaziu, to jako root (od MySQL) możemy nadać je na nowo (przyjmujemy, że użytkownik MySQL kaziu jest już utworzony):
logujemy się z pozycji zwykłego, systemowego użytkownika jako root (od MySQL) zleceniem: cd /etc/rc.d/init.d/ ENTER
oraz
mysql -u root -p licznik ENTER

następnie wykonujemy zlecenie:
mysql>grant all on licznik.* to kaziu identified by 'nowe_haslo';
Uwaga: grant all czyli WSZYSTKIE PRAWA, można zastąpić wybranymi np. grant select, insert, delete, create
Nazwa bazy danych: licznik, tabelki WSZYSTKIE (*), użytkownik kaziu, nowe hasło: nowe_haslo

*

WYPEŁNIANIE TABEL DANYMI

Powyżej pokazałem jak utworzyć bazę danych MySQL (wraz z tabelkami) oraz jak utworzyć użytkowników MySQL. Pobieranie danych załatwi nam instrukcja SELECT. Struktura tej instrukcji wygląda następująco:
SELECT co_wybrać
FROM z_której_tabeli
WHERE warunki

Czas wypełnić tabelki treścią za pomocą zwykłego użytkownika (tego od MySQL). Powinienem w tym miejscu zaprezentować PHP, czyli narzędzie idealnie nadające się do wprowadzania i odczytu danych. Niestety, pośpiech jest niewskazany i na razie trzeba poznać składnię poleceń MySQL, choćby dlatego, że PHP właśnie tych poleceń używa do pracy.

Przyjmuję, że pracujesz pod Linuksem w powłoce tekstowej. Na tym samym komputerze masz serwer MySQL i klienta MySQL. Na Linuksie jest systemowy (linuksowy) użytkownik antek oraz użytkownik MySQL o nazwie licznik3. Zaloguj się jako antek, a następnie wpisz zlecenie:
cd /etc/rc.d/init.d/ (ENTER)
mysql -u licznik3 -p (wciśnij klawisz ENTER i podaj hasło usera licznik3)
uruchomisz użytkownika licznik3 (tego od MySQL).

*

Ciąg dalszy nastąpi jak będę miał trochę czasu...

Acha... Można wpisywać ręcznie (w shellu) polecenia do MySQL oraz można te same polecenia wprowadzić do pliku i skorzystać z wpisu:
mysql -u user -p < plik_z_poleceniami
aby zobaczyć komunikaty MySQL można wpisać:
mysql -u user -p < plik_z_poleceniami > komunikaty

Przydatny jest też projekt PhpMyAdmin. Aby za jego pomocą wpisac komendy mysql (shelowe) wpisz do przegladarki adres:
http://localhost/phpmyadmin/index.php
Zakladka SQL- Wykonanie zapytania/zapytań SQL do bazy danych- wklej treść skryptu z poleceniami.

*

W katalogu /var/lib/mysql są podkatalogi o nazwach identycznych jak istniejące bazy danych. Przed reinstalacją systemu lub zapobiegawczo np. raz w tygodniu warto kopiować do np. katalogu /archiwum cały katalog /var/lib/mysql, by w przyszłości odtworzyć bazę danych (z userami, hasłami, przywilejami itd.) po np. reinstalacji systemu. Archiwizację można wykonać poprzez crona, który będzie odpalał skrypt o przykładowej treści (wyjaśnienia znajdziesz w artykule o archiwizowaniu):

```
#!/bin/sh
tar -zcf /archiwum/mysql_`date +%Y.%m.%d`.tar.gz /var/lib/mysql
```

*

Na koniec wskazówka związana z MySQL. Może się zdażyć, że pozornie bez powodu baza danych padnie. Jedna z potencjalnych przyczyn to... przepełnienie partycji zawierającej katalog /var/lib/mysql przez logi systemowe (o ile zawartość katalogu /var jest montowana na jednej partycji). Co robić?

Wylistować partycje zleceniem: mount lub wyedytować plik /etc/fstab

Wyświetlić listę partycji wraz z ich wielkością, zajętością itd. zleceniem df -h. Pokaże się tabelka z informacjami: ROZMIAR PARTYCJI, UŻYWANE (zajęte), DOSTĘPNE, PROCENT ZAJĘTOŚCI.

*

Autor artykułu Kamil Twarogal

Witryna była dostępna pod adresami: strony.wp.pl/wp/twarogal , strony.wp.pl/wp/linuxtwarka , twarogal.republika.pl , klub.chip.pl/twarogal oraz gorzow-wlkp.net (w latach 2003/04).

Informacje o odwiedzających są rejestrowane i publicznie udostępniane na pod adresem: http://gorzow-wlkp.pl/licznik/

_____

```
mysql> CREATA DATABASE jego_baza_sql
mysql> CREATE DATABASE uzytkownik_login CHARACTER SET utf8 COLLATE utf8_polish_ci;
mysql> GRANT USAGE ON uzytkownik_login.* TO jego_baza_sql(małpa)localhost;
mysql> GRANT ALL ON uzytkownik_login.* TO jego_baza_sql(małpa)localhost IDENTIFIED BY 'haslo_uzytkownika';
```

Manipulacja uprawnieniami

Po dłuższym wstępie przejd□my do nieco ciekawszej cz□ci, czyli nadawania uprawnień i ich odbierania. Wszelkie zmiany dotyczšce tabel w bazie mysql możesz przeprowadzać za pomocš znanych ci z poprzedniej cz□ci kursy poleceń, takich jak select, insert, update czy delete. Możesz również w łatwy sposób zarzšdzać uprawnieniami za pomocš poleceń GRANT (dodawanie uprawnień) i REVOKE (odbieranie uprawnień).

Na poczštku ustawmy hasło dla użytkownika root, aby nikt niepowołany nie mógł w łatwy sposób dostać się do serwera baz danych:

```
mysql> UPDATE user SET Password = PASSWORD('pass') WHERE user = 'root';
Query OK, 2 rows affected (0.05 sec)
Rows matched: 2 Changed: 2 Warnings: 0

mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

Za pomocš polecenia UPDATE zmienili□my warto□ć pola Password na nowe hasło, tam gdzie użytkownik jest zdefiniowany jako root (dwa rekordy). Hasła w MySQL kodowane sš za pomocš funkcji PASSWORD(), dlatego musisz jej użyć, aby zapisać hasło w tabeli.

Ostatnie polecenie (FLUSH PRIVILEGES;) przeładowuje uprawnienia. Gdyby□my tego nie zrobili, zmiany nie byłyby widoczne, root dalej logowałby się bez hasła. Nie musisz od□wieżać uprawnień, jeżeli używasz wbudowanych funkcji do obsługi przywilejów:

```
mysql> SET PASSWORD FOR root = PASSWORD('mypass');
Query OK, 0 rows affected (0.05 sec)

mysql>
```

Spróbujmy teraz zalogować się do mysql-a z hasłem 'pass':
C:\apache\mysql\bin> mysql -u root -p
Enter password: ****
ERROR 1045: Access denied for user: 'root@localhost' (using password: YES)

C:\apache\mysq\bin>

Jak widać hasło 'pass' nie jest już aktualne, ponieważ zmieniliśmy je na 'mypass' za pomocš funkcji SET PASSWORD...

Spróbujmy zatem jeszcze raz, tym razem podajšc haslo 'mypass':
C:\apache\mysql\bin> mysql -u root -p
Enter password: ******

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 40 to server version: 3.23.33
Type 'help' for help

mysql>

I spróbujmy jeszcze zalogować się bez hasła, jak dawniej:
C:\apache\mysql\bin> mysql -u root
ERROR 1045: Access denied for user 'root@localhost' (Using password: NO)

C:\apache\mysql\bin>

Jak wiadć, nasz MySQL jest zabezpieczony hasłem.

Polecenie GRANT

Za pomocš tego polecenia dodajesz uprawnienia dostępu do baz danych, tabel czy kolumn.

Spróbujmy teraz utworzyć nowego użytkownika admin z dostępem do bazy danych nasza_baza i tabeli ksiazka_adresowa:
mysql> GRANT SELECT ON nasza_baza.ksiazka_adresowa TO admin@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql>

Mamy już w tej chwili dostępnego użytkownika admin, który ma prawo wybierać dane z tabeli ksiazka_adresowa z naszej bazy. Ten użytkownik nie ma ustawionego hasła, ale musi logować się z localhost:
C:\apache\mysql\bin> mysql -u admin

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 52 to server version: 3.23.33
Type 'help' for help.

mysql>

Jak widać, nowy użytkownik może się już zalogować. Przetestujmy teraz jego uprawnienia, najpierw na bazie mysql, potem na nasza_baza:
mysql> use mysql
ERROR 1044: Access denied for user: 'admin@localhost' to database 'mysql'

mysql> use nasza_baza
Database changed

mysql>INSERT INTO ksiazka_adresowa VALUES(0, 'Imie', 'adres', 'telefon', 'adres@email.pl');
ERROR 1142: insert command denied to user 'admin@localhost' for table 'ksiazka_adresowa'

mysql> select * from ksiazka_adresowa;
Empty set (0.00 sec)

mysql>

Jak wynika z powyższego przykładu, nasz nowy użytkownik nie ma dostępu do bazy mysql. Nie może również dodawać rekordów do tabeli ksiazka_adresowa w naszej bazie. Jedynie polecenie SELECT zostało zakończone sukcesem, bo tylko do tego ma uprawnienia użytkownik admin.

Spróbujmy utworzyć teraz użytkownika admin2 z uprawnieniami do wybierania, dodawania, edycji i usuwania rekordów:
mysql> GRANT SELECT,INSERT,UPDATE,DELETE ON nasza_baza.* TO admin2@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql>

Ten użtykownik ma prawo do wykonywania instrukcji SELECT, INSERT, UPDATE i DELETE na wszystkich tabelach w bazie nasza_baza. Znak * użyty zamiast nazwy tabeli oznacza dostęp do wszystkich tabel.

Spróbujmy teraz utworzyć użytkownika admin3 z prawem dodawania rekordów we wszystkich bazach danych systemu:
mysql> GRANT INSERT ON *.* TO admin3@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql>

Jak widać, aby zaznaczyć wszystkie bazy danych wraz z towarzyszšcymi im tabelami, użyli□my konstrukcji *.*, co oznacza <dowolna_baza>.<dowolna_tabela>:.

Spróbujmy teraz utworzyć użytkownika admin4 przychodzšcego z dowolnego hosta z prawem wybierania rekordów z bazy nasza_baza i dowolnej tabeli:

mysql> GRANT SELECT ON nasza baza.* TO admin4@'%';
Query OK, 0 rows affected (0.00 sec)

mysql>

Jak widać, aby oznaczyć dowolny host, użyli□my znaku % umieszczonego w cudzysłowach.

Spróbujmy teraz utworzyć użytkownika admin5 ze wszystkimi prawami w bazie nasza_baza i dowolnej tabeli przychodzšcego z localhost:
mysql> GRANT ALL PRIVILEGES ON nasza_baza.* TO admin5@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql>

Jak widać, aby zaznaczyć, że chcemy nadać użytkownikowi wszystkie prawa, użyli□my polecenia ALL PRIVILEGES.

Spróbujmy jeszcze utworzyć takiego użytkownika jak poprzedni, ale z prawem nadawania uprawnień (grantów):
mysql> GRANT ALL PRIVILEGES ON nasza_baza.* TO admin6@localhost WITH GRANT OPTION;
Query OK, 0 rows affected (0.00 sec)

mysql>

Jak widać, aby zaznaczyć, że chchcemy nadać wszystkie prawa razem z grantami, użyli□my polecenia WITH GRANT OPTION.

Spróbujmy teraz utworzyć użytkownika admin7 z dostępem do wszystkich baz danych i tabel, ale z wymaganym hasłem 'haslo':
mysql> GRANT ALL PRIVILEGES ON *.* TO admin7@localhost IDENTIFIED BY 'haslo';
Query OK, 0 rows affected (0.00 sec)

mysql>

Spróbujmy teraz zalogować się na użytkownika admin7, raz bez podania hasła, a następnie z hasłem:
C:\apache\mysql\bin> mysql -u admin7
ERROR 1045: Access denied for user: 'admin7@localhost' (Using password: NO)

C:\apache\mysq\bin> mysql -u admin7 -p
Enter password: *****

Welcome to the MySQL monitor. Commands with ; or \g.
Your MySQL connection id is 78 to server version: 3.23.33
Type 'help' for help.

mysql>

Jak widać hasło dla tego użytkownika jest wymagane.

Spróbujmy jeszcze utworzyć użytkownika admin8 z prawem wybierania pola imie oraz edycji pól adres i email z tabeli ksiazka_adresowa w naszej baze testowej:
mysql> GRANT SELECT(imie),UPDATE(adres,email) ON nasza_baza.ksiazka_adresowa TO admin8@localhost;
Query OK, 0 rows affected (0.22 sec)

mysql>

Jak widać, aby okre□lić kolumny, na jakich majš działać uprawnienia, wystarczy zapisać je w nawiasach okršgłych występujšcych po uprawnieniu.

Polecenie REVOKE

Za pomocš tego polecenia usuwasz uprawnienia dostępu do baz danych, tabel czy kolumn.

Spróbujmy teraz usunšć uprawnienia nadane użytkownikowi admin z dostępem do bazy danych nasza_baza i tabeli ksiazka_adresowa:
mysql> REVOKE SELECT ON nasza_baza.ksiazka_adresowa FROM admin@localhost;
Query OK, 0 rows affected (0.05 sec)

mysql>

Jak widać z powyższego przykładu, aby usunšć uprawnienia nadane użytkownikowi, posługujemy się podobnš składniš jak przy dodawaniu uprawnień. Różnice sš dwie: zamiast słowa GRANT występuje REVOKE, a zamiast TO - FROM.

Spróbujmy sprawdzić, czy rzeczywi□cie nie mamy dostępu do bazy danych nasza_baza:
C:\apache\mysql\bin> mysql -u admin

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 83 to server version: 3.23.33
Type 'help' for help.

mysql> use nasza_baza
ERROR 1044: Access denied for user: 'admin@localhost' to database 'nasza_baza'

mysql>

Na koniec spróbujmy jeszcze odebrać uprawnienia użytkownikowi admin8:
mysql> REVOKE SELECT(imie),UPDATE(adres,imie) ON nasza_baza.ksiazka_adresowa FROM admin8@localhost;
Query OK, 0 rows affected (0.00 sec)

mysql>

Dane z pliku zewnętrznego

Temat: Import z arkusza do CSV do bazy mysql

Będę importować siedem kolumn nazwanych kolejno:
- pole_a [typ znakowy]
- pole_b [typ numeryczny]
- pole_c [typ numeryczny]
- pole_d [typ numeryczny]
- pole_e [typ numeryczny]
- pole_f [typ numeryczny]
- pole_g [typ znakowy]

Format pliku danych:
Poniżej fragment tak zapisanego pliku, jak widać każde pole jest ujęte w znaki apostrofu, a pola są porozdzielane przecinkami.

```
"1","3","0","1","0","1","A4"
"67/27","3","0","1","0","1","A4"
"62a","3","0","1","0","1","A4"
"63b","3","0","1","0","1","A4"
"1524/1","3","0","1","1","1","A4"
"15","3","0","1","1","1","A4"
"15cc","3","0","1","1","1","A4"
```

Dzialania praktyczne

1)Przejście do linii komend MySQL-a:

```
$ mysql -u root -p
Enter password:
```

2)Tworzę nową bazę danych pod zbiory importowane z arkuszy kalkulacyjnych o nazwie arkusz_kalkulacyjny.

```
mysql> create database arkusz_kalkulacyjny;
```

3)Tworzę jednocześnie użytkownika o nazwie arkusz i haśle arkuszhaslo, który będzie miał wszystkie prawa do tabel tworzonych w bazie danych arkusz_kalkulacyjny.

```
mysql> grant all on arkusz_kalkulacyjny.* to arkusz@localhost identified by 'arkuszhaslo';
```

4)Wylogowuję się z konta administratora i loguję na nowo utworzone przeznaczone dla moich arkuszy.

```
mysql>exit;
```

```
arkusz_kalkulacyjny -u arkusz –parkuszhaslo enter
```

5)stworzenie tabeli w której będę składował dane z przykładowego zbioru *.csv.

```
mysql> CREATE TABLE `arkusz` (
  `id` int(10) unsigned NOT NULL auto_increment,
  `pole_a` varchar(10) NOT NULL,
  `pole_b` int(1) NOT NULL,
  `pole_c` int(1) NOT NULL,
  `pole_d` int(1) NOT NULL,
  `pole_e` int(1) NOT NULL,
  `pole_f` int(1) NOT NULL,
  `pole_g` varchar(3) NOT NULL
  PRIMARY KEY  (`id`),
);
```

Do listy pól, które będę importować z arkusza kalkulacyjnego (7 pól), w tablicy bazy danych dodaje jedno o identyfikatorze id. Jest to pole będące kluczem głównym tablicy, a dzięki fladze auto_increment będzie jednoznacznie identyfikowało każdy wprowadzony rekord.

6)Ładowanie danych

załadowanie danych ze zbiorów CSV, nasz zbiór (101.csv) został zapisany w katalogu /tmp. Jesteśmy cały czas w linii komend MySQL-a:

```
mysql>  LOAD DATA INFILE '/tmp/101.csv' INTO TABLE `arkusz` FIELDS
TERMINATED BY ',' ENCLOSED BY "" ESCAPED BY '\\' LINES TERMINATED BY
'\n' (pole_a, pole_b, pole_c, pole_d, pole_e, pole_f, pole_g);
```

TERMINATED BY →ZAKOŃCZONY PRZEZ
ENCLOSED BY → ZAŁĄCZANY (OTOCZONY) PRZEZ
ESCAPED BY → WYDOBYWANY PRZEZ

7)Składa instrukcji

```
LOAD DATA [LOW_PRIORITY | CONCURRENT] [LOCAL] INFILE 'file_name'
    [REPLACE | IGNORE]
    INTO TABLE tbl_name
    [CHARACTER SET charset_name]
    [{FIELDS | COLUMNS}
        [TERMINATED BY 'string']
        [[OPTIONALLY] ENCLOSED BY 'char']
        [ESCAPED BY 'char']
    ]
    [LINES
        [STARTING BY 'string']
        [TERMINATED BY 'string']
    ]
    [IGNORE number LINES]
    [(col_name_or_user_var,...)]
```

[SET col_name = expr,...]

Typy danych w kolumnach w MySQL

W MySQL wyróżnia się trzy podstawowe typy danych:

- liczbowy,
- daty i czasu
- łańcuchowy.

Każda z tych kategori dzieli się na szereg podtypow.

W przypadku niektórych typów danych możliwe jest określenie maksymalnej szerokosci wyswietlania. W poniższych tabelach wiekośc tę oznaczono literą M znajduje się w nawiasach kwadratowych. Największa dopuszczalna zawartośc parametru M wynosi 255.
Wszystkie parametry opcjonalne są przedstawione w nawiasach kwadratowych.

Typy liczbowe

Typy liczbowe dzielą się na:

- całkowitoliczbowe
- zmiennoprzecinkowe.

W przypadku tych drugich istnieje możliwośc zdeklarowania liczby cyfr znaczących po przecinku. W tabelach wielkość ta jest oznaczona symbolem D.
W przypadku typów całkowito liczbowych można je zawęzić do typu UNSIGNED (nieujemny).
Kolumna "Pamięć" określa ilość zajmowanej pamięci w bajtach!

Typy całkowito liczbowe

Typ	Zakres	Pamięć	Opis
TINYINT[(M)]	-127..128 lub 0..255	1	Bardzo małe liczby całkowite
BIT	-	-	Synonim TINYINT
BOOL	-	-	Synonim TINYINT
SMALLINT[(M)]	-32768..32767 lub 0..65535	2	Małe liczby całkowite
MEDIUMINT[(M)]	-8388608..8388607 lub 0..16777215	3	Średnie liczby całkowite
INT[(M)]	$-2^{31}..2^{31}-1$ lub $0..2^{32}-1$	4	Zwykłe liczby całkowite
INTEGER[(M)]	-	-	Synonim typu INT
BIGINT[(M)]	$-2^{63}..2^{63}-1$ lub $0..2^{64}-1$	8	Duże liczby całkowite

Typy zmiennoprzecinkowe

Typ	Zakres	Pamięć	Opis
FLOAT(precyzja	zaleznie od precyzji	różny	Używany do

)			deklarowania liczb zmiennoprzecinkowych o pojedynczej lub podwójnej precyzji
FLOAT[(M, D)]	±1.175494351E-38 ±3.402823466E+38	4	Liczby zmiennoprzecinkowe o zmiennej precyzji. Typ ten jest równoznaczny z typem FLOAT(4), pozwala przy tym na określenie szerokości wyświetlania i liczby cyfr znaczących po przecinku.
DOUBLE[(M, D)]	±1.7976931348623157 E -308 ±2.2250738585072014 E +308	8	Liczby zmiennoprzecinkowe o podwójnej precyzji. Typ ten jest równoznaczny z typem FLOAT(8), pozwala przy tym na określenie szerokości wyświetlania i liczby cyfr znaczących po przecinku.
DOUBLE PRECSION[(M, D)]	-	-	Synonim typu DOUBLE[(M, D)]
REAL[(M, D)]	-	-	Synonim typu DOUBLE[(M, D)]
DECIMAL[(M[, D])]	różny	M+2	Liczba zmiennoprzecinkowa przechowywana jako zmienna typu CHAR. Zakreszależy od wartości M - szerokości wyświetlania.
NUMERIC[(M, D)]	-	-	Synonim typu DECIMAL
DEC[(M, D)]	-	-	Synonim typu DECIMAL

Typy daty i czasu

MySQL obsługuje wiele typów daty i czas, w formie lańcuchowej lub liczbowej. Charakterystyczną cechą typu TIMESTAMP jest to, że jeżeli pole tego typu zostanie nie wypełnione, wówczas automatycznie zostanie w nim zapisany aktualny czas i data.

Typy daty i czasu

Typ	Zakres	Opis
DATE	1000-01-01 do 9999-12-31	Data wyświetlana w formacie RRRR-MM-DD.
TIME	-838:59:59 do 838:59:50	Czas wyświetlany w formacie GG:MM:SS. Zakres typu jest tak szeroki, że zapewne nigdy nie będzie w pełni wykorzystany.
DATETIME	1000-01-01 00:00:00 do 9999-12-31	Data i czas wyświetlane w formacie RRRR-MM-DD GG:MM:SS
TIMESTAMP[(M)]	1970-01-01 00:00:00 do roku 2037	Typ szczególnie przydatny do śledzenia transakcji. Format wyświetlania zależy od wartości parametru M, a górny zakres typu od systemu operacyjnego.
YEAR[(2|4)]	70-69(1970-2069) lub 1901-2155	Rok wyświetlany w formie dwu- lub czterocfrowej. Jak widać każdy z nich ma odmienny zakres.

Formaty wyświetlania wartości typu TIMESTAMP

Podany typ	Format wyświetlania
TIMESTAMP	RRRRMMDDGGMMSS
TIMESTAMP(14)	RRRRMMDDGGMMSS
TIMESTAMP(12)	RRMMDDGGMMSS
TIMESTAMP(10)	RRMMDDGGMM
TIMESTAMP(8)	RRRRMMDD
TIMESTAMP(6)	RRMMDD
TIMESTAMP(4)	RRMM
TIMESTAMP(2)	RR

Typy łańcuchowe

Typy łańcuchowe dzileimy na trzy grupy

- krótkie - krótkie fragmenty tekstu np.: CHAR
- długie - długie fragmenty tekstu oraz dane binarne np.: BLOB
- specjalne - typy wyliczeniowe np.: ENUM

Zwykłe typy łańcuchowe

Typ	Zakres	Opis
[NATIONAL] CHAR(M) [BINARY]	1-255 znaków	Łańcuch znaków o stałej długości M, gdzie M może przybierać wartości od 1 do 255. Słowo kluczowe NATIONAL wymusza użycie domyślnego zbioru znaków. Zbiór ten jest i tak domyślnie wykorzystywany przez MySQL, jednak opcja ta została udostępniona jako część standardu ANSI SQL. Słowo kluczowe BINARY wyłącza rozpoznawanie wielkości liter (domyślnie wielkość liter jest rozpoznawana).
CHAR	1	Synonim typu CHAR(1)
[NATIONAL] VARCHAR(M) [BINARY]	1-255	Łańcuch znaków o różnejdługości, reszta jak wyżej.

Typy TEXT i BLOB

Typ	Maksymalna długość (w znakach)	Opis
TINYBLOB	255	Mały obiekt BLOB.
TINYTEXT	255	Krótkie pole tekstowe.
BLOB	65535	Zwykły obiekt BLOB.
TEXT	65535	Pole tekstowe o zwykłej długości
MEDIUMBLOB	16777215	Średni obiekt BLOB.
MEDIUMTXT	16777215	Pole tekstowe o średniej długości.
LONGBLOB	4294967295	Duży obiekt BLOB.
LONGTEXT	4294967295	Długie pole tekstowe.

Typy SET i ENUM

Typ	Maksymalna ilość wartości w zbiorze	Opis
ENUM('wartosc1','wartosc2',...)	65535	W kolumnie tego typu może znajdować się tylko jedna wartość ze zbioru wartości dopuszczalnych lub NULL

SET('wartosc1','wartosc2',...)	64	W kolumnie tego typu może znajdować podzbiór zbioru wartości dopuszczalnych lub NULL.

Przykłady

1) SELECT actkwd, 2 FROM act;

Pokozana zostanie dodatkowa kolumna wypełniona 2-kami, można przeprowadzać oparacje matematyczne na kolumnach

2) SELECT info, 'tekst' AS "kol txt" FROM customer;

Tworzy stałą tekstową, wyświetlaną we wszystkich wierszach, dodatkowo dzięki AS utworzony został dwuczłonowy alias kolumny (Nazwa aliasu nie może być użyta w WHERE, może być natomiast zastosowana w ORDER BY)

3) SELECT deptnumb FROM org ORDER BY deptname DESC;

Sortowanie od największego do najmniejszego w kolumnie, która nie została wymieniona w SELECT-ie, można sortować także poprzez alias kolumny i numer kolumny

4) SELECT DISTINCT actkwd FROM act;

Wybranie unikalnych wierszy, klauzulę DISTINCT można zastosować tylko raz w SELECT-ie

5) SELECT name FROM staff WHERE name LIKE 'A%';

Wypisuje wszystkich na literę "A", znak % zastępuje ciąg znaków, W operatorze LIKE możliwe jest stosowanie operatorów NOT, OR, AND

6) SELECT name FROM staff WHERE name LIKE '_o%';

Wypisuje nazwika w których druga litera to o, ważna jest wielkość liter (rozróżniana), znak _ zastępuje jeden znak

7) SELECT job stanowisko, avg(salary) srednia

FROM employee

WHERE salary > 60000

GROUP BY job

HAVING avg(salary) > 50000

ORDER BY srednia;

Wypisuje pracowników których średnia zarobków większa jest niż 50000, w pierwszej kolejności eliminowani są pracownicy, którzy zarabiają mniej niż 60000, dalej następuje grupowanie po stanowiskach, HAVING wyświetla średnie zarobki które są większe niż 50000. Klauzula GROUP BY grupuje wiersze o tej samej wartości, funkcje AYG, MAX, MIN, SUM oraz COUNT operują na każdej grupie osobno. Klauzula HAVING używana jest w połączeniu z klauzulą group by w celu ograniczenia wyświetlanych grup. Warunek szukania musi zawierać funkcję agregującą. Po zgrupowaniu wierszy przez klauzulę group by, klauzula HAVING wyświetla tylko te wiersze spośród zgrupowanych, które spełniają warunki wyszczególnione w klauzuli HAVING. Klauzula HAVING może być użyta tylko wówczas gdy w zapytaniu znajduje się GROUP BY.

8) SELECT lastname FROM employee GROUP BY lastname

HAVING COUNT(lastname) > 1;

Wypisuje powtarzające się nazwiska

9) SELECT YEAR (hiredate), MAX(salary) FROM employee GROUP BY YEAR(hiredate);

Wybiera rok z daty I grupuję ją wypisując maksymalny zarobek w danym roku.

10) SELECT e.emstdate, e.emendate, p.actno, pr.projname

FROM empproject e

INNER JOIN project p ON e.projno=p.projno

INNER JOIN project pr ON e.actno=p.actno ;

Połączenie 3 tabel dzięki klauzuli INNER JOIN

11) SELECT e.empno, e.firstnme, e.lastname, er.resume_format, er.resume

FROM emp_resume er

FULL JOIN employee e ON e.empno=er.empno;

Wybierane są wszystkie rekordy w tabelach, nawet te które nie pokrywają się z tymi w drugiej tabeli

12) SELECT sales_person, region FROM sales

UNION

SELECT lastname, firstnme FROM employee;

Dwie tabele połączone zostały w jeden zbiór wynikowy, musi być tutaj podana taka sama liczba kolumn w obu zapytaniach, musi być w nich także taki sam typ, zapytanie to automatycznie eliminuje duplikaty, jeżeli chcemy wyświetlić powtarzające się rekordy używamy klauzuli UNION ALL (zastosowanie ORDER BY odnosi się do obu zapytań)

13) SELECT e.empno, e.lastname, emp.projno FROM employee e

INNER JOIN empproject emp ON e.empno=emp.empno

UNION

SELECT e.empno, e.lastname, p.projno FROM employee e

INNER JOIN project p ON p.respemp=e.empno

ORDER BY 1;

Połączone zostały dwa złączenia INNER JOIN w jedno zapytanie wynikowe

14) SELECT deptno FROM Project

EXCEPT

SELECT workdept FROM employee;

Zapytanie to umożliwia odjęcie jednego zbioru od drugiego, w obu zapytaniach musi być podana taka sama liczba kolumn (takie same typy), zapytanie to automatycznie eliminuje duplikaty, jeśli chcemy wyświetlić powtarzające się rekordy możemy użyć polecenia EXCEPT ALL

15) SELECT workdept FROM employee

INTERSECT

SELECT deptno FROM Project;

Dzięki temu zapytaniu możemy wyznaczyć część wspólną zbioru, INTERSECT ALL nie eliminuje duplikatów

16) SELECT * FROM employee

```
WHERE salary < (SELECT avg(salary) FROM employee);
```

Wykonane zostało tutaj podzapytanie, czyli zagnieżdzenie jednego zapytania w drugim, w rezultacie otrzymujemy tych pracowników których zarobki są mniejsze od średnich zarobków.

17)
```
SELECT * FROM employee

WHERE salary = (SELECT MIN(salary) FROM employee)

UNION

SELECT * FROM employee

WHERE salary = (SELECT MAX(salary) FROM employee);
```

Zapytanie to wypisuje informacje o pracowniku zarabiającym w firmie najmniej i o pracowniku zarabiającym najwięcej (można to zapytanie napisać w 2 sposób, mianowicie zamiast UNION użyć OR)

18)
```
SELECT deptname FROM department

WHERE deptno NOT IN ( SELECT workdept FROM employee)
```

Wybiera te departamenty do których nie są przypisani pracownicy

19)
```
SELECT * FROM employee

WHERE salary > (SELECT MAX(salary) FROM employee

        WHERE job='DESIGNER');
```

Zapytanie wybiera tych pracowników, którzy zarabiają więcej niż wszyscy pracownicy pracujący na stanowisku DESIGNER, w WHERE typy znakowe, daty, lub czasu muszą być otoczone apostrofem, rozróżniane są także wielkie i małe litery

20)
```
SELECT firstnme, lastname, salary, sex, job

FROM employee

WHERE salary > 50000 AND ( job = 'MANAGER' OR job = 'DESIGNER');
```

Zapytanie to wyświetla osoby zarabiające powyżej 50000 na dwóch stanowiskach, dzięki nawiasom ustalona jest kolejność sprawdzania warunków.

21) SELECT firstnme, lastname, salary, sex, job

FROM employee

WHERE job IN ('MANAGER' , 'DESIGNER');

Wybiera określone w klauzurze IN wartości ze zbioru (wartości typu znakowego, daty, czasu muszą być objęte apostrofem

22) SELECT firstnme, lastname, salary, sex, job

FROM employee

WHERE salary BETWEEN 30000 AND 60000 ;

Wybiera wartości z przedziału określonego w klauzuli BETWEEN

23) SELECT firstnme, lastname, salary, sex, job

FROM employee

WHERE phoneno IS NULL ;

Wybiera wartości NULL, przeciwieństwo to NOT NULL

24) SELECT e.firstnme,e.lastname,d.deptname

FROM employee e, department d

WHERE e.workdept=d.deptno;

Połączenie za pomocą klauzuli WHERE (gdy chcemy połączyć więcej table musimy zastosować w WHERE operator AND)

25) SELECT P.IMIE, P.NAZWISKO, P.PENSJA,

COALESCE (P.DODATEK, O) AS DODATEK,

P.PENSJA + COALESCE (P.DODATEK, 0) AS DO_WYPŁATY

FROM DB2ADMIN.PRACOWNICY P

WHERE P,PENSJA > 1100 ORDER BY P.NAZWISKO;

Działanie funkcji coalesce najpierw w kolumnie DODATEK zamienia wszystkie wystąpienia wartości null na wartość zera, a następnie robi to samo przy obliczaniu wartości do wypłaty. Funkcja COASESCE może zmieniać wartości także na typ tekstowy np. COALESCE(nazwa_kolumny, 'Nie posiada')

26) SELECT P.IMIE, P.NAZWISKO, P.PENSJA,

DECIMAL ((P.PENSJA * 11.3)/100, 8, 2} AS KWOTA_PODWYZKI

FROM DB2ADMIN.PRACOWNICY P ORDER BY P.NAZWISKO;

Funkcja decimal została zaimplementowana tylko w systemie DB2. Funkia DBCIMAL zwraca dziesiętną reprezentację wartości numerycznej. Pierwszy parametr zawiera wartość do reprezentacji, drugi parametr określa ilość cyfr przed przecinkiem, trzeci parametr określa liczbę miejsc po przecinku.

27) ROUND ((P.PENSJA * 11.31/100, 0) AS KWOTA_PODWYZKI

Funkcja round została zaimplementowana tylko w systemie DB2. Służy ona do zaokrąglania wyników, Funkcja ta w pierwszym argumencie musi zawierać wartość do zaokrąglenia, w drugim natomiast podaje się liczbę miejsc po przecinku, do jakiej ma zostać zaokrąglona wartość. Poniższy przykład zaokrągla wartości do liczb całkowitych. Wartości dziesiętne poniżej 0,50 zostały zaokrąglone do zera, natomiast powyżej 0,50 do jedności.

28) SELECT P.IMIE, P.NAZWISKO, P.DZIAL,

P.STANOWISKO, P.DATA__ZATR

FROM DB2ADMIN.PRACOWNICY P

WHERE CURRENT DATE - P.DATA_ZATR >= 020000

ORDER BY P.NAZWISKO;

Zapytanie wybiera pracowników zatrudnionych co najmniej 2 lata. Porównywana wartość 020000 przedstawia 02 rok, 00 miesięcy i 00 dni. Funkcja current datę zwraca bieżącą datę.

29) SELECT P.DZIAL, AVG(P.PENSJA) AS SREDNIA_PENSJA

FROM DB2ADMIN.PRACOWNICY P

WHERE P.STANOWISKO <> 'KIEROWNIK’

```
GROUP BY P.DZIAL

HAVING AVG(P.PENSJA) > (SELECT AVG(P.PENSJA)

FROM DB2ADMIN.PRACOWNICY P

WHERE P.STANOWISKO <> 'KIEROWNIK') ORDER BY SREDNIA_PENSJA;
```

Podzapytanie umieszczone w klauzuli HAVING

30) Inne funkcje:

- CURRENT TIME – zwraca bieżący czas

- CURRENT TIMESTAMP – zwraca dokładny bieżący czas

- YEAR – pozwala na odczytanie samego roku z daty

- MONTH – wybiera miesiąc z daty

- DAY – wybiera dzień z daty

- SUBSTR – wybiera pewną część łańcucha

- CONCAT - pozwala łączyć ciągi znaków w jeden łańcuch wynikowy (można też użyć ||)

- CASE - pozwala na wybranie pewnej wartości w zależności od wartości w innej kolumnie

Wyrażenie case dostępne jest tylko w systemie DB2. W przykładzie poniżej sprawdzamy, czy klient pochodzi z Warszawy; jeżeli tak, to w kolumnie wpisywana jest wartość „Klient oddziału macierzystego", w przeciwnym razie jest to „Klient z przedstawicielstwa".

```
SELECT K.IMIE, K.NAZWISKO, K.MIASTO,

CASE K.MIASTO

WHEN 'WARSZAWA' THEN 'Klient oddziału macierzystego'

ELSE 'Klient z przedstawicielstwa'

END

FROM DB2ADMIN.KLIENCI K ORDER BY K.NAZWISKO;
```

- SUM - funkcja służąca do obliczenia sumy wartości w określonych kolumnach,

- AVG - oblicza średnią wartości \v kolumnie,

- MIN - znajduje minimalną wartość,

- MAX - znajduje maksymalną wartość,

- COUNT - służy do zliczania wystąpień pewnej wartości w wierszach (funkcja ta może zliczać powtarzające się wiersze np. COUNT(DISTINCT P.DZIAL))

- ALL – stosowane w podzapytaniach, złuży np. do zsumowania wszystkich warości znajdujących się w GROUP BY i porównania ich do największej wartości

- ANY – stosowane w podzapytaniach, złuży np. do zsumowania wszystkich warości znajdujących się w GROUP BY i porównania ich do najmniejszej wartości