Bazy danych .

Komunikacja z serwerem

Aby połączyć się z serwerem i móc wykonywać czynności związane z obsługą baz, potrzebny jest program klienta. Razem z serwerem ***MySQL*** dostępny jest działający w wierszu poleceń program ***mysql***.

Aby uruchomić program klienta i połączyć się z serwerem, należy komendę:

mysql –u root –phasło

Po chwili nastąpi nawiązanie połączenia z serwerem i będzie można wykonywać zapytania i inne czynności związane z zarządzaniem, takie jak tworzenie i usuwanie baz, tworzenie, usuwanie i modyfikacja kont użytkowników oraz tabel, a także wydawanie zapytań w języku SQL.

Zarządzanie bazami danych

Serwer ***MySQL*** może jednocześnie obsługiwać wiele baz danych. W celu stworzenia bazy danych należy wydać polecenie ***create database*** w schematycznej postaci:

create database nazwa_bazy;

Jeśli chcemy na przykład utworzyć bazę o nazwie ***testphp***, napiszemy

create database nazwa_bazy;

Po kliknięciu przycisku Enter serwer potwierdzi utworzenie bazy, wyświetlając tekst:

Query OK, 1 row affected (0.05 sec)

Można teraz wydawać kolejne polecenia

W celu usunięcia bazy z serwera należy wykorzystać komendę ***drop database*** w postaci:

drop database nazwa_bazy

Wybór bazy danych

Jak wiadomo, na serwerze może istnieć wiele baz danych tworzonych za pomocą polecenia ***CREATE DATABASE***. Ponieważ każda baza jest niezależna i może zawierać takie same elementy jak każda inna, serwer musi wiedzieć, z którą bazą aktualnie chcemy pracować. Takiego wyboru można dokonać po zalogowaniu się do serwera, za pomocą instrukcji ***use*** w postaci:

use nazwa_bazy

Wyboru bazy można także dokonać w wierszu poleceń, podając nazwę podczas wywoływania aplikacji klienta ***mysql*** (przy założeniu, że chcemy używać bazy o nazwie ***testphp***, logując się jako użytkownik ***php***), np.:

mysql –u php –ptest testphp

Tworzenie tabel

Do tworzenia tabel służy instrukcja ***CREATE TABLE*** w schematycznej postaci:

CREATE TABLE nazwa_tabeli

(

nazwa_kolumny_1 typ_kolumny_1 [atrybuty],

nazwa_kolumny_2 typ_kolumny_2 [atrybuty],

…

nazwa_kolumny_N typ_kolumny_N [atrybuty]

)

Dla treningu spróbujmy utworzyć prostą tabele klient, która będzie zawierała dwie kolumny. Pierwsza kolumna o nazwie Indeks będzie przechowywała liczby całkowite, a druga o nazwie Nazwa – ciągi o maksymalnej długości 20 znaków.

CREATE TABLE klient(

Indeks INTEGER,

Nazwa VARCHAR(20)

);

Każda kolumna może mieć dodatkowe atrybuty. Najczęściej spotykane to: ***PRIMARY KEY, NOT NULL, AUTO_INCREMENT, INDEX, UNIQUE.***

Atrybut ***PRIMARY KEY*** oznacza, że dana kolumna jest kluczem podstawowym (głównym).

Atrybut ***NOT NULL*** oznacza, że w danej kolumnie nie mogą znajdować się wartości puste.

Atrybut ***AUTO_INCREMENT*** na zastosowanie jedynie do kolumn, które przechowują wartości całkowite. Podczas wstawiania nowego wiersza wartość takiej kolumny może być automatycznie zwiększana o jeden.

Atrybut ***INDEKS*** oznacza, że dana kolumna będzie indeksowana. Indeks to specjalna dodatkowa struktura porządkująca dane w kolumnie.

Atrybut ***UNIQUE*** oznacza, że dane w kolumnie będą musiały być unikatowe, tzn. nie będą mogły istnieć dwa wiersze o takiej samej wartości.

Jeśli zatem w naszej przykładowej tabeli klient kolumna ***Indeks*** miałaby być kluczem podstawowym, należałoby zastosować konstrukcje:

CREATE TABLE klient(

Indeks INTEGER NOT NULL PRIMARY KEY,

Nazwa VARCHAR(20)

);

Modyfikacja tabel

Jeśli konieczna jest zmiana struktury już istniejącej tabeli, należy skorzystać z polecenia ***ALTER TABLE*** w postaci:

ALTER TABLE nazwa_tabeli zmiana1[, zmiana2 [, . . . ,[zmianaN]]]

Przykład użycia ***ALTER TABLE - ADD COLUMN:***

ALTER TABLE ADD pozwala na dodanie nowej kolumny do istniejącej tabeli.

ALTER TABLE nazwa_tabeli ADD COLUMN nazwa_kolumny typ_danych

Przykład użycia ***ALTER TABLE - DROP COLUMN:***

ALTER TABLE DROP usuwa istniejącą kolumnę z tabeli.

ALTER TABLE nazwa_tabeli DROP COLUMN nazwa_kolumny typ_danych;
UWAGA!
Wraz z usunięciem kolumny zostaną usunięte wszystkie dane zapisane w kolumnie.

Przykład użycia ***ALTER TABLE - ALTER COLUMN:***

ALTER TABLE ALTER COLUMN pozwala na modyfikację typu danych w danej kolumnie. Czasami na przykład wymagane jest zwiększenie dopuszczalnej ilości znaków w polu typu ***VARCHAR().***
W tej sytuacji ***ALTER TABLE ALTER COLUMN*** okaże się pomocne.

ALTER TABLE nazwa_tabeli ALTER COLUMN nazwa_kolumny typ_danych;

Usuwanie tabel

Tabele usuwa się za pomocą instrukcji ***DROP TABLE*** o schematycznej postaci:

DROP TABLE nazwa1, nazwa2, … , nazwaN

Przykładowo usunięcie tabeli o nazwie Klient wykonamy, wydając polecenie:

DROP TABLE Klient

Natomiast równoczesne usunięcie dwóch tabel o nazwach Klient i Zamowienia uzyskamy, stosując konstrukcję:

DROP TABLE Klient, Zamowienia

Typy danych w kolumnach

Każda kolumna tabeli w bazie danych ma przypisany typ, który określa rodzaj danych. Występujące w SQL typy danych można podzielić na cztery główne rodzaje:

- **LICZBY CAŁKOWITE**
- **LICZBY ZMIENNOPRZECINKOWE**
- **DATA I CZAS**
- **TEKST**

LICZBY CAŁKOWITE

TYP	ZAKRES WARTOŚCI
tinyint(M) (lub bit, bool, boolean)	-128 do 127
tinyint unsigned(M) (lub bit unsigned, bool unsigned, boolean unsigned)	0 do 255
smallint(M)	-32768 do 32767
smallint unsigned(M)	0 do 65535
mediumint(M)	-8388608 do 8288607
mediumint unsigned(M)	0 do 16777215

int(M) (lub integer)	-2147483648 do 2147483648
int unsigned(M) (lub integer unsigned)	0 do 4294967295
bigint(M)	-9223372036854775808 do 9223372036854775807
bigint unsigned(M)	0 do 18446744073709551615

LICZBY ZMIENNOPRZECINKOWE

TYP	ZAKRES WARTOŚCI
float (M oraz **D)**	-3.402823466E+38 do 1.175494351E-38
double (M oraz **D)**	-1.7976931348623157E+308 do 2.2250738585072014E-308
real (M oraz **D)**	jak wyżej, czyli: -1.7976931348623157E+308 do 2.2250738585072014E-308
decimal (M oraz **D)**	Liczba zmiennoprzecinkowa zapisana w postaci łańcucha znaków o zakresie jak **double/real**

DATA I CZAS

TYP	ZAKRES WARTOŚCI
date	przykładowy wygląd: **2004-12-31**
datetime	przykładowy wygląd: **2004-12-31 23:59:59**
timestamp(M)	data i godzina zapisana w formacie: RRRRMMDDGGMMSS
time	ilość godzin (minut/sekund) zapisana w formacie: od -838:59:59 do 838:59:59
year(2)	Rok zapisany za pomocą dwóch cyfr

year(4)	Rok zapisany za pomocą czterech cyfr

TEKST

TYP	**ZAKRES WARTOŚCI**
char	Jeden pojedyńczy znak jak **char(1)**
char(M) tylko w typie char (oraz varchar) można narzucić kodowanie Patrz na dodatkowe wyjaśnienie pod tabelką	Ciąg znaków o wartości wpisanej jako **M** - od 0 do 255 znaków Uwaga: można narzucić kodowanie wpisując: **binary** lub **ascii** lub **unicode** Praktycznie wygląd parametru może być np. taki: **char(27) unicode**
varchar(M)	Czym się różni **char** od **varchar** wyjaśniłem pod tabelką
tinyblob lub **tinytext**	Ciągi znaków o długości max. 255 znaków
blob lub **text**	Ciągi znaków o długości do 65538 znaków (64KB)
mediumblob lub **mediumtext**	Ciągi znaków o długości do 16777215 znaków
longblob lub **longtext**	Ciągi znaków o długości do 4294967295 znaków
enum(wartosc1, wartosc2, wartosc3...)	Pole tekstowe, które może mieć długość znaków zgodną z jedną z kilku wartości. W jednej takiej wartości może być **maksymalnie**65535 liter-cyfr-spacji
set(wartosc1, wartosc2...)	Pole tekstowe, które może przyjąć zero lub kilka wartości (zgodnych z listą). W jednej takiej wartości może być **maksymalnie** 64 liter-cyfr-spacji

Wprowadzanie danych

Utworzone tabele trzeba w jakiś sposób wypełnić danymi. Służy do tego występująca w kilku wersjach instrukcja ***INSERT INTO***. Typowe jej wywołanie ma postać:

INSERT INTO tabela[(kolumna1, kolumna2, …. kolumnaN) VALUES (wartość1, wartość2, …. wartość)

Powoduje ona wprowadzenie do tabeli nowego wiersza, w którym w polu kolumna1 zostala zapisana wartość2, w polu kolumna2 – wartość 2 itd. Załóżmy, że w bazie istnieje tabela Klienci utworzona za pomocą instrukcji:

CREATE TABLE Klienci(

KlientID INTEGER PRIMARY KEY,

Imie VARCHAR (25),

Nazwisko VARCHAR (25),

Adres VARCHAR (60)

);

I chcielibyśmy zapisać w niej nowy wiersz z danymi. Należałoby zastosować instrukcje postaci:

INSERT INTO Klienci

(KlientId, Imie, Nazwisko, Adres)

VALUES

(1, 'Jan', 'Kowalski', 'Klonowa 24, Poznań');

Pobieranie danych

Dane zapisane w tabelach bazy można pobierać za pomocą instrukcji ***SELECT***, która schematycznie wygląda następująco:

SELECT kolumna1, kolumna2, … , kolumnaN

FROM tabela

[WHERE warunek]

[ORDER BY kolumna1, kolumna2, … , kolumnaN [ASC|DEC]]

Oznacza ona: pobierz wartości wymienionych kolumn z tabeli tabela spełniających warunek, a wyniki posortuj względem kolumn wymienionych w klauzuli ***ORDR BY*** rosnąco ***(ASC)*** lub malejąco ***(DEC).***

Aby zobaczyć, jak w praktyce działają proste zapytania typu ***SELECT***, utworzymy tabelę przechowującą dane o osobach: imię, nazwisko oraz rok i miejsce urodzenia.

CREATE TABLE osoba

(

Id INTEGER PRIMARY KEY,

Imie VARCHAR(25),

Nazwisko VARCHAR(35)

Rok_urodzenia YEAR,

Miejsce_urodzenia VARCHAR(35)

);

Za pomocą serii instrukcji ***INSERT*** wprowadzimy teraz do tak utworzonej tabeli przykładowe dane, w sumie 10 wierszy.

Pobieranie wszystkich wierszy tabeli

Najprostsza instrukcja SELECT pozwoli na pobranie wszystkich wierszy zawartych w tabeli. Będzie ona miała postać:

SELECT * FROM osoba

```
mysql> SELECT *FROM osoba
    -> ;
+----+----------+-------------+----------------+--------------------+
| Id | Imie     | Nazwisko    | Rok_urodzenia  | Miejsce_urodzenia  |
+----+----------+-------------+----------------+--------------------+
|  1 | Adam     | Kowalski    |           1964 | Bydgoszcz          |
|  2 | Adam     | Nowak       |           1972 | Szczecin           |
|  3 | Andrzej  | Kowalski    |           1986 | Nidzica            |
|  4 | Arkadiusz| Malinowski  |           1986 | Kielce             |
|  5 | Andrzej  | Malinowski  |           1989 | Kielce             |
|  6 | Krzysztof| Nowicki     |           1986 | Bydgoszcz          |
|  7 | Kacper   | Adamczyk    |           1971 | Kielce             |
|  8 | Kamil    | Andrzejczak |           1971 | Radom              |
|  9 | Krzysztof| Arkuszewski |           1989 | Szczecin           |
| 10 | Kamil    | Borowski    |           1976 | Skierniewice       |
+----+----------+-------------+----------------+--------------------+
10 rows in set (0.00 sec)
```

Sortowanie wyników

Gdybyśmy chcieli, aby wyniki zostały posortowane, należałoby użyć dodatkowej klauzuli ***ORDER BY.***

SELECT * FROM osoba ORDER BY Nazwisko

```
mysql> SELECT * FROM osoba ORDER BY Nazwisko
    -> ;
+----+-----------+-------------+---------------+--------------------+
| Id | Imie      | Nazwisko    | Rok_urodzenia | Miejsce_urodzenia  |
+----+-----------+-------------+---------------+--------------------+
|  7 | Kacper    | Adamczyk    |          1971 | Kielce             |
|  8 | Kamil     | Andrzejczak |          1971 | Radom              |
|  9 | Krzysztof | Arkuszewski |          1989 | Szczecin           |
| 10 | Kamil     | Borowski    |          1976 | Skierniewice       |
|  1 | Adam      | Kowalski    |          1964 | Bydgoszcz          |
|  3 | Andrzej   | Kowalski    |          1986 | Nidzica            |
|  5 | Andrzej   | Malinowski  |          1989 | Kielce             |
|  4 | Arkadiusz | Malinowski  |          1986 | Kielce             |
|  2 | Adam      | Nowak       |          1972 | Szczecin           |
|  6 | Krzysztof | Nowicki     |          1986 | Bydgoszcz          |
+----+-----------+-------------+---------------+--------------------+
10 rows in set (0.00 sec)
```

Pobieranie zawartości wybranych kolumn

Jeżeli chcemy wyświetlić zawartość jedynie niektórych kolumn z wybranej tabeli ich nazwy należy umieścić za słowem ***SELECT***, oddzielając je przecinkiem.

```
mysql> SELECT Imie, Nazwisko FROM osoba;
+-----------+-------------+
| Imie      | Nazwisko    |
+-----------+-------------+
| Adam      | Kowalski    |
| Adam      | Nowak       |
| Andrzej   | Kowalski    |
| Arkadiusz | Malinowski  |
| Andrzej   | Malinowski  |
| Krzysztof | Nowicki     |
| Kacper    | Adamczyk    |
| Kamil     | Andrzejczak |
| Krzysztof | Arkuszewski |
| Kamil     | Borowski    |
+-----------+-------------+
10 rows in set (0.00 sec)
```

Selektywne pobieranie danych

Pobieranie całej zawartości tabeli stosuje się rzadko. W praktyce najczęściej interesuje nas pewien podzbiór danych. Otrzymanie określonego zestawu wierszy zapewni nam klauzula ***WHERE*** instrukcji ***SELECT***. Należy za nią umieścić warunek, jaki muszą spełniać wiersze, aby znalazły się w wynikach zapytania. Warunek w klauzuli może zawierać operatory:

- =, <>, <, >,<=
- ***IS NULL*** – Zwraca wartość true, jeśli argument znajdujący się z lewej strony jest różny od NULL.
- ***BETWEEN N AND M*** -zwraca wartość true, jeśli argument znajdujący się z lewej strony ma wartość z przedziału od N do M

- ***IN*** – zwraca wartość true, jeśli argument znajdujący się z lewej strony jest równy jednej z wartości wymienionych w nawiasie okrągłym za operatorem.
- ***AND, OR, XOR, NOT.***

Oprócz wymienionych powyżej operatorów często wykorzystywana jest funkcja operująca na ciągach znaków – ***LIKE***. Jej wywołanie ma postać:

wyrażenie LIKE wzorzec

Zwraca ona true, jeśli wyrażenie pasuje do wzorca. Jako wyrażenie zazwyczaj jest stosowana nazwa kolumny. Argument wzorzec może zawierać dwa znaki specjalne. Pierwszy z nich to %, który zastępuje dowolną liczbę znaków, drugi znak specjalny zastępujący dokładnie jeden znak _ (podkreślenie).

Spróbujmy wykonać kilka praktycznych przykładów:

- Pobierzmy wszystkie wiersze tabeli osoba, które w polu nazwisko mają zapisaną wartość Kowalski

```
+----+---------+----------+---------------+--------------------+
| Id | Imie    | Nazwisko | Rok_urodzenia | Miejsce_urodzenia  |
+----+---------+----------+---------------+--------------------+
|  1 | Adam    | Kowalski |          1964 | Bydgoszcz          |
|  3 | Andrzej | Kowalski |          1986 | Nidzica            |
+----+---------+----------+---------------+--------------------+
2 rows in set (0.00 sec)
```

- Lista osób urodzonych po roku 1985

```
+----+-----------+-------------+---------------+--------------------+
| Id | Imie      | Nazwisko    | Rok_urodzenia | Miejsce_urodzenia  |
+----+-----------+-------------+---------------+--------------------+
|  3 | Andrzej   | Kowalski    |          1986 | Nidzica            |
|  4 | Arkadiusz | Malinowski  |          1986 | Kielce             |
|  5 | Andrzej   | Malinowski  |          1989 | Kielce             |
|  6 | Krzysztof | Nowicki     |          1986 | Bydgoszcz          |
|  9 | Krzysztof | Arkuszewski |          1989 | Szczecin           |
+----+-----------+-------------+---------------+--------------------+
5 rows in set (0.00 sec)
```

- Lista osób o identyfikatorach z przedziału 3-6

```
+----+-----------+------------+---------------+--------------------+
| Id | Imie      | Nazwisko   | Rok_urodzenia | Miejsce_urodzenia  |
+----+-----------+------------+---------------+--------------------+
|  3 | Andrzej   | Kowalski   |          1986 | Nidzica            |
|  4 | Arkadiusz | Malinowski |          1986 | Kielce             |
|  5 | Andrzej   | Malinowski |          1989 | Kielce             |
|  6 | Krzysztof | Nowicki    |          1986 | Bydgoszcz          |
+----+-----------+------------+---------------+--------------------+
4 rows in set (0.00 sec)
```

- Dane wszystkich osób, których imiona zaczynają się od ciągu Ka

```
+----+--------+-------------+---------------+--------------------+
| Id | Imie   | Nazwisko    | Rok_urodzenia | Miejsce_urodzenia  |
+----+--------+-------------+---------------+--------------------+
|  7 | Kacper | Adamczyk    |          1971 | Kielce             |
|  8 | Kamil  | Andrzejczak |          1971 | Radom              |
| 10 | Kamil  | Borowski    |          1976 | Skierniewice       |
+----+--------+-------------+---------------+--------------------+
3 rows in set (0.00 sec)
```

- Dane osób, których imiona zaczynają się na literę A, urodzonych po roku 1970 w Kielcach lub w Szczecinie

```
+----+-----------+------------+---------------+--------------------+
| Id | Imie      | Nazwisko   | Rok_urodzenia | Miejsce_urodzenia |
+----+-----------+------------+---------------+--------------------+
|  2 | Adam      | Nowak      |          1972 | Szczecin           |
|  4 | Arkadiusz | Malinowski |          1986 | Kielce             |
|  5 | Andrzej   | Malinowski |          1989 | Kielce             |
+----+-----------+------------+---------------+--------------------+
3 rows in set (0.00 sec)
```

Modyfikacja danych

Dane zapisane w tabelach mogą być zmieniane i modyfikowane. Służy do tego instrukcja ***UPDATE***, która ma ogólną postać:

UPDATE tabela

SET kolumna1=wartość1, kolumna2=wartość2, … , kolumnaN=wartość

[WHERE warunek]

Oznacza ona : w tabeli tabela, w wierszach spełniających warunek ***warunek***, zmień pole ***kolumna1*** na ***wartość1*** itd.

Np.:

UPDATE osoba SET Rok_urodzenia=1988 WHERE id=5;

Usuwanie danych

Do usuwania wierszy z tabel służy instrukcja ***DELETE*** o schematycznej postaci:

DELETE FROM tabela

WHERE warunek

Oznacza ona: usuń z tabeli ***tabela*** wszystkie wiersze spełniające ***warunek***.