Bazy danych

6. SQL — funkcje daty i czasu, zmienne tymczasowe, aliasy

P. F. Góra

http://th-www.if.uj.edu.pl/zfs/gora/

semestr letni 2007/08

MySQL i programowanie wsadowe

`C:\wyklady\bazy> mysql` < *nazwa_pliku*

`C:\wyklady\bazy> mysql -h`*hostname* `-u`*username* `-p` < *nazwa_pliku*

Plik musi być dostępny dla klienta.

Pliki wsadowe można też wołać “z wnętrza” klienta:

`mysql> source` *nazwa_pliku*`;`

Uwaga: Jeśli któreś polecenie w pliku wsadowym spowoduje błąd, dalsze polecenia nie są wykonywane.

```
mysql> CREATE DATABASE Data_I_Czas CHARACTER SET cp1250;
Query OK, 1 row affected (0.06 sec)

mysql> source utworz.sql;
Database changed
Query OK, 0 rows affected (0.31 sec)

Query OK, 150 rows affected (0.26 sec)
Records: 150  Duplicates: 0  Warnings: 0

mysql> DESCRIBE Zamowienia;
+--------------+----------------------+------+-----+---------+----------------+
| Field        | Type                 | Null | Key | Default | Extra          |
+--------------+----------------------+------+-----+---------+----------------+
| NrZam        | smallint(5) unsigned | NO   | PRI | NULL    | auto_increment |
| NrKlienta    | smallint(5) unsigned | NO   |     |         |                |
| Kwota        | float unsigned       | NO   |     |         |                |
| DataZlozenia | date                 | NO   |     |         |                |
| DataZaplaty  | date                 | YES  |     | NULL    |                |
+--------------+----------------------+------+-----+---------+----------------+
5 rows in set (0.26 sec)
```

Funkcje daty i czasu

`CURDATE()`, `CURRENT_DATE()`, `CURRENT_DATE` — bieżąca data

```
mysql> SELECT CURDATE();
+------------+
| CURDATE()  |
+------------+
| 2008-04-09 |
+------------+
1 row in set (0.00 sec)
mysql> SELECT CURDATE() AS Dzisiaj;
+------------+
| Dzisiaj    |
+------------+
| 2008-04-09 |
+------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT * FROM Zamowienia WHERE DataZaplaty > CURDATE()
    -> ORDER BY DataZaplaty ASC LIMIT 5;
+-------+-----------+---------+--------------+-------------+
| NrZam | NrKlienta | Kwota   | DataZlozenia | DataZaplaty |
+-------+-----------+---------+--------------+-------------+
|    87 |         3 | 3763.55 | 2008-04-03   | 2008-04-12  |
|   127 |         9 |  9015.9 | 2008-04-02   | 2008-04-13  |
|    79 |        11 |  413.35 | 2008-04-01   | 2008-04-15  |
|    28 |         7 | 2190.72 | 2008-04-10   | 2008-04-16  |
|    39 |         1 | 5457.17 | 2008-04-15   | 2008-04-17  |
+-------+-----------+---------+--------------+-------------+
5 rows in set (0.23 sec)
```

CURTIME(), CURRENT_TIME(), CURRENT_TIME — bieżący czas

```
mysql> SELECT CURTIME();
+-----------+
| CURTIME() |
+-----------+
| 22:36:00  |
+-----------+
1 row in set (0.03 sec)
```

NOW(), CURRENT_TIMESTAMP(), CURRENT_TIMESTAMP, LOCALTIME(), LOCALTIME, LOCALTIMESTAMP(), LOCALTIMESTAMP — bieżąca data i czas

```
mysql> SELECT NOW();
+---------------------+
| NOW()               |
+---------------------+
| 2008-04-09 22:41:45 |
+---------------------+
1 row in set (0.00 sec)
mysql> SELECT CURRENT_TIMESTAMP();
+---------------------+
| CURRENT_TIMESTAMP() |
+---------------------+
| 2008-04-09 22:41:46 |
+---------------------+
1 row in set (0.00 sec)
mysql> SELECT LOCALTIMESTAMP;
+---------------------+
| LOCALTIMESTAMP      |
+---------------------+
| 2008-04-09 22:41:50 |
+---------------------+
1 row in set (0.00 sec)
```

`DATE()` — wybiera datę z argumentu typu DATETIME

`TIME()` — wybiera czas z argumentu typu DATETIME

```
mysql> SELECT DATE('2008-04-10 17:30:28') AS ToJestData;
+------------+
| ToJestData |
+------------+
| 2008-04-10 |
+------------+
1 row in set (0.00 sec)

mysql> SELECT TIME('2008-04-10 17:30:28') AS ToJestCzas;
+------------+
| ToJestCzas |
+------------+
| 17:30:28   |
+------------+
1 row in set (0.00 sec)
```

`DAYOFMONTH()`, `DAY()` — numer dnia w miesiącu

`DAYOFYEAR()` — numer dnia w roku

`DAYOFWEEK()` — numer dnia w tygodniu (domyślnie liczymy od niedzieli)

`DAYNAME()` — nazwa dnia tygodnia

```
mysql> SELECT DAYOFMONTH('2008-04-10');
+--------------------------+
| DAYOFMONTH('2008-04-10') |
+--------------------------+
|                       10 |
+--------------------------+
1 row in set (0.00 sec)

mysql> SELECT DAYOFYEAR('2008-04-10');
+-------------------------+
| DAYOFYEAR('2008-04-10') |
+-------------------------+
|                     101 |
+-------------------------+
1 row in set (0.00 sec)
```

```
mysql> SELECT DAYOFWEEK('2008-04-10');
+--------------------------+
| DAYOFWEEK('2008-04-10') |
+--------------------------+
|                        5 |
+--------------------------+
1 row in set (0.00 sec)

mysql> SELECT DAYNAME('2008-04-10');
+-----------------------+
| DAYNAME('2008-04-10') |
+-----------------------+
| Thursday              |
+-----------------------+
1 row in set (0.00 sec)
```

LAST_DAY() — ostatni dzień miesiąca z podanej daty

```
mysql> SELECT LAST_DAY('2008-02-10');
+------------------------+
| LAST_DAY('2008-02-10') |
+------------------------+
| 2008-02-29             |
+------------------------+
1 row in set (0.00 sec)
```

`YEAR()` — rok

`MONTH()` — miesiąc, `MONTHNAME()` — nazwa miesiąca

```
mysql> SELECT YEAR(CURDATE());
+-----------------+
| YEAR(CURDATE()) |
+-----------------+
|            2008 |
+-----------------+
1 row in set (0.00 sec)
mysql> SELECT MONTH(CURDATE());
+------------------+
| MONTH(CURDATE()) |
+------------------+
|                4 |
+------------------+
1 row in set (0.00 sec)
mysql> SELECT MONTHNAME(CURDATE());
+----------------------+
| MONTHNAME(CURDATE()) |
+----------------------+
| April                |
+----------------------+
1 row in set (0.01 sec)
```

`HOUR()` — godzina, `MINUTE()` — minuta, `SECOND()` — sekunda

```
mysql> SELECT HOUR('2008-04-10 17:30:28');
+-----------------------------+
| HOUR('2008-04-10 17:30:28') |
+-----------------------------+
|                          17 |
+-----------------------------+
1 row in set (0.00 sec)

mysql> SELECT MINUTE('2008-04-10 17:30:28');
+-------------------------------+
| MINUTE('2008-04-10 17:30:28') |
+-------------------------------+
|                            30 |
+-------------------------------+
1 row in set (0.00 sec)

mysql> SELECT SECOND('2008-04-10 17:30:28');
+-------------------------------+
| SECOND('2008-04-10 17:30:28') |
+-------------------------------+
|                            28 |
+-------------------------------+
1 row in set (0.00 sec)
```

`MAKEDATE(rok,dzień_roku)` — utwórz datę

```
mysql> SELECT MAKEDATE(2007,365) AS Sylwester, MAKEDATE(2008,365) AS JeszczeNie;
+------------+------------+
| Sylwester  | JeszczeNie |
+------------+------------+
| 2007-12-31 | 2008-12-30 |
+------------+------------+
1 row in set (0.00 sec)
```

`MAKETIME(godzina,minuta,sekunda)` — utwórz godzinę

```
mysql> SELECT MAKETIME(18,20,47);
+--------------------+
| MAKETIME(18,20,47) |
+--------------------+
| 18:20:47           |
+--------------------+
1 row in set (0.00 sec)
```

Dodawanie dat

Uwaga: Zwykłe dodanie liczby do daty spowoduje

- najpierw skonwertowanie daty do “zapisu liczbowego”, a następnie
- wykonanie dodawania

```
mysql> SELECT CURDATE(), CURDATE() + 0, CURDATE() + 30;
+------------+---------------+----------------+
| CURDATE()  | CURDATE() + 0 | CURDATE() + 30 |
+------------+---------------+----------------+
| 2008-04-09 |      20080409 |       20080439 |
+------------+---------------+----------------+
1 row in set (0.00 sec)
```

`DATE_ADD(data, INTERVAL wyrażenie jednostka)`

`DATE_SUB(data, INTERVAL wyrażenie jednostka)` — dodanie, odjęcie czegoś do (od) daty

```
mysql> SELECT DATE_ADD('2008-04-10', INTERVAL 3 DAY),
    -> DATE_ADD('2008-04-10', INTERVAL 2 WEEK);
+--------------------------------------+---------------------------------------+
| DATE_ADD('2008-04-10', INTERVAL 3 DAY) | DATE_ADD('2008-04-10', INTERVAL 2 WEEK) |
+--------------------------------------+---------------------------------------+
| 2008-04-13                           | 2008-04-24                            |
+--------------------------------------+---------------------------------------+
1 row in set (0.00 sec)
mysql> SELECT DATE_SUB('2008-04-10',INTERVAL 2 MONTH),
    -> DATE_SUB('2004-04-10', INTERVAL 13 HOUR);
+---------------------------------------+----------------------------------------+
| DATE_SUB('2008-04-10',INTERVAL 2 MONTH) | DATE_SUB('2004-04-10', INTERVAL 13 HOUR) |
+---------------------------------------+----------------------------------------+
| 2008-02-10                            | 2004-04-09 11:00:00                    |
+---------------------------------------+----------------------------------------+
1 row in set (0.00 sec)
```

`DATEDIFF(data1,data2)` — różnica dwu dat, wyrażona w dniach

```
mysql> SELECT DATEDIFF('2008-04-10','2007-11-29');
+-------------------------------------+
| DATEDIFF('2008-04-10','2007-11-29') |
+-------------------------------------+
|                                 133 |
+-------------------------------------+
1 row in set (0.00 sec)
```

SUBTIME(dataczas$_1$,czas$_2$) — różnica dwu czasów

```
mysql> SELECT SUBTIME('18:01:22','00:05:01'), SUBTIME('18:01:22','0:5:1');
+-------------------------------+----------------------------+
| SUBTIME('18:01:22','00:05:01') | SUBTIME('18:01:22','0:5:1') |
+-------------------------------+----------------------------+
| 17:56:21                      | 17:56:21                   |
+-------------------------------+----------------------------+
1 row in set (0.00 sec)
```

ADDTIME(czas$_1$,czas$_2$) — suma dwu czasów

```
mysql> SELECT ADDTIME('18:01:48','1:2:0'),ADDTIME('18:01:48','25');
+----------------------------+-------------------------+
| ADDTIME('18:01:48','1:2:0') | ADDTIME('18:01:48','25') |
+----------------------------+-------------------------+
| 19:03:48                   | 18:02:13                |
+----------------------------+-------------------------+
1 row in set (0.00 sec)
```

`TIMEDIFF(dataczas1,dataczas2)` — różnica dwu czasów

```
mysql> SELECT TIMEDIFF('2008-04-10 18:02:26','2008-04-09 22:42:18');
+----------------------------------------------------+
| TIMEDIFF('2008-04-10 18:02:26','2008-04-09 22:42:18') |
+----------------------------------------------------+
| 19:20:08                                           |
+----------------------------------------------------+
1 row in set (0.00 sec)

mysql> SELECT TIMEDIFF('18:02:26','22:42:18');
+-----------------------------------+
| TIMEDIFF('18:02:26','22:42:18') |
+-----------------------------------+
| -04:39:52                         |
+-----------------------------------+
1 row in set (0.00 sec)
```

Formatowanie daty i czasu

```
SELECT DATE_FORMAT('2008-04-10','%Y %m %d'); ⟶ 2008 04 10

SELECT DATE_FORMAT('2008-04-10','%Y-%m-%d'); ⟶ 2008-04-10

SELECT DATE_FORMAT('2008-04-10','%M %d, %Y'); ⟶ April 10, 2008

SELECT DATE_FORMAT('2008-04-10','%W, %d.%m.%y'); ⟶ Thursday, 10.04.08

SELECT DATE_FORMAT('2008-04-10 17:46:24','%H:%i'); ⟶ 17:46

SELECT TIME_FORMAT('17:46:24','%H:%i'); ⟶ 17:46

SELECT TIME_FORMAT('17:46:24','%g'); ⟶ 17:46:24

SELECT TIME_FORMAT('17:46:24','%h:%i:%s %p'); ⟶ 05:46:24 PM

SELECT TIME_FORMAT('17:46:24','%r'); ⟶ 05:46:24 PM
```

Wynikiem działania funckji `DATE_FORMAT()`, `TIME_FORMAT()` nie są wielkości typu datowego lub czasowego, ale *łańcuchy znaków*.

`STR_TO_DATE(łańcuch,format)` — odwrotność funkcji `DATE_FORMAT()`

```
SELECT STR_TO_DATE('20080410','%Y%m%d'); ⟶ 2008-04-10
SELECT STR_TO_DATE('April 10, 2008','%M %d, %Y'); ⟶ 2008-04-10
SELECT STR_TO_DATE('05:54 PM','%h:%i %p'); ⟶ 17:54:00
```

Wynikiem działania funkcji `STR_TO_DATE()` jest wielkość typu `DATETIME`.

Jeżeli którykolwiek z argumentów którejkolwiek podanej wyżej funkcji będzie niepoprawny, w szczególności jeżeli nie będzie odpowiadał poprawnej dacie/czasowi, w wyniku otrzymamy obiekt `NULL`.

Wyrażenia zawierające NULL

```
mysql> SELECT * FROM Zamowienia;
+-------+-----------+---------+--------------+-------------+
| NrZam | NrKlienta | Kwota   | DataZlozenia | DataZaplaty |
+-------+-----------+---------+--------------+-------------+
|     1 |         6 | 4543.78 | 2008-04-15   | 2008-04-22  |
|     2 |        11 | 4702.25 | 2008-01-11   | 2008-01-13  |
|     3 |         8 |  796.73 | 2008-03-08   | 2008-03-23  |
|     4 |        12 | 7298.23 | 2008-02-13   | NULL        |
|     5 |         5 | 5314.03 | 2008-03-27   | NULL        |
|     6 |        11 | 1122.14 | 2008-02-26   | 2008-02-28  |
.....................................................
+-------+-----------+---------+--------------+-------------+
150 rows in set (0.00 sec)
```

Widzimy, że w tabeli jest pewna ilość wierszy zawierających NULL. Spróbujmy je wypisać.

```
mysql> SELECT * FROM Zamowienia WHERE DataZaplaty = NULL;
Empty set (0.02 sec)

mysql> SELECT * FROM Zamowienia WHERE NOT DataZaplaty = NULL;
Empty set (0.01 sec)
```

Oops ☹

Dzieje się tak dlatego, że *wyrażenia* (logiczne, arytmetyczne) *zawierające* `NULL` *zawsze dają w wyniku* `NULL`.

```
mysql> SELECT 1+2,1+NULL,1=NULL;
+-----+--------+--------+
| 1+2 | 1+NULL | 1=NULL |
+-----+--------+--------+
|   3 |   NULL |   NULL |
+-----+--------+--------+
1 row in set (0.01 sec)
```

Do sprawdzania, czy jakieś wyrażenie przybiera wartość `NULL`, służy fuknkcja `ISNULL()`.

```
mysql> SELECT * FROM Zamowienia WHERE ISNULL(DataZaplaty);
+-------+-----------+---------+--------------+-------------+
| NrZam | NrKlienta | Kwota   | DataZlozenia | DataZaplaty |
+-------+-----------+---------+--------------+-------------+
|     4 |        12 | 7298.23 | 2008-02-13   | NULL        |
|     5 |         5 | 5314.03 | 2008-03-27   | NULL        |
|     7 |         7 | 2091.78 | 2008-02-02   | NULL        |
|    10 |        10 | 8033.76 | 2008-02-04   | NULL        |
|    11 |         4 | 5879.96 | 2008-02-20   | NULL        |
.....................................................
+-------+-----------+---------+--------------+-------------+
71 rows in set (0.41 sec)
mysql> SELECT * FROM Zamowienia WHERE NOT ISNULL(DataZaplaty);
+-------+-----------+---------+--------------+-------------+
| NrZam | NrKlienta | Kwota   | DataZlozenia | DataZaplaty |
+-------+-----------+---------+--------------+-------------+
|     1 |         6 | 4543.78 | 2008-04-15   | 2008-04-22  |
|     2 |        11 | 4702.25 | 2008-01-11   | 2008-01-13  |
|     3 |         8 |  796.73 | 2008-03-08   | 2008-03-23  |
|     6 |        11 | 1122.14 | 2008-02-26   | 2008-02-28  |
|     8 |         2 | 6757.77 | 2008-03-03   | 2008-04-28  |
|     9 |         6 |  3932.1 | 2008-03-30   | 2008-03-31  |
.....................................................
+-------+-----------+---------+--------------+-------------+
79 rows in set (0.01 sec)
```

Zmienne tymczasowe

SQL pozwala na definiowanie zmiennych tymczasowych — nazwa zmiennej tymczasowej zawsze zaczyna się od `@`. Zmienna istnieje dopóty, dopóki nie zostanie zakończone połączenie z serwerem. Procesy klienckie nie widzą zmiennych zdefiniowanych przez *inne* procesy.

```
mysql> SET @a=5;
Query OK, 0 rows affected (0.01 sec)

mysql> SELECT @a-2;
+------+
| @a-2 |
+------+
| 3    |
+------+
1 row in set (0.00 sec)
```

Zmiennej można przypisać wartość, która jest *wynikiem zapytania*:

```
mysql> SET @b=(SELECT Kwota FROM Zamowienia WHERE NrZam=116);
Query OK, 0 rows affected (0.89 sec)

mysql> SELECT @b, @a+@b;
+----------------+----------------+
| @b             | @a+@b          |
+----------------+----------------+
| 4456.419921875 | 4461.419921875 |
+----------------+----------------+
1 row in set (0.18 sec)

mysql> SELECT FORMAT(@b,2), FORMAT(@a+@b,2);
+--------------+----------------+
| FORMAT(@b,2) | FORMAT(@a+@b,2) |
+--------------+----------------+
| 4,456.42     | 4,461.42       |
+--------------+----------------+
1 row in set (0.05 sec)
```

Aliasy

W zapytaniu `SELECT` możemy nadawać kolumnom inne nazwy niż te, które występują w definicji tabeli. Noszą one nazwę *aliasów*.

```
mysql> Select NrZam AS Zamowienie, DataZlozenia AS 'Kiedy Przyszlo'
    -> FROM Zamowienia
    -> LIMIT 6;
+------------+----------------+
| Zamowienie | Kiedy Przyszlo |
+------------+----------------+
|          1 | 2008-04-15     |
|          2 | 2008-01-11     |
|          3 | 2008-03-08     |
|          4 | 2008-02-13     |
|          5 | 2008-03-27     |
|          6 | 2008-02-26     |
+------------+----------------+
6 rows in set (1.40 sec)
```

Niekiedy wygodnie jest użyć aliasu w dalszej części zapytania, ale nie można go użyć w kaluzuli `WHERE`...

```
mysql> SELECT NrZam, Kwota AS Wartosc FROM Zamowienia
    -> WHERE Wartosc > 9000.0;
ERROR 1054 (42S22): Unknown column 'Wartosc' in 'where clause'
```

...ale można go użyć w klauzuli `HAVING`

```
mysql> SELECT NrZam, Kwota AS Wartosc FROM Zamowienia
    -> HAVING Wartosc > 9000.0
    -> ORDER BY Wartosc DESC LIMIT 7;
+-------+---------+
| NrZam | Wartosc |
+-------+---------+
|    58 | 9974.23 |
|    55 | 9963.39 |
|   141 | 9838.69 |
|    89 | 9790.21 |
|    13 | 9774.76 |
|   125 | 9674.87 |
|    27 | 9669.91 |
+-------+---------+
7 rows in set (0.12 sec)
```

Jest to szczególnie wygodne, gdy w warunku występuje wartość obliczna na podstawie więcej niż jednej kolumny:

```
mysql> SELECT NrZam, DATEDIFF(DataZaplaty,DataZlozenia) AS Roznica, Kwota
    -> FROM Zamowienia
    -> HAVING Roznica > 60;
+-------+---------+---------+
| NrZam | Roznica | Kwota   |
+-------+---------+---------+
|    31 |     108 | 5713.96 |
|    55 |      72 | 9963.39 |
|    58 |      73 | 9974.23 |
|    64 |     115 | 1829.71 |
|    71 |      79 | 6292.35 |
|    83 |      74 | 8057.01 |
|   118 |      94 | 7529.13 |
|   120 |     109 |  826.73 |
|   124 |      72 | 9016.46 |
+-------+---------+---------+
9 rows in set (0.06 sec)
```