# Funkcje operujące na datach i czasie

**MySQL** oferuje sporą liczbę funkcji służących do wykonywania operacji na datach i czasie, które mogą ułatwić życie programiście. Warto się z nimi zapoznać aby nie wykonywać niepotrzebnej pracy po stronie skryptu czy programu.

Przykłady dla wersji MySQL w wersji 5.1.x, aczkolwiek większość z przykładów powinna działać bez problemów również w wersji 4.1.x.

Na początek poznajmy funkcje `CURDATE()` i `CURTIME()`, zwracające odpowiednio aktualną datę i aktualny czas. Nie wyróżniają się one niczym szczególnym, poza tym, że mogą również zwracać datę lub czas w postaci liczbowej (przy wywołaniu `CURDATE() + 0`). Połączeniem obu tych funkcji jest funkcja `NOW()`, która zwraca datę i czas. Aby otrzymać uniksowy znacznik czasu (Unix timestamp) należy użyć funkcji `UNIX_TIMESTAMP()`, która może być również wywoływana z parametrem w postaci daty (zwraca wtedy timestamp dla podanej daty).

### Działania na datach i czasie

Jedną z najczęściej wykonywanych operacji na datach i czasie jest dodawanie lub odejmowanie. Za przykład weźmy najprostszą sytuację, w której do daty chcemy dodać jeden dzień. Można to zrobić na kilka sposobów:

```
mysql> SELECT ADDDATE('2010-01-01', INTERVAL 1 DAY);
+---------------------------------------+
| 2010-01-02                            |
+---------------------------------------+

mysql> SELECT DATE_ADD('2010-01-01', INTERVAL 1 DAY);
+----------------------------------------+
| 2010-01-02                             |
+----------------------------------------+

mysql> SELECT '2010-01-01' + INTERVAL 1 DAY;
+------------------------------+
| 2010-01-02                   |
+------------------------------+
```

Oczywiście datę, na której wykonujemy operację, możemy wziąć z pola w tabeli lub przyjąć datę bieżącą (`CURDATE`).

Żeby bardziej skomplikować sprawę, odejmowanie jednego dnia od jakiejś daty można zrobić na cztery sposoby (w działaniu nie różnią się one w ogóle):

```
mysql> SELECT ADDDATE('2010-01-01', INTERVAL -1 DAY);
+---------------------------------------+
| 2009-12-31                            |
+---------------------------------------+

mysql> SELECT DATE_ADD('2010-01-01', INTERVAL -1 DAY);
+----------------------------------------+
| 2009-12-31                             |
+----------------------------------------+
```

```
mysql> SELECT '2010-01-01' - INTERVAL 1 DAY;
+-----------------------------+
| 2009-12-31                  |
+-----------------------------+

mysql> SELECT DATE_SUB('2010-01-01', INTERVAL 1 DAY);
+--------------------------------------+
| 2009-12-31                           |
+--------------------------------------+
```

Warto przyjrzeć się bliżej klauzuli `INTERVAL`, która określa rodzaj interwału używanego przy operacji na dacie lub czasie. W powyższych przykładach mamy `DAY` (dzień), ale nie jest to jedyna dostępna wartość. Do dyspozycji mamy następujące rodzaje interwałów:

| | | |
|---|---|---|
| `MICROSECOND` | mikrosekundy | MICROSECONDS |
| `SECOND` | sekundy | SECONDS |
| `MINUTE` | minuty | MINUTES |
| `HOUR` | godziny | HOURS |
| `DAY` | dni | DAYS |
| `WEEK` | tygodnie | WEEKS |
| `MONTH` | miesiące | MONTHS |
| `QUARTER` | kwartały | QUARTERS |
| `YEAR` | lata | YEARS |
| `SECOND_MICROSECOND` | sekundy i mikrosekundy | 'SECONDS.MICROSECONDS' |
| `MINUTE_MICROSECOND` | minuty, sekundy i mikrosekundy | 'MINUTES:SECONDS.MICROSECONDS' |
| `MINUTE_SECOND` | minuty i sekundy | 'MINUTES:SECONDS' |
| `HOUR_MICROSECOND` | godziny, minuty, sekundy i mikrosekundy | 'HOURS:MINUTES:SECONDS.MICROSECONDS' |
| `HOUR_SECOND` | godziny, minuty i sekundy | 'HOURS:MINUTES:SECONDS' |
| `HOUR_MINUTE` | godziny i minuty | 'HOURS:MINUTES' |
| `DAY_MICROSECOND` | dni, godziny, minuty, sekundy i mikrosekundy | 'DAYS HOURS:MINUTES:SECONDS.MICROSECONDS' |
| `DAY_SECOND` | dni, godziny, minuty i sekundy | 'DAYS HOURS:MINUTES:SECONDS' |
| `DAY_MINUTE` | dni i minuty | 'DAYS HOURS:MINUTES' |
| `DAY_HOUR` | dni i godziny | 'DAYS HOURS' |
| `YEAR_MONTH` | lata i miesiące | 'YEARS-MONTHS' |

Jak widać, możliwości są bardzo duże i właściwie do każdego zastosowania możemy znaleźć coś odpowiedniego. Oto kilka przykładów:

```
mysql> SELECT '2010-01-01 23:59:59' + INTERVAL 1 SECOND;
+-----------------------------------------+
| 2010-01-02 00:00:00                     |
+-----------------------------------------+

mysql> SELECT '2010-01-01 00:00:00' + INTERVAL '2:5:30' HOUR_SECOND;
+-------------------------------------------------------+
| 2010-01-01 02:05:30                                   |
+-------------------------------------------------------+

mysql> SELECT '2010-01-01 00:00:00' + INTERVAL 2 WEEK;
+---------------------------------------+
| 2010-01-15 00:00:00                   |
+---------------------------------------+
```

```
mysql> SELECT '2010-01-01 00:00:00' + INTERVAL '2-5' YEAR_MONTH;
+-----------------------------------------------+
| 2012-06-01 00:00:00                           |
+-----------------------------------------------+
```

Wykonując operacje na datach trzeba uważać na odpowiedni typ argumentu, szczególnie gdy jest on np. wynikiem działania skryptu.

```
mysql> SELECT '2010-01-01' + INTERVAL 1.5 MINUTE_SECOND;
+-----------------------------------------+
| 2010-01-01 00:01:05                     |
+-----------------------------------------+

mysql> SELECT '2010-01-01' + INTERVAL 3/2 MINUTE_SECOND;
+-----------------------------------------+
| 2010-01-01 01:24:20                     |
+-----------------------------------------+
```

3/2 daje nam 1.5, tak więc teoretycznie zapytanie powinno przesunąć podaną datę o 1 minutę i 5 sekund, tak jak ma to miejsce w pierwszym przykładzie. Tak się jednak nie dzieje, ponieważ MySQL interpretuje wynik dzielenia 3/2 jako 1.5000, a to daje 1 minutę i 5000 sekund. Aby takie działanie dawało wynik zgodny z naszymi oczekiwaniami należy wymusić odpowiedni format za pomocą funkcji `CAST()`.

```
mysql> SELECT '2010-01-01' + INTERVAL CAST(3/2 AS DECIMAL(3,1))
MINUTE_SECOND;
+-------------------------------------------------------------------+
| 2010-01-01 00:01:05                                               |
+-------------------------------------------------------------------+
```

MySQL radzi sobie z miesiącem lutym i latami przestępnymi. Wykonuje również automatycznie niezbędnej konwersji typów z `DATE` na `DATETIME`. W przypadku, gdy podana data jest niepoprawna (np. '2010-01-00'), jako wynik otrzymamy `NULL`.

**Obliczanie różnicy pomiędzy datami**

Funkcje `DATEDIFF()` i `TIMEDIFF()` służą do obliczania różnicy pomiędzy datami. Różnią się typem przyjmowanych argumentów oraz wynikiem jaki zwracają. DATEDIFF() zwraca wynik w postaci liczby dni, na przykład:

```
mysql> SELECT DATEDIFF('2010-05-01', '2011-04-01');
+-------------------------------------+
|                              -335 |
+-------------------------------------+
```

`TIMEDIFF()` zwraca natomiast wynik w postaci czasu, na przykład:

```
mysql> SELECT TIMEDIFF('2010-05-01 00:00:00', '2011-04-01 00:00:00');
+-----------------------------------------------------+
| -838:59:59                                          |
+-----------------------------------------------------+
```

Funkcja ta przyjmuje jako argumenty zarówno wartości typu `DATETIME` jak i `TIME`, ale oba muszą mieć ten sam typ.

### Formatowanie daty i czasu

Do formatowania daty i czasu służy funkcja `DATE_FORMAT()`. Ma ona dwa parametry: datę i/lub czas, która zostanie sformatowana, oraz opis formatu. Sposób, w jaki zapisuje się format daty i czasu, nie powinien być zaskakujący dla większości osób zajmujących się programowaniem. Dostępne są następujące znaczniki:

| | |
|---|---|
| `%a` | Skrócona nazwa dnia tygodnia (Sun..Sat / niedz..sob) |
| `%b` | Skrócona nazwa miesiące (Jan..Dec / sty..gru) |
| `%c` | Miesiąc, liczbowo (0..12) |
| `%D` | Dzień miesiąca z angielskim sufiksem (0th, 1st, 2nd, 3rd, …) |
| `%d` | Dzień miesiąca, liczbowo (00..31) |
| `%e` | Dzień miesiąca, liczbowo (0..31) |
| `%f` | Mikrosekundy (000000..999999) |
| `%H` | Godzina (00..23) |
| `%h` | Godzina (01..12) |
| `%I` | Godzina (01..12) |
| `%i` | Minuty, liczbowo (00..59) |
| `%j` | Dzień w roku (001..366) |
| `%k` | Godzina (0..23) |
| `%l` | Godzina (1..12) |
| `%M` | Nazwa miesiąca (January..December / styczeń..grudzień) |
| `%m` | Miesiąc, liczbowo (00..12) |
| `%p` | AM lub PM |
| `%r` | Czas, zapis 12-godzinny (gg:mm:ss z AM lub PM) |
| `%S` | Sekundy (00..59) |
| `%s` | Sekundy (00..59) |
| `%T` | Czas, zapis 24-godzinny (hh:mm:ss) |
| `%U` | Tydzień (00..53), gdzie niedziela jest pierwszym dniem tygodnia |
| `%u` | Tydzień (00..53), gdzie poniedziałek jest pierwszym dniem tygodnia |
| `%V` | Tydzień (01..53), gdzie niedziela jest pierwszym dniem tygodnia; używane z %X |
| `%v` | Tydzień (01..53), gdzie poniedziałek jest pierwszym dniem tygodnia; używane z %x |
| `%W` | Nazwa dnia tygodnia (Sunday..Saturday / niedziela..sobota) |
| `%w` | Dzień tygodnia (0=niedziela..6=sobota) |
| `%X` | Rok dla tygodnia, w którym niedziela jest pierwszym dniem tygodnia, liczbowo, cztery cyfry; używany z %V |
| `%x` | Rok dla tygodnia, w którym poniedziałek jest pierwszym dniem tygodnia, liczbowo, cztery cyfry; używany z %v |
| `%Y` | Rok, liczbowo, cztery cyfry |
| `%y` | Rok, liczbowo, dwie cyfry |
| `%%` | Znak "%" |

Za język, w jakim zwracane są nazwy miesięcy i dni tygodnia odpowiada zmienna systemowa `lc_time_names`. Jej wartość można zmienić "w locie" używając funkcji `SET`:

```
mysql> SET lc_time_names = 'pl_PL';

mysql> SELECT DATE_FORMAT('2010-02-11', '%W %M %Y');
+--------------------------------------+
| czwartek luty 2010                   |
+--------------------------------------+
```

Do szybkiej zmiany formatu daty i/lub czasu można wykorzystać funkcję `GET_FORMAT()`, która zwraca jeden z predefiniowanych formatów. Dostępne są formatowania daty (`DATE`), czasu (`TIME`) oraz daty i czasu (`DATETIME`). Jako drugi parametr funkcja przyjmuje region, którego formatowanie dotyczy, na przykład:

```
mysql> SELECT GET_FORMAT(DATE, 'EUR');
+-------------------------+
| %d.%m.%Y                |
+-------------------------+

mysql> SELECT GET_FORMAT(DATE, 'USA');
+-------------------------+
| %m.%d.%Y                |
+-------------------------+
```

Często przydatne okazują się funkcje do operowania na uniksowych znacznikach czasu (Unix timestamp). `UNIX_TIMESTAMP()` konwertuje datę na timestamp, a `FROM_UNIXTIME()` wykonuje operację odwrotną.

**Wyciąganie fragmentów daty i czasu**

Bardzo często zachodzi potrzeba wyciągnięcia jedynie fragmentu daty zapisanej w bazie danych. Przydaje się to do grupowań (np. według roku i miesiąca) albo po prostu do wyświetlania w określony sposób na stronie. MySQL oferuje sporo funkcji przeznaczonych do tego typu operacji.

*Wszystkie poniższe funkcje jako parametr przyjmują datę.*

`DAY()` (synonim `DAYOFMONTH()`) - zwraca dzień (liczbowo)
`DAYNAME()` - zwraca nazwę dnia tygodnia (lokalizacja za pomocą zmiennej `lc_time_names`)
`DAYOFWEEK()` - zwraca dzień tygodnia (liczbowo)
`DAYOFYEAR()` - zwraca dzień w roku
`HOUR()` - zwraca liczbę godzin
`LAST_DAY()` - zwraca ostatni dzień miesiąca w formacie `DATE`
`MINUTE()` - zwraca liczbę minut
`MONTH()` - zwraca miesiąc (liczbowo)
`MONTHNAME()` - zwraca nazwę miesiąca (lokalizacja za pomocą zmiennej `lc_time_names`)
`QUARTER()` - zwraca kwartał (liczbowo)
`SECOND()` - zwraca liczbę sekund
`WEEK()` - zwraca numer tygodnia w roku
`WEEKDAY()` - zwraca numer dnia tygodnia (0 - poniedziałek, 6 - niedziela)
`YEAR()` - zwraca rok

Istnieje jeszcze dosyć uniwersalna funkcja `EXTRACT()`, z której korzysta się następująco:

```
mysql> SELECT EXTRACT(YEAR_MONTH FROM '2010-05-12 23:55:09');
+-----------------------------------------------+
|                                        201005 |
+-----------------------------------------------+
```

Jako pierwszy argument funkcja przyjmuje te same wartości co opisana wcześniej funkcja `DATE_ADD()`.

Więcej informacji na temat funkcji operujących na dacie i czasie znaleźć można w dokumentacji MySQL.

### Ćwiczenie 1

W tabeli **Rejestry** oblicz różnicę, ile czasu (np. ile dni) minęło od sprzedaży samochodu o ID=1 i ID=2 Wynik wyświetl w polu obliczeniowym o nagłówku **różnica**

### Ćwiczenie 2

Ile lat, miesięcy, dni minęło od dnia dzisiejszego od sprzedaży wszystkich samochodów z tabeli **Rejestry**. Wyniki wyświetl w polach obliczeniowych o nagłówkach: **rok**, **miesiąc**, **dzień**

### Ćwiczenie 3

Do tabeli Klienci dodaj nową kolumnę **data_urodzenia** (ALTER TABLE) typu **date** wypełnij ją przykładowymi danymi, a następnie wyświetl **imię**, **nazwisko** oraz jako pole wyliczeniowe **wiek**, w którym będzie wyliczony aktualny wiek (w latach) klienta liczony od bieżącej daty.