

Bazy danych



Andrzej Łachwa, UJ, 2015

andrzej.lachwa@uj.edu.pl

2/14

Języki SQL, SQL 2 (1992), SQL 3 (1999-2002)

<https://pl.wikipedia.org/wiki/SQL>

SQLite

<https://www.sqlite.org/>

MySQL

<https://www.mysql.com/>

<http://download.komputerswiat.pl/akcesoria-i-narzedzia/programy-na-pendrive/mysql-community-server-portable>

Dane dotyczące czasu

Typy danych czasowych pojawiły się w SQL 2. Opis ich jest tak rozbudowany, że nikt go jeszcze nie zaimplementował!

Dlaczego jest to tak trudne?

1

Rok słoneczny składa się z ok. 365,2422 dni i trwa coraz dłużej. Każdy kalendarz ma całkowitą liczbę dni w roku. Trzeba więc co pewien czas dokonać wyrównania kalendarza z czasem słonecznym.

2

Pierwsze ważne wyrównanie kalendarza z czasem słonecznym nastąpiło za Juliusza Cezara (stąd mówimy o kalendarzu juliańskim) w 46 p.n.e. (ten rok miał 445 dni). Wprowadził on lata przestępne co 4 lata (niestety przez pomyłkę kilka pierwszych lat przestępnych przypadało co 3 lata). Pewne są natomiast lata przestępne od 5 roku n.e. i stąd dopiero od tej daty można wyliczać dni tygodnia.

Kalendarz juliański spóźniał się o 1 dzień na 128 lat i dlatego zastąpiono go w 1582 kalendarzem gregoriańskim (spóźnienie wynosiło wtedy już około 10 dni, więc w roku 1582 usunięto daty od 5 do 14 października).

3

Kalendarz juliański obowiązywał w różnych krajach w różnych okresach, np. w Rosji od 1700 do 1918, w Polsce tylko do 1582, a w Grecji aż do 1923.

Zmiana kalendarza na gregoriański wprowadzana była w różnych krajach w różnym czasie, np. w Wielkiej Brytanii korektę przeprowadzono dopiero w 1752.

4

Przez wiele stuleci dodatkową komplikacją było rozpoczynanie roku w różnym czasie, np. w Wielkiej Brytanii i koloniach początek roku przypadał na 25 marca (październik to zatem miesiąc ósmy, a grudzień – dziesiąty: po łacinie *octo* to osiem, *decem* – dziesięć).

WNIOSEK

Historyczne daty zwykle nie odpowiadają datom obliczonym dzisiaj według kalendarza gregoriańskiego. Dotyczy to również dni tygodnia i świąt. Niektóre daty w ogóle nie istnieją, podobnie jak nie istnieje 29 lutego 2014.

5

FORMATY DAT

Dzień 12/16/95 w Bostonie to 16/12/95 w Londynie,
16.12.95 w Berlinie i 95-12-16 w Sztokholmie.

W NATO próbowano stosować liczby rzymskie na oznaczanie miesięcy, więc powyższa data to 16 XII 1995. Z kolei w armii amerykańskiej stosowano format *YYYY mmm DD*, gdzie *mmm* to pierwsze 3 litery angielskiej nazwy miesiąca, zatem: 1995 JAN 16.

Ponadto istnieje wiele standardów przemysłowych.

W elektronicznym przetwarzaniu informacji używa się normy ISO 8601.

6

CZAS LOKALNY

Czas uniwersalny oznacza się jako UTC. Co jakiś czas dodawana jest tzw. sekunda przestępna przez *International Earth Rotation Service*.

Strefy czasowe zmieniały się w niektórych krajach co 15 minut, ale obecnie (od 1998) zmiany dotyczą całych godzin. W USA mamy 4 strefy czasowe oraz tzw. czas prawny.

7

CZAS LETNI

W USA używa się określenia DST (*Daylight Saving Time*),
w krajach środkowoeuropejskich mamy CET i CEST (*Central European Summer Time*).

8

Istnieje wiele formatów zapisywania czasu.

Zwykle w elektronicznym przetwarzaniu informacji stosuje się format zgodny z normą ISO-8601.

Jednak różni producenci serwerów bazodanowych wprowadzają dodatkowe formaty.

Daty i czas wg ISO 8601

YYYY-MM-DD (data kalendarzowa, format rozszerzony)

YYYYMMDD (jw, format podstawowy)

YYYY-DDD lub **YYYYDDD** (data porządkowa, **DDD** to kolejny dzień w roku w formacie 3-cyfrowym)

YYYY-Www-D (data tygodniowa, **W** to symbol stały, **ww** to numer tygodnia w roku, **D** to numer dnia tygodnia, np. dziś to **2014W482** (48 tydzień roku, wtorek).

hh:mm:ss, **hh:mm** (czas, format rozszerzony)

hhmmss, **hhmm** i **hh** (czas, format podstawowy, po liczbie sekund może wystąpić część ułamkowa)

2013:11:19 T 10:40.85+01:00 (łączy zapis daty i czasu)

Uwagi:

rok **0001** to 1 rok n.e., a **0000** to 1 rok p.n.e. (1 B.C.)

północ w Sylwestra to **24:00** dnia 31 grudnia albo **00:00** dnia 1 stycznia

w **dacie porządkowej** 3 lutego to zawsze **034**, ale 31 grudnia to albo **365** albo **366**)

pierwszy tydzień roku to taki, w którym są co najmniej 4 dni stycznia, więc 30 grudnia 2014 to **2015-W01-2**

czas odmierza się zgodnie ze standardem UTC, a nie poprzednio stosowanym GMT, czas lokalny oznacza się przez wskazanie przesunięcia, np. w zimie w Polsce godzina 10:40 to **10:40+01** i jest równa **9:40Z**

Daty i czas w SQL

Standard SQL ma typy danych związane z datą i czasem:

DATE, TIME, TIMESTAMP, INTERVAL

Interwały typu rok-miesiąc mają domyślną precyzję obejmującą pola YEAR i MONTH. Interwały typu dzień-czas mają precyzję DAY, HOUR, MINUTE i SECOND (z częścią ułamkową).

Standard obejmuje też pełny zestaw operatorów dla czasowych typów danych (w różnych produktach pełna składnia i funkcjonalność nie jest implementowana).

Koniec czasu to w SQL data ***9999-12-31***.

Daty i czas w MySQL

DATE ma zakres od **1000-01-01** do **9999-12-31** oraz **0000-00-00**
(zakres DATETIME kończy się więc na **9999-12-31 23:59:59**)

TIMESTAMP ma zakres jak DATETIME, ale jest zapisywany jako liczba sekund od początku zakresu i nie przyjmuje wartości początku przedziału (wartość 0 jest interpretowana jako **0000-00-00 00:00:00**).

TIME ma zakres od **-838:59:59** do **838:59:59** (w tym zero).

YEAR ma zakres od **1901** do **2155** oraz **0000** (ale w formacie 2-cyfrowym od 1970 do 2069, więc 00 to 2000).

Przykłady

```
SELECT ADDDATE('2008-01-02', INTERVAL 31 DAY);  
-> '2008-02-02'
```

```
SELECT ADDTIME('2007-12-31 23:59:59.999999',  
'1 1:1:1.000002');  
-> '2008-01-02 01:01:01.000001'
```

```
SELECT DATE_FORMAT('2009-10-04 22:23:00', '%W %M %Y');  
-> 'Sunday October 2009'
```

```
SELECT DATE_FORMAT('1999-01-01', '%X %V');  
-> '1998 52'
```



```
SELECT NOW() , SLEEP(2) , NOW() \G
```

```
-> 2006-04-12 13:47:36
```

```
0
```

```
2006-04-12 13:47:36
```

```
SELECT SYSDATE() , SLEEP(2) , SYSDATE() \G
```

```
-> 2006-04-12 13:47:44
```

```
0
```

```
2006-04-12 13:47:46
```

```
SELECT WEEKDAY('2008-02-03 22:23:00') ;
```

```
-> 6
```

```
SELECT MAKETIME(12,15,30) ;
```

```
-> '12:15:30'
```

```
SELECT CURRENT_DATE(), DATEDIFF(CURRENT_DATE(),  
'1967-01-26') AS "Zyje juz tyle dni!";
```

```
+-----+-----+  
| CURRENT_DATE() | Zyje juz tyle dni! |  
+-----+-----+  
| 2006-04-26      |                  14335 |  
+-----+-----+
```

```
SELECT CONCAT('Dzisiaj mamy ', DAYOFYEAR(NOW()),  
' dzien roku, ', WEEKOFYEAR(NOW()),  
' tydzien roku oraz ', DAYOFWEEK(NOW()),  
' dzien tygodnia. ');
```

```
+-----+  
| Dzisiaj mamy 116 dzien roku, 17 tydzien roku oraz 4 dzien tygodnia. |  
+-----+
```

```
SELECT CURRENT_TIMESTAMP,
       DATE_FORMAT(CURRENT_TIMESTAMP, '%W :: %M :: %d
       :: %Y :: %T') AS "biezaca data i godzina";
```

```
+-----+-----+
| CURRENT_TIMESTAMP | biezaca data i godzina |
+-----+-----+
| 2006-03-28 11:57:20 | Tuesday :: March :: 28 :: 2006 :: 11:57:20 |
+-----+-----+
```

Funkcje SUM() i AVG() nie działają na datach i czasie, ale można sobie z tym poradzić, np.:

```
SELECT SEC_TO_TIME(SUM(TIME_TO_SEC(time_col)))
FROM ...
```

Przykłady w SQLite3 i MySQL

```
sqlite>
sqlite>
sqlite>
sqlite> select date();
2014-10-02
sqlite> select time();
19:55:10
sqlite> select datetime();
2014-10-02 19:55:12
sqlite> select datetime('now','localtime');
2014-10-02 21:55:24
sqlite>
sqlite>
sqlite> select strftime('%s','now');
1412279735
sqlite> select strftime('%f','now');
37.926
sqlite> select datetime(),strftime('%f','now');
2014-10-02 19:55:51.51479
sqlite>
sqlite>
sqlite> select strftime('%J','1970-01-01 01:00');
2440587.541666667
sqlite>
sqlite>
sqlite>
sqlite>
```

```
sqlite>
sqlite>
sqlite>
sqlite>
sqlite>
sqlite>
sqlite> select datetime('now');
2014-09-30 09:30:01
sqlite>
sqlite> select datetime('now','localtime');
2014-09-30 11:30:06
sqlite>
sqlite>
sqlite>
sqlite> select strftime('%W','now');
39
sqlite>
sqlite>
sqlite> select strftime('%m-%W','now');
09-39
sqlite>
sqlite>
sqlite>
```

```
sqlite>
sqlite> create table foo(a int,b timestamp default current_timestamp);
sqlite> insert into foo(a) values (35);
sqlite> insert into foo(a) values (36);
sqlite> insert into foo(a) values (37);
sqlite> select * from foo;
35|2014-10-02 18:39:40
36|2014-10-02 18:39:47
37|2014-10-02 18:39:54
sqlite>
sqlite>
sqlite> select current_timestamp;
2014-10-02 18:40:31
sqlite>
sqlite> insert into foo values (38,'2014-10-02 15:10:10');
sqlite> select * from foo;
35|2014-10-02 18:39:40
36|2014-10-02 18:39:47
37|2014-10-02 18:39:54
38|2014-10-02 15:10:10
sqlite>
sqlite>
sqlite> update foo set a=a+10 where a=38;
sqlite> select * from foo;
35|2014-10-02 18:39:40
36|2014-10-02 18:39:47
37|2014-10-02 18:39:54
48|2014-10-02 15:10:10
sqlite>
sqlite>
```

```
mysql>
mysql>
mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| kodekswykroczen |
| mysql |
| performance_schema |
| phpmyadmin |
| test |
| world |
+-----+
7 rows in set (0.06 sec)

mysql> create table F00(NUM int, DATE timestamp);
Query OK, 0 rows affected (0.30 sec)

mysql> select * from F00;
Empty set (0.00 sec)

mysql> insert into F00(NUM) values (34);
Query OK, 1 row affected (0.01 sec)

mysql> insert into F00(NUM) values (35);
Query OK, 1 row affected (0.01 sec)

mysql> select * from F00;
+-----+
| NUM | DATE |
+-----+
| 34 | 2014-10-04 10:38:52 |
| 35 | 2014-10-04 10:38:56 |
+-----+
2 rows in set (0.00 sec)

mysql>
```



```
mysql> select * from F00;
```

NUM	DATE
34	2014-10-04 10:38:52
35	2014-10-04 10:38:56
NULL	2014-10-11 00:00:00
NULL	2014-10-04 10:42:50

```
4 rows in set (0.00 sec)
```

```
mysql> update F00 set NUM=NUM+10 where NUM<50;
```

```
Query OK, 2 rows affected (0.02 sec)
```

```
Rows matched: 2 Changed: 2 Warnings: 0
```

```
mysql> select * from F00;
```

NUM	DATE
44	2014-10-04 10:45:10
45	2014-10-04 10:45:10
NULL	2014-10-11 00:00:00
NULL	2014-10-04 10:42:50

```
4 rows in set (0.00 sec)
```

```
mysql> update F00 set NUM=56 where NUM is null;
```

```
Query OK, 2 rows affected (0.00 sec)
```

```
Rows matched: 2 Changed: 2 Warnings: 0
```

```
mysql> select * from F00;
```

NUM	DATE
44	2014-10-04 10:45:10
45	2014-10-04 10:45:10
56	2014-10-04 10:46:36
56	2014-10-04 10:46:36

```
4 rows in set (0.00 sec)
```

```
mysql>
```

mysql> select * from F00;

NUM	DATE
44	2014-10-04 10:45:10
45	2014-10-04 10:45:10
56	2014-10-04 10:46:36
56	2014-10-04 10:46:36

4 rows in set (0.00 sec)

mysql> update F00 set DATE=now() where NUM<50;

Query OK, 2 rows affected (0.01 sec)

Rows matched: 2 Changed: 2 Warnings: 0

mysql> select * from F00;

NUM	DATE
44	2014-10-04 10:49:54
45	2014-10-04 10:49:54
56	2014-10-04 10:46:36
56	2014-10-04 10:46:36

4 rows in set (0.00 sec)

mysql>

mysql> select * from F00;

NUM	DATE
44	2014-10-04 10:45:10
45	2014-10-04 10:45:10
56	2014-10-04 10:46:36
56	2014-10-04 10:46:36

4 rows in set (0.00 sec)

mysql> update F00 set DATE=now() where NUM<50;

Query OK, 2 rows affected (0.01 sec)

Rows matched: 2 Changed: 2 Warnings: 0

mysql> select * from F00;

NUM	DATE
44	2014-10-04 10:49:54
45	2014-10-04 10:49:54
56	2014-10-04 10:46:36
56	2014-10-04 10:46:36

4 rows in set (0.00 sec)

mysql>

Wstęp do problematyki danych związanych z czasem:

Joe Celko: SQL. *Zaawansowane techniki programowania*. PWN
2008, rozdział 4 i rozdział 29

[Wiele problemów dotyczących czasu dla różnych dialektów języka SQL
rozwiązano i opublikowano, m.in. na stronie uniwersytetu
\[www.arizona.edu\]\(http://www.arizona.edu\).](#)