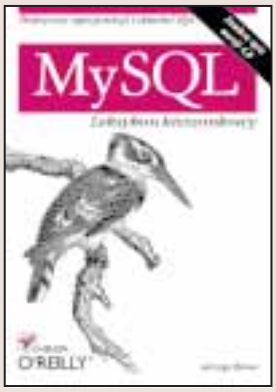
MySQL. Leksykon kieszonkowy



Autor: George Reese
Tłumaczenie: Tomasz Żmijewski
ISBN: 83-7361-164-9
Tytuł oryginału: MySQL Pocket Reference
Format: B5, stron: 104

Trudno znaleźć obecnie dziedzinę, w której nie jest używane oprogramowanie MySQL – najpopularniejszy na świecie system zarządzania bazą danych na licencji open source. Jest to bardzo solidna baza danych do stosowania w połączeniu z serwerami sieciowymi zawierająca szereg niespotykanych gdzie indziej instrukcji i funkcji, a jednocześnie obsługująca dużą część standardowej składni SQL.

Żaden administrator i programista nie jest w stanie nadążyć za mnogością opcji dostępnych w poszczególnych instrukcjach i funkcjach MySQL. Wiele poleceń niezastąpionych w pewnych sytuacjach ma skomplikowaną składnię – nawet najbardziej doświadczeni administratorzy i programiści miewają kłopoty z zapamiętaniem dokładnej postaci instrukcji. Niniejszy leksykon będzie stanowić dla Ciebie swoisty „niezbędnik", przypominający o składni, poleceniach i funkcjach MySQL.

Dzięki tej książce zaoszczędzisz swój cenny czas i przyspieszysz wykonanie zadania. Zawiera ona krótkie przypomnienie procesu instalacji, pełną składnię SQL-a dostępną w MySQL oraz opisuje wszystkie typy danych, operatory i funkcje. Niniejsza pozycja jest doskonałym podręcznym uzupełnieniem innych książek poświęconych MySQL i oprogramowaniu bazodanowemu.

Wydawnictwo Helion
ul. Chopina 6
44-100 Gliwice
tel. (32)230-98-63
e-mail: helion@helion.pl

Spis treści

Rozdział 3. Typy danych

We wszystkich typach danych nawiasami kwadratowymi ([]) oznaczane są fragmenty opcjonalne. Poniższy przykład pokazuje sposób prezentacji typu BIGINT, opisanego dalej w tym rozdziale:

```
BIGINT[(wielkość_pokazywana)]
```

Oznacza to, że słowo BIGINT może wystąpić samodzielnie lub z pokazywaną wartością. Użycie kursywy wskazuje, że nie należy wpisywać słowa *wielkość_pokazywana*, ale podać własną wartość. Oto przykłady użycia:

```
BIGINT
BIGINT(20)
```

Poza typem BIGINT także wiele innych typów danych MySQL uwzględnia deklarację rozmiaru wyświetlania. Jeśli nie powiedziano inaczej, musi to być liczba od 1 do 255.

W niektórych przypadkach MySQL zmienia podany typ kolumny, nie informując o tym użytkownika:

VARCHAR -> CHAR
: Jeśli podana kolumna VARCHAR ma rozmiar mniejszy od czterech znaków, jest przekształcana w kolumnę CHAR.

CHAR -> VARCHAR
: Jeśli tabela zawiera co najmniej jedną kolumnę o zmiennej długości, wszystkie kolumny typu CHAR dłuższe niż trzy znaki są zamieniane na VARCHAR.

Rozmiar wyświetlania TIMESTAMP
: Rozmiar wyświetlania pól TIMESTAMP musi być zawsze wielkością parzystą od 2 do 14. Rozmiar równy 0 lub większy od 14 powoduje przyjęcie 14. Wszelkie liczby nieparzyste są zamieniane na następną liczbę parzystą.

Liczby

MySQL obsługuje liczbowe typy danych zgodne z ANSI SQL 2. Typy te dzielimy na dwie grupy: całkowitoliczbowe i zmiennoprzecinkowe. W ramach tych grup dzielimy typy dalej, według zajmowanej przez nie pamięci.

W przypadku typów liczbowych można podać rozmiar wyświetlania, który wpływa na sposób pokazywania przez MySQL wyników. Rozmiar ten nie ma żadnego związku z wielkością pamięci zajmowanej przez dany typ. Dodatkowo w przypadku liczb zmiennoprzecinkowych można podać liczbę cyfr znajdujących się za kropką dziesiętną. Wtedy liczba cyfr powinna należeć do zakresu od 0 do 30, czyli być co najmniej o dwa mniejsza od rozmiaru wyświetlania. Jeśli warunek ten nie zostanie dotrzymany, MySQL automatycznie zmieni liczbę cyfr tak, aby była mniejsza o dwa od rozmiaru wyświetlania. Przykładowo, MySQL automatycznie zmieni `FLOAT(6,5)` na `FLOAT(7,5)`.

Próba wstawienia do kolumny wartości przekraczającej dopuszczalny zakres tej kolumny powoduje obcięcie tej wartości do najmniejszej (dla liczb ujemnych) lub największej (dla liczb dodatnich) wartości dla danej kolumny dopuszczalnej. Jeśli takie obcięcie jest robione podczas wykonywania instrukcji `ALTER TABLE`, `LOAD DATA INFILE`, `UPDATE` lub wielowierszowej instrukcji `INSERT`, MySQL pokazuje ostrzeżenie.

Atrybutu `AUTO_INCREMENT` można użyć do co najwyżej jednej kolumny całkowitoliczbowej w tabeli. Atrybut `UNSIGNED` może być łączony z dowolnym liczbowym typem danych. Użycie tego atrybutu powoduje, że do kolumny nie można wpisywać liczb ujemnych. Atrybut `ZEROFILL` nakazuje wypełnienie kolumny od lewej strony zerami podczas wyświetlania jej wartości. O liczbie tych zer decyduje szerokość wyświetlania danej kolumny.

BIGINT

```
BIGINT[(rozmiar_wyświetlany)] [AUTO_INCREMENT] [UNSIGNED] [ZEROFILL]
```

Rozmiar w pamięci 8 bajtów

Opis

Największy z typów całkowitoliczbowych, pozwalający zapisywać liczby od –9 223 372 036 854 775 808 do 9 223 372 036 854 775 807 (jeśli bez znaku, to od 0 do 18 446 744 073 709 551 615). MySQL wszelkie operacje arytmetyczne wykonuje, korzystając z wartości `BIGINT` lub `DOUBLE`, ale w przypadku `BIGINT` operacje robione są na liczbach bez znaku. Wobec tego należy unikać operacji na liczbach `BIGINT` bez znaku większych niż 9 223 372 036 854 775 807, gdyż może to zaowocować nieprawidłowymi wynikami.

DEC

Synonim typu `DECIMAL`.

DECIMAL

```
DECIMAL[(dokładność, [skala])] [ZEROFILL]
```

Rozmiar w pamięci *dokładność* +2 bajty

Opis

Pozwala zapisywać liczby zmiennoprzecinkowe w sytuacjach, kiedy istotna jest dokładność — na przykład przy operowaniu kwotami pieniędzy. Stosując typ `DECIMAL`, trzeba podać dwa jego parametry, dokładność i skalę. Dokładność to liczba znaczących cyfr, zaś skala to liczba znaczących cyfr po kropce dziesiętnej. Przykładowo, kolumna `SALDO` typu `DECIMAL(9,2)` pozwoliłaby zapisywać liczby dziewięciocyfrowe, przy czym na prawo od kropki dziesiętnej mogłyby być dwie cyfry. Zakres dopuszczalnych liczb to w takiej sytuacji od –9 999 999,99 do 9 999 999,99. Jeśli

podana zostanie liczba zawierająca więcej cyfr po przecinku, niż przewiduje to definicja, liczba zostanie zaokrąglona. Wartości spoza zakresu DECIMAL są obcinane tak, aby się w nim zmieściły.

MySQL zapisuje wartości DECIMAL nie jako liczby zmiennoprzecinkowe, lecz jako łańcuch znaków. Na każdą cyfrę zużywany jest jeden znak w przypadku skali większej od 0, poza tym jeden dodatkowy znak zużywany jest w przypadku liczb ujemnych. Jeśli skala wyniesie 0, liczby nie zawierają części ułamkowej. W MySQL w wersjach starszych niż 3.23 decydując o dokładności, trzeba było uwzględnić przecinek i znak liczby, aczkolwiek obecnie, zgodnie ze specyfikacją ANSI, nie jest to już wymagane.

SQL zgodnie z normą ANSI pozwala pomijać dokładność i (lub) skalę. Jeśli brak dokładności, przyjmowane jest ustawienie domyślne charakterystyczne dla implementacji. Jeśli brak skali, przyjmowane jest zero. W MySQL domyślna wartość dokładności to 10.

DOUBLE

```
DOUBLE[(rozmiar_wyświetlany, cyfr)] [ZEROFILL]
```

Rozmiar w pamięci 8 bajtów

Opis

Liczba zmiennoprzecinkowa podwójnej precyzji. Ten typ danych pozwala zapisywać duże wartości zmiennoprzecinkowe. W kolumnach tego typu można zapisać wartości ujemne od –1,7976931348623157E+308 do –2,2250738585072014E–308, 0 oraz wartości dodatnie od 2,2250738585072014E–308 do 1,7976931348623157E+308.

DOUBLE PRECISION

Synonim DOUBLE.

FLOAT

```
FLOAT[(rozmiar_wyświetlany, cyfr)] [ZEROFILL]
```

Rozmiar w pamięci 4 bajty

Opis

Liczba zmiennoprzecinkowa pojedynczej precyzji. Ten typ danych pozwala zapisywać małe wartości zmiennoprzecinkowe. W kolumnach tego typu można zapisać wartości ujemne od –3,402823466E+38 do –1,175494351E–38, 0 oraz wartości dodatnie od 1,175494351E–38 do 3,402823466E+38.

INT

```
INT[(rozmiar_wyświetlany)] [AUTO_INCREMENT] [UNSIGNED] [ZEROFILL]
```

Rozmiar w pamięci 4 bajty

Opis

Podstawowy rodzaj liczb całkowitych od –2 147 483 648 do 2 147 483 647 (lub od 0 do 4 294 967 295 w przypadku liczb bez znaku).

INTEGER

Synonim `INT`.

MEDIUMINT

```
MEDIUMINT[(rozmiar_wyświetlany)] [AUTO_INCREMENT] [UNSIGNED] [ZEROFILL]
```

Rozmiar w pamięci 3 bajty

Opis

Liczby całkowite od –8 388 608 do 8 388 607 (lub od 0 do 16 777 215 w przypadku liczb bez znaku).

NUMERIC

Synonim DECIMAL.

REAL

Synonim DOUBLE.

SMALLINT

```
SMALLINT[(rozmiar_wyświetlany)] [AUTO_INCREMENT] [UNSIGNED] [ZEROFILL]
```

Rozmiar w pamięci 2 bajty

Opis

Liczby całkowite z zakresu od –32 768 do 32 767 (od 0 do 65 535 w przypadku liczb bez znaku).

TINYINT

```
TINYINT[(rozmiar_wyświetlany)] [AUTO_INCREMENT] [UNSIGNED] [ZEROFILL]
```

Rozmiar w pamięci 1 bajt

Opis

Liczby całkowite od –128 do 127 (od 0 do 255 w przypadku liczb bez znaku).

Łańcuchy

Łańcuchowe typy danych pozwalają zapisywać różne rodzaje danych tekstowych. Różne typy pozwalają zapisywać różne ilości takich danych. W ramach każdej wielkości istnieje typ pozwalający sortować i porównywać znaki z domyślnego zestawu znaków z pominięciem wielkości liter. Odpowiedni typ binarny porównuje i sortuje dane po prostu bajt po bajcie, czyli typy

binarne rozróżniają wielkość liter. W przypadku typów CHAR i VARCHAR typy binarne deklaruje się za pomocą słowa kluczowego BINARY. Jeśli zaś chodzi o typy TEXT, istnieją ich odpowiedniki, BLOB.

BLOB

Binarny odpowiednik typu TEXT.

CHAR

```
CHAR(rozmiar) [BINARY]
```

Rozmiar według *rozmiar*, do 255 (w wersjach MySQL przed 3.23 od 1 do 255)

Rozmiar w pamięci *rozmiar* bajtów

Opis

Pole tekstowe ustalonej długości. Łańcuchy zawierające mniej znaków, niż to wynika z rozmiaru kolumny, wypełniane są po prawej stronie spacjami. Podczas pobierania danych z bazy te dodatkowe spacje są usuwane.

Pola CHAR(0) zostały zachowane w celu zapewnienia zgodności ze starymi systemami, w których w kolumnach nie są zapisywane żadne wartości.

CHARACTER

Synonim CHAR.

CHARACTER VARYING

Synonim VARCHAR.

LONGBLOB

Binarny odpowiednik `LONGTEXT`.

LONGTEXT

```
LONGTEXT
```

Rozmiar 0 do 4 294 967 295

Rozmiar w pamięci Długość wartości+4 bajty

Opis

Typ pozwala zapisywać duże wartości tekstowe. Teoretyczne ograniczenie rozmiaru tekstu to ponad 4 GB, ale praktycznymi ograniczeniami są ograniczenia protokołu komunikacyjnego MySQL oraz ilość pamięci przeznaczonej na komunikację na serwerze i na stacji klienckiej.

MEDIUMBLOB

Binarna postać `MEDIUMTEXT`.

MEDIUMTEXT

```
MEDIUMTEXT
```

Rozmiar 0 do 16 777 215

Rozmiar w pamięci Długość wartości+3 bajty

Opis

Typ pozwala zapisywać średniej wielkości wartości tekstowe.

NCHAR

Synonim `CHAR`.

NATIONAL CHAR

Synonim CHAR.

NATIONAL VARCHAR

Synonim VARCHAR.

TEXT

```
TEXT
```

Rozmiar 0 do 65 535

Rozmiar w pamięci Długość wartości tekstowej +2 bajty

Opis

Typ pozwala zapisywać typowe wartości tekstowe.

TINYBLOB

Binarny odpowiednik TINYTEXT.

TINYTEXT

```
TINYTEXT
```

Rozmiar 0 do 255

Rozmiar w pamięci Długość wartości tekstowej +1 bajt

Opis

Pozwala zapisywać krótkie dane tekstowe.

VARCHAR

```
VARCHAR(rozmiar) [BINARY]
```

Rozmiar	Wskazana przez *rozmiar* wartość z zakresu od 0 do 255 (od 1 do 255 w wersjach MySQL wcześniejszych niż 3.23)
Rozmiar w pamięci	Długość zapisywanej wartości +1 bajt

Opis

Pozwala zapisywać wartości tekstowe zmiennej długości. Z wartości VARCHAR usuwane są spacje końcowe.

Daty

Typy datowe MySQL są wyjątkowo elastycznym narzędziem, pozwalającym zapisywać wszelki informacje dzienne. MySQL jest bardzo tolerancyjny i zakłada, że to aplikacja, a nie baza danych, ma sprawdzać poprawność tych danych. MySQL sprawdza jedynie, czy miesiąc nie wykracza poza zakres 0 – 12 i czy dzień nie wykracza poza zakres 0 – 31. Wobec tego z punktu widzenia MySQL 31 lutego 2001 roku jest poprawną datą. Bardziej przydatną wartością jest data 0 lutego 2001 roku; cyfra zero może zastępować tę część daty, której dokładnie nie znamy.

Wprawdzie MySQL dopuszcza dość dużą swobodę formatów wejściowych dat, to należy starać się w aplikacjach daty formatować zgodnie z formatem wewnętrznym MySQL w celu uniknięcia nieporozumień. MySQL zawsze zakłada, że rok jest pierwszym elementem po lewej stronie daty. Jeśli w operacji SQL podana zostanie nieprawidłowa wartość daty, MySQL wstawi w jej miejsce zero.

MySQL w kontekście liczb całkowitych automatycznie konwertuje daty i czas na liczby całkowite.

DATE

```
DATE
```

Format YYYY-MM-DD (2001-01-01)

Rozmiar w pamięci 3 bajty

Opis

Data kalendarza gregoriańskiego z zakresu od 1 stycznia 1000 roku ('1000-01-01') do 31 grudnia 9999 roku ('9999-12-31').

DATETIME

```
DATETIME
```

Format YYY-MM-DD hh:mm:ss (2001-01-01 01:00:00)

Rozmiar w pamięci 8 bajtów

Opis

Zapisuje czas z zakresu od 00:00:00 1 stycznia 1000 roku ('1000-01-01 00:00:00') do 23:59:59 31 grudnia 9999 ('9999-12-31 23:59:59') według kalendarza gregoriańskiego.

TIME

```
TIME
```

Format hh:mm:ss (01:00:00)

Rozmiar w pamięci 3 bajty

Opis

Zapisuje czas od północy ('00:00:00') do sekundy przed północą ('23:59:59').

TIMESTAMP

```
TIMESTAMP[(rozmiar_wyświetlania)]
```

Format YYYYMMDDhhmmss (20010101060000)

Rozmiar w pamięci 4 bajty

Opis

Zapis chwili z dokładnością do sekundy od północy 1 stycznia 1970 roku do minuty przed północą 31 grudnia 2037 roku. Podstawowym zastosowaniem tego typu jest rejestracja modyfikacji tabel. Przy wstawianiu do takiej kolumny wartości NULL wstawiane są aktualna data i czas. W przypadku modyfikowania jakiejkolwiek wartości w wierszu z kolumną TIMESTAMP pierwsza kolumna tego typu zostanie zaktualizowana bieżącą datą i czasem.

YEAR

```
YEAR[(rozmiar)]
```

Format YYYY (2001)

Rozmiar w pamięci 1 bajt

Opis

Pozwala zapisać rok z kalendarza gregoriańskiego. Parametr *rozmiar* umożliwia zapisywanie roku dwu- lub czterocyfrowo. Zakres YEAR(4) rozciąga się od 1900 do 2155, dla YEAR(2) od 1970 do 2069. Domyślnie przyjmowane jest YEAR(4).

Typy złożone

Złożone typy danych MySQL, ENUM i SET, są po prostu specjalnymi przypadkami typów łańcuchowych. Opisujemy je osobno, gdyż są bardziej złożone pojęciowo i stanowią wprowadzenie

do typów danych SQL3, które być może MySQL będzie obsługiwał w przyszłości.

ENUM

```
ENUM(wartość1, wartość2, ...)
```

Rozmiar w pamięci 1 – 255 elementów: 1 bajt
255 – 65 535 elementów: 2 bajty

Opis

Typ danych `ENUM` pozwala zapisywać jeden z wielu zdefiniowanych wcześniej łańcuchów. Przy tworzeniu kolumny typu `ENUM` podaje się listę dopuszczalnych jej wartości. Dane mogą być do tej kolumny wstawiane i aktualizowane jedynie z tej listy; każda próba wstawienia wartości spoza niej powoduje wstawienie pustego łańcucha.

Do listy dopuszczalnych wartości można się odwoływać przez indeks, przy czym pierwszy element otrzymuje numer 0. Na przykład:

```
SELECT COLID FROM TBL WHERE COLENUM = 0;
```

Jeśli `COLID` jest kolumną z kluczem głównym, a `COLENUM` kolumną typu `ENUM`, taka instrukcja SQL spowoduje pobranie kluczy głównych wszystkich wierszy, dla których `COLENUM` jest pierwszą wartością z listy. Analogicznie, sortowanie względem kolumn `ENUM` powoduje sortowanie według indeksu, nie łańcucha.

Największa możliwa liczba elementów kolumny `ENUM` to 65 535.

SET

```
SET(wartość1, wartość2, ...)
```

Rozmiar w pamięci 1 – 8 elementów: 1 bajt
9 – 16 elementów: 2 bajty
17 – 24 elementy: 3 bajty
25 – 32 elementy: 4 bajty
33 – 64 elementy: 8 bajtów

Opis

Lista wartości wybieranych z określonego wcześniej zbioru. Pole może zawierać dowolną liczbę łańcuchów wskazanych w instrukcji SET, szczególnie może nie zawierać żadnej takiej wartości. SET jest podobny do ENUM, ale każde pole może zawierać więcej niż jedną z podanych wartości. Dane typu SET nie są jednak zapisywane za pomocą indeksów, ale w złożonej mapie bitowej. Jeśli dany jest zbiór SET, zawierający elementy: Mandarynka, Pomidor, Gruszka i Banan, każdy z tych elementów jest reprezentowany jako włączony bit w bajcie, jak to pokazano w tabeli 3.1.

Tabela 3.1. Reprezentacja zbioru elementów w MySQL

Element	Wartość dziesiętna	Zapis bitowy
Mandarynka	1	0001
Pomidor	2	0010
Gruszka	4	0100
Banan	8	1000

W powyższym przykładzie zapisanie jednocześnie wartości Mandarynka i Gruszka wymaga użycia wartości 5 (0101).

W kolumnie SET można zapisać najwyżej 64 wartości. Wprawdzie tę samą wartość można w jednym wyrażeniu SQL wpisać wielokrotnie, ale w bazie danych wartość ta zostanie zapisana raz.