1. Typy danych

a). Typy liczbowe:

- **INT**

INT czyli INTEGER (liczba całkowita) może przechowywać wartości **od -2147483648 do 2147483647**. W przypadku wykorzystania atrybutu **UNSIGNED** (czyli bez znaku) pozwala przechowywać wartości **od 0 do 4294967295**.
Przykłady wykorzystania INTEGER:
Jest to dość uniwersalny typ danych numerycznych – może mieć zastosowanie w obliczeniach. Z atrybutem bez znaku może być wykorzystany w kolumnach ID. Pojedyncza wartość INT zajmuje **4 bajty**.

- **TINYINT**

TINYINT to odchudzona wersja INTEGER. Zajmując **tylko 1 bajt** przestrzeni pozwala przechować wartości od **-128 do 127**. Bez znaku (atrybut UNSIGNED) pozwala przechować wartości **od 0 do 255**.

Przykład wykorzystania TINYINT:

TINYINT nadaje się do przechowywania ustawień, stanów oraz kluczy w tabelach, w których z góry wiadomo, ze będą miały stosunkowo małą ilość rekordów. W tabeli z użytkownikami może to być poziom uprawnień.

- **SMALLINT**

SMALLINT czyli „mała liczba całkowita” pozwala na zapis wartości **od -32768 do 32767**. Bez znaku: **od 0 do 65535**. Zajmuje przestrzeń 2 bajtów.

- **MEDIUMINT**

MEDIUMINT (średniej wielkości liczba całkowita) na 3 bajtach pamięci przechowuje wartości **od -8388608 do 8388607**. Bez znaku potrafi przechować wartość **od 0 do 16777215**.

Zarówno SMALLINT jak i MEDIUMINT są pośrednimi typami pomiędzy TINYINT a INTEGER.

- **BIGINT**

BIGINT (duża liczba całkowita) pozwala na 8 bajtach przechować wartości całkowite **od -2^{63} do $2^{63}-1$**. Bez znaku będą to wartości **od 0 do $2^{64}-1$**.

BIGINT jest w stanie przechowywać bardzo duże wartości. Może być wykorzystany we wszelkiego rodzaju obliczeniach.

- **DECIMAL**

Typ DECIMAL wykorzystujemy do zapisu wartości stałoprzecinkowych czyli o określonej precyzji. Dla typu DECIMAL należy też zdefiniować długość – domyślna to (10,0) co oznacza że wartość przechowa 10 cyfr, w tym 0 cyfr po przecinku.

Przykład wykorzystania:

DECIMAL często znajduje zastosowane w cenach i kwotach walutowych. Przykładowo **dla polskich cen odpowiedni będzie format (6,2)**. Pozwoli przechować kwoty od -9999.99 do maks. 9999.99zł. Jeżeli kwota ta stanowi ograniczenie można zmienić 6 w deklaracji długości na inną wartość.

- **FLOAT**

FLOAT to typ do zapisu liczb zmiennoprzecinkowych **pojedynczej precyzji**. (ang. *floating-point number*). Oznacza to, że może przechowywać wartości o różnej precyzji. **Wartością w polu może być zarówno 2.7 jak i 2.7413089332**. Do rozdzielenia części po przecinku używamy kropki.
FLOAT do zapisania pojedynczej precyzji wykorzystuje 4 bajty.

- **DOUBLE**

DOUBLE to również typ do zapisu liczb zmiennoprzecinkowych ale **z podwójną precyzją**. MySQL do podwójnej precyzji wykorzystuje 8 bajtów.

- **REAL**

REAL to synonim dla typu DOUBLE o ile nie jest wyłączony tryb REAL_AS_FLOAT.

Opcjonalnie, dla FLOAT, DOUBLE i REAL możemy zadeklarować długość. MySQL dopuszcza tutaj niestandardową składnię np. FLOAT(M,D). Oznacza to że w polu możemy zapisać **wartość z M cyframi z czego D cyfr może znajdować się po przecinku**. Deklaracja FLOAT (8,4) pozwoli przechować wartość -1234.567. **MySQL automatycznie wykona odpowiednie zaokrąglenia** np. chcąc zapisać wartość 1111.00009 do pola w kolumnie FLOAT(8,4) zostanie zapisana wartość 1111.0001

- **BIT**

Typ BIT przechowuje M bitową wartość binarną. Domyślnie jest to jeden bit a maksymalnie może ich być 64.

- **BOOLEAN**

To typ wartości boolowskich (prawda lub fałsz). Wartość 0 uznaje się za fałsz (FALSE) a wszelkie inne wartości (niezerowe) uznaje się za prawdę. Synonim dla TINYINT(1). Wartości BOOLEAN mogą mieć szerokie wykorzystanie w aplikacjach np. zapisu stanu ustawień Tak/Nie np. czy checkbox został zaznaczony.

b). Typy daty i czasu:

- **DATE**

DATE pozwala zapisać datę (bez czasu) w zakresie **od dnia 1000-01-01 do dnia 9999-12-31**.
Formatem jest tak jak w przykładzie: YYYY-MM-DD ale separatorem może być inny znak np. YYYY/MM/DD, YYYY^MM^DD lub YYYY@MM@DD.

- **DATETIME**

DATETIME pozwala zapisać datę z czasem. MySQL pobiera i wyświetla takie wartości w formacie YYYY-MM-DD hh:mm:ss. Obsługiwany zakres to od **1000-01-01 00:00:00 do 9999-12-31 23:59:59**.

- **TIMESTAMP**

TIMESTAMP czyli pieczątka czasu obsługuje **zakres od 1970-01-01 00:00:01 UTC do 2038-01-19 03:14:07 UTC**.

W języku MySQL możemy łatwo wybierać daty za pomocą zakresów oraz przeprowadzać na nich standardowe dla takich typów działania.

Czym się różni DATETIME od TIMESTAMP w SQL?

- różnią się zakresami,
- DATETIME wymaga 8 bajtów pamięci a TIMESTAMP 7 bajtów,
- TIMESTAMP można konwertować w zależności od czasu lokalnego względem UTC. DATETIME nie umożliwia takiej funkcji.
- TIMESTAMP może być indeksowany, DATETIME nie umożliwia indeksowania
- wartości TIMESTAMP można ponadto przechowywać w query cache.

- **TIME**

TIME jest typem zarezerwowanym do samego czasu bez daty. Przykładowym zastosowaniem może być długość utworu multimedialnego.

- **YEAR**

YEAR jest oszczędnym typem, który może zawierać tylko rok. Obsługuje zapis czterocyfrowy (np. 2020) i dwucyfrowy. Zapis z minusem oznacza lata dwu tysięczne, np. -20 to rok 2020 a 88 oznacza rok 1988.

c). Typy znakowe:

- **CHAR**

CHAR to ciąg znaków o konkretnej długości. Domyślnie jest to 1 znak ale można też zdefiniować CHAR o długości **maksymalnie 255 znaków** (np.: CHAR(15)). W przypadku wprowadzenia ciągu krótszego od zadeklarowanego jest on w prawo dopełniony spacjami. Nie widać ich przy pobieraniu danych jeżeli nie włączono opcji PAD_CHAR_TO_FULL_LENGTH.

- **VARCHAR**

VARCHAR reprezentuje ciąg znaków o różnej długości (**od 0 do 65,535 znaków**). Nie jest dopełniany spacjami do maksymalnego rozmiaru.

Czym się różni CHAR od VARCHAR w SQL?

Dane są przechowywane w inny sposób: CHAR jest dopełniany spacjami a VARCHAR nie. CHAR zawsze zajmuje stały rozmiar a VARCHAR nie zajmuje przestrzeni, jeżeli wprowadzono krótsze wartości.

Dla zobrazowania można stwierdzić następujące fakty:

Wartość „pusta” w CHAR(4) zajmie 4 bajty będzie zapisana jako cztery spacje `'    '`, natomiast VARCHAR zostanie zapisany jako wartość „pusta” - `''` i zajmie 1 bajt.

Wartość **`'Le'`** zostanie zapisana w CHAR(4) jako **`'Le  '`** i zajmie 4 bajty. W VARCHAR(4) zajmie tylko 3 bajty bo **`'Le'`** wymaga dwóch bajtów na znaki i 1 bajt na deklarację długości.

Wartość **`'Lekc'`** zostanie zapisana w CHAR(4) jako **`'Lekc'`** i zajmie 4 bajty. W VARCHAR(4) zajmie natomiast 5 bajtów.

Wartość **`'Lekcja'`** zostanie zapisana w CHAR(4) jako **`'Lekc'`** (ciąg będzie obcięty) zajmie 4 bajty a w VARCHAR(4) **`'Lekc'`** (ciąg będzie również obcięty) i zajmie 5 bajtów.

Uwaga: wartości będą obcięte tylko kiedy opcja SQL strict jest wyłączona. Jeżeli opcja SQL strict jest włączona, próba zapisu dłuższego ciągu od deklaracji kolumny CHAR albo VARCHAR spowoduje błąd.

- **TEXT**

TEXT też służy do przechowywania danych tekstowych. Pozwala przechować pełne **65,535 znaki**. TEXT ma stały rozmiar i nie można deklarować innych długości.

Różnice pomiędzy TEXT a VARCHAR

Zarówno TINYTEXT, TEXT, MEDIUMTEXT i LONGTEXT pozwalają przechować tekst. Od typów CHAR i VARCHAR różnią się te typy tym, że tak naprawdę w komórkach tabeli zawierają tylko referencję do tekstu umieszczonego poza tabelą. Kolejną różnicą jest to, że tekst w CHAR i VARCHAR może być częścią indeksu.

Pod kątem wydajnościowym warto wyrobić sobie nawyk wybierania VARCHAR dla tekstu, który nie będzie przekraczał 65,535 znaków.

- **TINYTEXT**

TINYTEXT pozwala przechować już tylko **255 znaki**.

- **MEDIUMTEXT**

MEDIUMTEXT jak można się domyślić pozwala przechować większą ilość tekstu w tym wypadku **maksymalny rozmiar danych to 16MB**.

- **LONGTEXT**

LONGTEXT może przechowywać duże zbiory tekstu **aż do 4GB**.

- **BINARY i VARBINARY**

BINARY i VARBINARY są bardzo podobne do CHAR i VARCHAR z tą różnicą że przechowują ciągi binarne zamiast ciągi znaków. W praktyce oznacza to, że porównywanie i sortowanie są oparte o wartości liczbowe bajtów.
Maksymalne długości wartości są takie same jak w przypadku CHAR (255) i VARCHAR (65535) z tą różnicą że mówimy o bajtach a nie o znakach.

- **BLOB**

BLOB można wykorzystać do przechowywania dużych obiektów binarnych. BLOB może przechowywać zmienną ilość danych. Od typu TEXT różni się tym, że nie przechowuje danych w postaci tekst tylko w postaci binarnej.

- **TINYBLOB, MEDIUMBLOB, LONGBLOB**

Różnią się od BLOB maksymalnym rozmiarem danych. Obowiązują analogiczne wartości jak w przypadku TINYTEXT z tą różnicą, że mówimy o bajtach.

W praktyce typ BLOB może być wykorzystywany do przechowywania np. obrazów czy innych plików multimedialnych.