Bazy danych - Ćw.5

mgr inż. Nikodem Bulanda

12 kwietnia 2023

Streszczenie

DDL i DML - część I

Spis treści

1 Typy dnaych w SZDB na przykładzie MySQL

1.1 Zmienne liczbowe

- **Całkowite duże** - w przedziale od -9.223.372.036.854.775.808 do 9.223.372.036.854.775.807. Bez znaku w przedziale od 0 to 18446744073.709.551.615

  BIGINT[(M)] [UNSIGNED] [ZEROFILL]

- **Całkowite małe** - w przedzialeod -128 do 127 lub bezznakowo od 0 do 255:

  TINYINT[(M)] [UNSIGNED] [ZEROFILL]

- **Całkowite normalne** - w przedziale -2.147.483.648 to 2.147.483.647 Bez znaku w przedziale od 0 to 4.294.967.295:

  INT[(M)] [UNSIGNED] [ZEROFILL]

- **Całkowite normalne** - w przedziale -2.147.483.648 to 2.147.483.647 Bez znaku w przedziale od 0 to 4.294.967.295:

  INTEGER[(M)] [UNSIGNED] [ZEROFILL]

- **Całkowita z precyzją mniejszą niż INT** w przedziale od -32.768 do 32.767 lub bezznakowo od 0 do 65.535:

  SMALLINT[(M)] [UNSIGNED] [ZEROFILL]

- **Zmiennoprzecinkowe** - w zakresie od -3.402823466E+38 do -1.175494351E-38, 0, oraz 1.175494351E-38 do 3.402823466E+38

  FLOAT[(M,D)] [UNSIGNED] [ZE- ROFILL]

- **Zmiennoprzecinkowe duże** - w zakresie od -1.7976931348623157E+308 do - 2.2250738585072014E-308 oraz 2.2250738585072014E-308 do 1.7976931348623157E+308

  DOUBLE[(M,D)] [UNSIGNED] [ZEROFILL]

- **Zmiennoprzecinkowa duża** - zapisana jako łańcuch znaków, w zakresie od - 1.7976931348623157E+308 do -2.2250738585072014E-308, 0, oraz 2.2250738585072014E- 308 do 1.7976931348623157E+308

  DECIMAL[(M[,D])] [UNSIGNED] [ZEROFILL]

1.2 Zmienne daty i czasu

- **Czas** - *godzina, minuta, sekunda i separatory, razem 8 pozycji*: TIME
- **Data**: DATE
- **Data i czas**: DATETIME
- **Data i czas** - *19 pozycyjna*: TIMESTAMP
- **Rok - ma 2 lub 4 pola**: YEAR [(2|4)]

1.3 Zmienne łańcuchowe / teskstowe

- **Tekst do 255 znaków**: TINYBLOB
- **Tekst do 255 znaków**: TINYTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
- **Tekst do 255 znaków, uzupełniany zerami**: [NATIONAL] CHAR(M) [CHARACTER SET charset_name] [COLLATE collation_name]
- **Tekst do 255 znaków, nie uzupełniany zerami**: [NATIONAL] VARCHAR(M) [CHARACTER SET charset_name] [COLLATE collation_name]
- **Tekst do 65.538 znaków**: BLOB[(M)]
- **Tekst do 65.538 znaków**: TEXT[(M)] [CHARACTER SET charset_name] [COL- LATE collation_name]
- **Tekst do 16.777.215 znaków**: MEDIUMBLOB
- **Tekst do 16.777.215 znaków**: MEDIUMTEXT [CHARACTER SET charset_name] [COLLATE collation_name]
- **Tekst do do 4.294.967.295 znaków**: LONGBLOB
- **Tekst do 4.294.967.295 znaków**: LONGTEXT [CHARACTER SET charset_name] [COLLATE collation_name]

1.4 Różne typy danych

- **Bit na znak**: BINARY(M) M bytes, 0 <= M <= 255
- **Boolowskie** BOOLEAN - Logiczne - 0 jest fałszem, a każda liczba w zakresiod -128 do 127 lub od 1 do 255 jest prawdą: mysql> SELECT IF(0, 'true', 'false')
- **ENUM('value1','value2',...)** - Typ danych wyliczeniowy, przyjmuje jako wartość jeden element z zadeklarowanej listy.
- **SET('value1','value2',...)** - Typ danych wyliczeniowy, może przyjąć jako wartość listę (wiele) pozycje z zadeklarowanej, oddzielonych przecinkiem.
- **JSON** - Typ danych umożliwiający przechowywanie i walidowanie obiektów typu JSON

Zapoznaj się z ich dokumentacją oraz pozostałymi typami `https://dev.mysql.com/doc/refman/8.0/en/data-types.html`

1.4.1 Atrybuty

Mianem atrybutu możemy określić cechę kolumny pozwalającą na wyróżnienie jej z pośród pozostałych kolumn. Poniżej przedstawiono najczęściej stosowane atrybuty.

- **DEFAULT** value - wartość domyślna pola

Listing 1: Deklaracja na poziomie tworzenia tabeli:

```
1     CREATE TABLE Pacjent (
2         id int NOT NULL,
3         imie varchar (255) NOT NULL,
4         nazwisko varchar (255),
5         wiek int,
6         miasto (255) DEFAULT 'Nowy Sacz'
7     );
```

- **NULL, NOT NULL** - Dopuszcza lub nie dopuszcza możliwość wystąpienia wartość null dla pól.

Listing 2: Deklaracja na poziomie tworzenia tabeli:

```
1     CREATE TABLE Pacjent (
2         id int NOT NULL ,
3         imie varchar (255) NULL ,
4         nazwisko varchar (255) ,
5         wiek int ,
6         miasto (255) DEFAULT 'Nowy Sacz'
7     );
```

- **PRIMARY KEY** - tworzy unikalny klucz tabeli

Listing 3: Deklaracja na poziomie tworzenia tabeli:

```
1     CREATE TABLE Pacjent (
2         id int NOT NULL ,
3         imie varchar (255) NOT NULL ,
4         nazwisko varchar (255) ,
5         wiek int ,
6         miasto (255) DEFAULT 'Nowy␣Sacz' ,
7         PRIMARY KEY ( id , nzawisko )
8     );
9
10    CREATE TABLE Pacjent (
11        id int NOT NULL PRIMARY KEY ,
12        imie varchar (255) NOT NULL ,
13        nazwisko varchar (255) ,
14        wiek int ,
15        miasto (255) DEFAULT 'Nowy␣Sacz'
16    );
17
18    CREATE TABLE Pacjent (
19        id int NOT NULL ,
20        imie varchar (255) NOT NULL ,
21        nazwisko varchar (255) ,
22        wiek int ,
23        miasto (255) DEFAULT 'Nowy␣Sacz' ,
24        CONSTRAINT PK_Pacjent
25        PRIMARY KEY ( ID , nazwisko )
26    );
```

- **UNSIGNED** - Wszystkie typy liczb całkowitych mogą mieć opcjonalny (niestandardowy) atrybut UNSIGNED. Typu bez znakowy można wykorzystać, w celu ograniczenia wartości kolumny tylko do liczby nieujemnych lub gdy konieczne jest zwiększenie górnego zakresu liczbowego. Na przykład, jeśli kolumna INT przyjmuje atrybut UNSIGNED, rozmiar zakresu kolumny jest taki sam, ale jej punkty końcowe zmieniają się z -2147483648 i 2147483647 do 0 i 4294967295.

- **UNIQUE** - Definiuje pole jako unikalne, atrybut konieczny do zrealizowania relacji 1:1.

Listing 4: Deklaracja na poziomie tworzenia tabeli:

```
1     CREATE TABLE Pacjent (
2         id int NOT NULL PRIMARY KEY ,
3         imie varchar (255) NOT NULL ,
4         nazwisko varchar (255) ,
5         wiek int ,
6         miasto (255) DEFAULT 'Nowy␣Sacz' ,
7         UNIQUE ( ID )
8     );
9
10    CREATE TABLE Pacjent (
11        id int NOT NULL PRIMARY KEY ,
12        imie varchar (255) NOT NULL ,
13        nazwisko varchar (255) ,
14        wiek int ,
15        miasto (255) DEFAULT 'Nowy␣Sacz' ,
16        CONSTRAINT UC_Pacjent UNIQUE ( id , nazwisko )
17    );
```

Listing 5: Deklaracja postkreacyjna oraz modyfikacja jak i usuniecie:

```
1     ALTER TABLE Pacjent ALTER ADD UNIQUE ( id );
2     ALTER TABLE Pacjent ADD CONSTRAINT UC_Pacjent
3         UNIQUE ( id , nazwisko );
4
5     ALTER TABLE Pacjent ALTER DROP UNIQUE ( id );
6     ALTER TABLE Persons DROP CONSTRAINT UC_Pacjent ;
```

- **ZEROFILL** - dopełnia vartości pola zerami do pełnego jego rozmiar
- **BINARY** - określa, iż kolumna tekstowa CHAR lub VARCHAR przechowuje wartości binarne i uwzględnia wielkość znaków. Przy porównywaniu pól tekstowych wielkość liter jest ignorowana
- **CHECK** - Począwszy od MySQL 8.0.16, CREATE TABLE umożliwia deklaracje podstawowej funkcji ograniczeń CHECK w obrębie tabeli i kolumn dla wszystkich silników pamięci masowej. CREATE TABLE umożliwia następującą składnię ograniczeń CHECK, zarówno dla ograniczeń tabeli, jak i ograniczenia kolumny. Wyrażenie (expr) określa warunek ograniczenia (traktowanego jako wyrażenie logiczne), którego wartością musi być TRUE lub UNKNOWN (dla wartości NULL) dla każdego wiersza tabeli. Jeśli warunek przyjmie wartość FALSE, jego przejście nie powiedzie się i nastąpi naruszenie ograniczenia.

Listing 6: Deklaracja postkreacyjna oraz modyfikacja jak i usuniecie:

```
1      [CONSTRAINT [symbol]] CHECK (expr) [[NOT] ENFORCED]
2
3      CREATE TABLE t1(
4          CHECK (c1 <> c2),
5          c1 INT CHECK (c1 > 10),
6          c2 INT CONSTRAINT c2_positive CHECK (c2 > 0),
7          c3 INT CHECK (c3 < 100),
8          CONSTRAINT c1_nonzero CHECK (c1 <> 0),
9          CHECK (c1 > c3)
10     );
```

Zapoznaj się z dokumentacja
https://dev.mysql.com/doc/refman/8.0/en/create-table-check-constraints.html

2 Budowanie relacji i ich modyfikacja

2.1 Budowanie relacji

- **Primary key** - Klucz główny, odnosi się do kolumny lub zestaw kolumn, który posłuży do jednoznacznego identyfikowania wiersza w tabeli. Tworzenie klucza - metoda.1 polega na zadeklarowaniu go na końcu tabeli oraz określeniu atrybutach wchodzących w jego skład. metoda.2 polega na zadeklarowaniu go w tej samej lini w której deklarowaliśmy kolumnę.

- **Foreign key** - Reprezentuje związki między tabelami. Wymagania: kolumny na których zakładany jest klucz obcy muszą być tego samego typu (jak klucz główny w dowiązanej tabeli), znak i rozmiar dla typu INTEGER muszą być identyczne, nazwy zaś mogą się różnić.

Listing 7: Relacja jeden do wielu między tabelą Samochody a Osoby

```
1     CREATE TABLE pojazdy (
2         id_pojazdu INTEGER PRIMARY KEY NOT NULL AUTO_INCREMENT ,
3         marka VARCHAR (40) ,
4         model VARCHAR (40) ,
5         rok_produkcji YEAR
6     );
```

Listing 8: Relacja jeden do wielu między tabelą Samochody a Osoby cd

```
1     CREATE TABLE osoby (
2         id_osoby INTEGER PRIMARY KEY NOT NULL AUTO_INCREMENT ,
3         nazwisko VARCHAR (35) ,
4         imie VARCHAR (15) ,
5         pesel CHAR (11) ,
6         data_urodzenia DATE ,
7         pojazd_id INTEGER ,
8         FOREIGN KEY fk_samochodu ( pojazd_id )
9         REFERENCES pojazdy ( id_pojazdu )
10    );
```

Listing 9: Budowa tabeli na podstawie inne tabeli z przeniesieniem danych i atrybutów

```
1     CREATE TABLE nowa_tabela AS
2     SELECT * FROM stara_tabela;
```

Listing 10: Budowa tabeli na podstawie inne tabeli bez przeniesieniem danych

```
1     CREATE TABLE nowa_tabela LIKE stara_tabela_wzorcowa;
```

2.2 Utworzenie klucza obcego na istniejącej tabeli

Listing 11: Utworzenie klucza obcego na istniejącej tabeli

```
1     ALTER TABLE Oooby ADD FOREIGN KEY fk_pojazdu ( pojazd_id )
2     REFERENCES pojazdy ( id_pojazdu ) on delete restrict ;
```

2.3 Edycja struktury tabeli-relacji

Ogólna formuła instrukcji ALTER TABLE jest następująca

Listing 12: Definicja zmian

```
1  ADD [ COLUMN ] definicja_tworzenia [ FIRST | AFTER nazwa_kol ]
2  ADD [ COLUMN ] ( definicja_tworzenia, definicja_tworzenia, ...)
3  ADD INDEX [nazwa_indeksu]( nazwa_indexu, nazwa_indexu_kol, ...)
4  ADD PRIMARY KEY ( nazwa_indexu_kol, ...)
5  ADD UNIQUE ( nazwa_indexu, nazwa_indexu_kol, ...)
6  ADD FULLTEXT ( nazwa_indexu, nazwa_indexu_kol, ...)
7  CREATE VIEW v_z AS SELECT * FROM zwierzeta ;
8  ADD [ CONSTRAINT symbol ]
9  FOREIGN KEY [ nazwa_indexu
10 ( nazwa_indexu , nazwa_indeksu_kol , ...)
11 [ definicja_odwolania ]
12
13 ALTER [ COLUMN ] nazwa_kol { SET DEFAUL litera | DROP DEFAULT }
14 CHANGE [ COLUMN ] dawna_nazwa_kol definicja_tworzenia
15         [ FIRST | AFTER nazwa_kol ]
16 MODYFI
17 DROP [ COLUMN ] nazwa_kol
18 DROP PRIMARY KEY
19 DROP INDEX nazwa_indexu
20 DISABLE KEYS
21 RENAME [ TO ] nowa_nazwa_tab
22 ORDER BY nazwa_kol
```

3 Wprowadzanie i edycja danych

3.1 Wprowadzanie danych

Istnieje kilka sposobów na realizacje wprowadzania danych do bazy typu sql.

Listing 13: Wprowadzenie danych

```
1 INSERT [ INTO ] nazwa_tabeli [( nazwa_kolumny ,...)]
2 VALUES ({ wyrazenie_lub_wartosc | DEFAULT } ,...) ,
3 (...) ,...[ ON DUPLICATE KEY UPDATE nazwa_kolumny = wyrazenie , ... ]
4
5 INSERT [ INTO ] nazwa_tabeli
6     SET nazwa_kolumny ={ wyrazenie | DEFAULT } , ...
```

Listing 14: Wprowadzenie danych na podstawie wyboru danych z innej tabeli - kopiowanie

```
1 INSERT INTO tabela2 (kolumna1, kolumna2, kolumna3, ...)
2     SELECT kolumna1, kolumna2, kolumna3, ... FROM tabela1
3 WHERE warunek;
```

Listing 15: Wprowadzenie danych z pliku csv

```
1 LOAD DATA INFILE 'c:/tmp/test.csv'
2     INTO TABLE tabela2
3     FIELDS TERMINATED BY ','
4     ENCLOSED BY '"'
5     LINES TERMINATED BY '\n'
6     IGNORE 1 ROWS;
```

Listing 16: Przykład bez i z autonumeracją

```
1  INSERT INTO pojazdy (id_pojazdu, marka, model, rok_produkcji)
2      VALUES  ( '1', 'Opel', 'Astra', '2010') ,
3              ( '2', 'Opel', 'Vectra', '2009' );
4
5  INSERT INTO pojazdy( id, marka, model, rok_produkcji)
6      VALUES  ( DEFAULT, 'Opel' , 'Astra' , '2010'),
7              ( DEFAULT, 'Opel' , 'Vectra' , '2009');
8
9  INSERT INTO pojazdy
10     VALUES  ( DEFAULT, 'Opel', 'Astra', '2010'),
11             ( DEFAULT, 'Opel' , 'Vectra' , '2009' );
```

3.2 Edycja danych

Listing 17: Definicja

```
1  UPDATE nazwa_tabeli SET
2      nazwa_kolumny1 = wyrazenie1
3      [, nazwa_kolumny2 = wyrazenie2 ...]
4  [WHERE warunek_dzialania] [ORDER BY ...] [LIMIT warunek_limitu]
```

Listing 18: Praca na wielu tabelach

```
1  UPDATE tabele_referencyjne
2      SET
3      nazwa_kolumny1 = wyrazenie1 [ ,
4      nazwa_kolumny2 = wyrazenie2 ...]
5  [WHERE warunek_dzialania]
```

4 Zadania

Zadanie 1 Zbuduj relację user składającą się z następujących atrybutów: id_user, login, password, active, first_login, id_employees. Dopasuj do nich typy danych i ograniczenia wiedząc, że: kolumna active przechowuje flagę TRUE lub FALSE, frist_login powinien przechowywać pełną datę, godzinę, minutę sekundę i mikrosekundę po aktywacji, id_employees będzie kluczem obcym pochodzącym z tabeli employees i kolumny emp_no. Pamiętaj, aby id_user było automatycznie podnoszone jako klucz podstawowy. Wypełnij co najmniej cztery rekordy.

Zadanie 2 Do zbudowanej wcześniej tabeli dodaj kolumnę last_login i przedstaw polecenie aktualizujące dane tylko w tej kolumnie, wpisujące w miejsce wartości aktualny czas. Typ danych użyty powinien być zbieżny z typem kolumny first_login.

Zadanie 3 Zbuduj relację roles składającą sie z atrybutów id_role, name, superior będzie miał identyczny typ jak id_role. Pamiętaj, aby id_role było automatycznie podnoszone jako klucz podstawowy. Dodaj 4 rekordy, w których klucz podstawowy będzie nadawane domyślnie zaś nazwa roli to : superadmin, admin, manager, user, pole superior pozostaw puste.

Zadanie 4 Do relacji user dodaj pole role, które będzie kluczem obcym z tabeli roles(id_role). Następnie z edytuj rolę wszystkim użytkownikom, ustawiając ją na user, jedynym wyjątkiem będzie pierwszy użytkownik, jemu nadaj rolę superadmin

Zadanie 5 Stwórz relację o nazwie user_log, składać się będzie ona z kolumn: id, log_time, user_name , action (zmienna enumeratywna). Dobierz typy danych i ich atrybuty. Dokonaj wpisania co najmniej 5 rekordów, po czym usuń rekordy parzyste.

Zadanie 6 Na podstawie tabeli employees zbuduj tabelę employees_copy jednak pomiń kolumne birth_date.

Zadanie 7 Do tabeli employees_copy dodaj pole birth_date_new i dopasuj wartości z oryginalnej tabeli (employees). Wynik pracy przedstaw za pomocą złączenia ustawiając kolumny birth_date i birth_date_new obok siebie.

Zadanie 8 Z tabeli user usuń kolumnę i klucz obcy role. Na jego miejsce osadź pole role zbudowane z wykorzystaniem pola enumeratywnego, pamiętaj, że użytkownik może posiadać więcej niż jedną rolę. Opisz przebieg zadania.

.