



Sprawozdanie z laboratorium

Bazy Danych

Lab 10: Podstawy języka SQL, część 5, polecenia DDL (CREATE, ALTER, DROP)

19.05.2020

Prowadzący: dr inż. Jacek Tkacz

Grupa: 24-INF/A

Gabryjolek Bartosz
98854@stud.uz.zgora.pl
Grześków, Grzegorz
98859@stud.uz.zgora.pl

1. Wstęp

Nauka pisania poleceń SQL bez używania narzędzi wspomagających projektowanie

2. Zadania

1. Utworzyć strukturę relacyjną pokazaną na rysunku.

Zaczęliśmy od utworzenia nowej bazy danych Po czym stworzyliśmy tabele zgodnie z poleceniem
A następnie określiliśmy dla nich ograniczenia.

```
DROP DATABASE UZ10;  
CREATE DATABASE UZ10;  
USE UZ10;  
CREATE TABLE Pracownicy(prac_id int AUTO_INCREMENT PRIMARY KEY, imie VARCHAR(20), nazwisko VARCHAR(30), data_ur DATE, pesel DECIMAL(11,0) UNIQUE,zarobki DECIMAL(10,2), plec ENUM('M','K'))  
ENGINE = InnoDB;  
CREATE TABLE Projekty(proj_id int AUTO_INCREMENT PRIMARY KEY, nazwa VARCHAR(200) UNIQUE, kod VARCHAR(20) UNIQUE, data_rozp DATE, data_zak DATE)  
ENGINE = InnoDB;  
DROP TABLE IF EXISTS Zespoly;  
CREATE TABLE Zespoly(data_rozp DATE, data_zak DATE)  
ENGINE = InnoDB;  
DROP TABLE IF EXISTS Funkcje;  
CREATE TABLE Funkcje(funkcja_id int AUTO_INCREMENT PRIMARY KEY, nazwa VARCHAR(100) UNIQUE)  
ENGINE = InnoDB;
```

```
ALTER TABLE Pracownicy MODIFY imie VARCHAR(20) NOT NULL;  
ALTER TABLE Pracownicy MODIFY nazwisko VARCHAR(30) NOT NULL;  
ALTER TABLE Pracownicy MODIFY data_ur DATE NOT NULL;  
ALTER TABLE Pracownicy MODIFY pesel DECIMAL(11,0) NOT NULL UNIQUE;  
ALTER TABLE Pracownicy MODIFY zarobki DECIMAL(10,2) NOT NULL;  
  
ALTER TABLE Projekty MODIFY nazwa VARCHAR(200) UNIQUE NOT NULL;  
ALTER TABLE Projekty MODIFY kod VARCHAR(20) UNIQUE NOT NULL;  
ALTER TABLE Projekty MODIFY data_rozp DATE NOT NULL;  
  
ALTER TABLE Pracownicy ADD szef_id int;  
ALTER TABLE Pracownicy MODIFY szef_id int NOT NULL;  
  
ALTER TABLE Pracownicy  
ADD CONSTRAINT prac_prac_FK  
FOREIGN KEY(szef_id)  
REFERENCES Pracownicy(prac_id);
```

```

ALTER TABLE projekty ADD kierownik_id int;

ALTER TABLE Projekty MODIFY kierownik_id int NOT NULL;
ALTER TABLE Projekty
ADD CONSTRAINT prac_proj_FK
FOREIGN KEY(kierownik_id)
REFERENCES Pracownicy(prac_id);

ALTER TABLE Zespoly ADD funkcja_id int;
ALTER TABLE Zespoly MODIFY funkcja_id int NOT NULL;

ALTER TABLE Zespoly
ADD CONSTRAINT fun_zesp_FK
FOREIGN KEY(funkcja_id)
REFERENCES Funkcje(funkcja_id);

ALTER TABLE Zespoly ADD proj_id int;
ALTER TABLE Zespoly MODIFY proj_id int NOT NULL;
ALTER TABLE Zespoly MODIFY prac_id INT;
ALTER TABLE Zespoly MODIFY prac_id int NOT NULL;
ALTER TABLE Zespoly ADD PRIMARY KEY(proj_id, prac_id);

ALTER TABLE Zespoly
ADD CONSTRAINT proj_zesp_FK
FOREIGN KEY(prac_id)
REFERENCES Pracownicy(prac_id);

```

Ograniczenia klucza obcego powinny mieć zdefiniowane nazwy (czy potrafisz wyjaśnić dlaczego?) Unikalne nazwy są im nadawane w celach identyfikacyjnych, znacznie ułatwia to analizę bazy. W przypadku nie nadania żadnej nazwy MySQL robi to automatycznie.

2. Zwróć uwagę, że w jednym przypadku (którym?) klucz główny tabeli jest kluczem złożonym, opartym o dwie kolumny. Jednocześnie kolumny tworzące klucz główny są też kolumnami tworzącymi klucze obce. Przedyskutuj wady i zalety takiego podejścia.

Klucz główny jest kluczem złożonym w tabeli zespoły, wykorzystanie klucza złożonego w relacji identyfikującej umożliwia tworzenie zespołu, tylko wtedy gdy mamy pracownika i projekt, co wyklucza logicznie błędne rekordy. Minusem jest to że wykluczamy również rekord w którym posiadamy pracownika, ale nie istnieje żaden projekt. Wtedy taki pracownik pozostaje bez zespołu. W bieżącej konfiguracji tabeli, zespoły mogą być jedynie jednoosobowe.

3. Opisać własnymi słowami utworzony model. Jakie jest jego potencjalne zastosowanie. Do opisu jakiego rzeczywistego problemu został on stworzony?

Ten model bazy może być wykorzystywany np. w firmie zajmującej się architekturą wnętrz (jeden architekt, jedno mieszkanie, jednoosobowe niezależne zespoły).

4. Wstawić do utworzonych tabel przykładowe rekordy: do tabeli pracownicy 20 rekordów, do tabeli projekty 3 rekordy, do tabeli funkcje 5 rekordów, do tabeli zespoły 20 rekordów. Wartości dla kolumn będących PRIMARY KEY pobierać wykorzystując zdefiniowaną wcześniej opcję AUTO INCREMENT.

```

INSERT INTO Pracownicy (imie, nazwisko, data_ur, pesel, zarobki, plec, szef_id)
VALUES ("Andrzej", "Golota", "1968-01-05", "12345678911", "2100", "M", 1);
INSERT INTO Pracownicy (imie, nazwisko, data_ur, pesel, zarobki, plec, szef_id)
VALUES ("Andrzej", "Golota", "1968-01-05", "12345678912", "2100", "M", 1);
INSERT INTO Pracownicy (imie, nazwisko, data_ur, pesel, zarobki, plec, szef_id)
VALUES ("Andrzej", "Golota", "1968-01-05", "12345678913", "2100", "M", 1);
INSERT INTO Pracownicy (imie, nazwisko, data_ur, pesel, zarobki, plec, szef_id)
VALUES ("Andrzej", "Golota", "1968-01-05", "12345678914", "2100", "M", 1);
INSERT INTO Pracownicy (imie, nazwisko, data_ur, pesel, zarobki, plec, szef_id)
VALUES ("Andrzej", "Golota", "1968-01-05", "12345678915", "2100", "M", 1);
INSERT INTO Pracownicy (imie, nazwisko, data_ur, pesel, zarobki, plec, szef_id)
VALUES ("Andrzej", "Golota", "1968-01-05", "12345678916", "2100", "M", 1);

```

```

INSERT INTO Projekty(nazwa,kod,kierownik_id,data_rozp,data_zak)
VALUES("dieta", "1","1","1988-01-05","1998-01-05");
INSERT INTO Projekty(nazwa,kod,kierownik_id,data_rozp,data_zak)
VALUES("trening", "2","2","1988-01-05","1998-01-05");
INSERT INTO Projekty(nazwa,kod,kierownik_id,data_rozp,data_zak)
VALUES("walka", "3","3","1988-01-05","2000-01-05");

```

```

INSERT INTO Funkcje(nazwa) VALUES ("nie");
INSERT INTO Funkcje(nazwa) VALUES ("jestem");
INSERT INTO Funkcje(nazwa) VALUES ("za");
INSERT INTO Funkcje(nazwa) VALUES ("bardzo");
INSERT INTO Funkcje(nazwa) VALUES ("kreatywny");

```

```

INSERT INTO Zespoly(prac_id, proj_id, funkcja_id, data_rozp, data_zak)
VALUES (20,1,5,"2000-01-01","1968-01-05");

```

5. Zmodyfikować definicję tabeli projekty (polecenie ALTER). Dodać kolumnę poziom trudności, tak jak to pokazano na poniższym rysunku.

Projekty		
proj_id	Integer	NN (PK)
nazwa	Varchar(200)	NN
kod	Varchar(20)	NN
poziom_trudnosci	Integer	NN
kierownik_id	Integer	NN (FK)
data_rozp	Date	NN
data_zak	Date	

```

MariaDB [UZ10]> ALTER TABLE Projekty
-> ADD poziom_trudnosci
-> int
-> NOT NULL
-> Check(poziom_trudnosci <=3);
Query OK, 0 rows affected (0.020 sec)

```

6. Na bazie tabel customer, ord,item oraz product ze schematu demonstracyjnego zbudować widok (polecenie CREATE VIEW) o nazwie zamówienia view.

```
CREATE VIEW zamówienia_view AS
SELECT c.name AS "klient",
p.name AS "produkt",
i.price AS "cena jedn."
i.quantity AS "ilosc"
i.quantity * i.price AS "razem"
FROM customer c, ord o, item i, product p
WHERE c.id = o.customer_id AND o.id = i.ord_id and p.id = i.product_id
ORDER BY Klient;
```

7. Zaproponować oraz zbudować 2 inne sensowne widoki na bazie schematu demonstracyjnego.

Widok pokazujący pracownika, zarobki i region w którym pracuje.

```
CREATE VIEW emp_view2 AS
SELECT e.first_name AS "First Name",
e.last_name AS "Last Name",
e.salary AS "Slavery Fee",
d.region_id AS "Region number"
FROM emp e, dept d;

CREATE VIEW
```

Widok pokazujący magazyny z nazwami ich regionów i ich kierowników.

```
CREATE VIEW war_view1 AS
SELECT w.id AS "warehouse id",
r.name AS "region",
e.last_name AS "Manager"
FROM warehouse w, region r, emp e
WHERE e.id=w.manager_id;
```

3. Wnioski

Polecenie ALTER TABLE umożliwia szybką edycję tabel, co w znaczący sposób przyspiesza usuwanie błędów i edycję bazy. Widoki z kolei są błyskawiczną metodą na zestawienie danych z różnych tabel bazy. Przy wykorzystywaniu polecenia DROP należy być szczególnie ostrożnym, gdyż MySQL nie tworzy kopii zapasowych i raz usunięta tabela lub baza danych może zostać bezpowrotnie utracona.