

Bazy Danych i Systemy Informacyjne

Laboratorium: sprawozdanie z listy 3

1. Wprowadzenie

1.1. Cel zadań

Polecenia z listy 3. sprawdzają i pozwalają na zapoznanie się z podstawami:

- wykorzystywania kursorów do generowania rekordów,
- tworzenia zaawansowanych warunków dla wstawianych rekordów do tabel,
- zastosowania transakcji i prepared statements,
- robienia kopii zapasowych oraz odtwarzania danych z kopii
- działania SQL Injection i zabezpieczania baz danych przed nim.

1.2. Środowisko SQL wykorzystane w zadaniach

W rozwiązywaniu zadań wykorzystano darmowe zintegrowane środowisko programistyczne MySQL, w którego skład wchodzi:

- MySQL Workbench 8.0 – wizualne, programistyczne narzędzie do projektowania baz danych zawierające pole do wprowadzania zapytań w składni MySQL (wszystkie zadanie zostały rozwiązane z pomocą kwerend pisanych w języku SQL)
- MySQL Server – oprogramowanie służące za serwer bazodanowy, umożliwia połączenie programu MySQL Workbench z bazami danych

2. Opis rozwiązań poleceń

Uwaga – w rezultatach często są pokazane tylko fragmenty wypisanych rekordów ze względu na ich dużą liczbę.

2.1. Zadanie 1

2.1.1. Polecenie

Utwórz nową bazę danych o dowolnej nazwie a w niej tabele:

- Ludzie (PESEL: char(11), imie: varchar(30), nazwisko: varchar(30), data_urodzenia: date, plec: enum('K', 'M'))
- Zawody (zawod_id: int, nazwa: varchar(50), pensja_min: float, pensja_max: float)
- Pracownicy (PESEL: char(11), zawod_id: int, pensja: float).

- Czy dobrym pomysłem jest stosowanie nr PESEL jako klucza? Jeżeli uważasz, że nie, popraw to.

Zadbaj o prawidłowy format kolumny PESEL. Dopilnuj, by nie można było wprowadzić także ujemnych wartości liczbowych do bazy oraz aby pensja_min < pensja_max.

- Do tabeli Ludzie wprowadź informacje na temat 5 osób niepełnoletnich oraz 45 osób pełnoletnich, ale mających zarazem mniej niż 60 lat oraz 5 osób w wieku co najmniej 60 lat. Tabelę zawody uzupełnij zawodami - polityk, nauczyciel, lekarz, informatyk wraz z odpowiednimi widełkami pensji. Następnie, z wykorzystaniem kursora na tabeli Ludzie, przypisz każdej pełnoletniej osobie zawód (wraz z odpowiednią pensją) i uzupełnij tabelę Pracownicy (Uwaga: zadbaj o to, aby żaden lekarz płci męskiej nie był starszy niż 65 lat a żaden lekarz płci żeńskiej nie był starszy niż 60 lat).

2.1.2. Kwerendy niezbędne do realizacji

```
1 • CREATE DATABASE IF NOT EXISTS Praca;
```

Tabela ludzie:

```
1 • CREATE TABLE IF NOT EXISTS Ludzie(  
2   id INT NOT NULL AUTO_INCREMENT,  
3   PESEL CHAR(11),  
4   imie VARCHAR(30) NOT NULL,  
5   nazwisko VARCHAR(30) NOT NULL,  
6   data_urodzenia DATE NOT NULL,  
7   plec ENUM('K', 'M') NOT NULL,  
8   PRIMARY KEY (id),  
9   CONSTRAINT pesel_poprawna_data  
10  CHECK (  
11    SUBSTRING(PESEL, 1, 2) = SUBSTRING(YEAR(data_urodzenia), 3, 2) AND  
12    (  
13      ( SUBSTRING(YEAR(data_urodzenia), 1, 2) = '18' AND SUBSTRING(PESEL, 3, 2) = MONTH(data_urodzenia) + 80) OR  
14      ( SUBSTRING(YEAR(data_urodzenia), 1, 2) = '19' AND SUBSTRING(PESEL, 3, 2) = MONTH(data_urodzenia)) OR  
15      ( SUBSTRING(YEAR(data_urodzenia), 1, 2) = '20' AND SUBSTRING(PESEL, 3, 2) = MONTH(data_urodzenia) + 20)  
16    ) AND  
17    SUBSTRING(PESEL, 5, 2) = DAY(data_urodzenia)),  
18  CONSTRAINT pesel_poprawna_plec  
19  CHECK (  
20    ( SUBSTRING(PESEL, 10, 1) IN ('0', '2', '4', '6', '8') AND plec = 'K' ) OR  
21    ( SUBSTRING(PESEL, 10, 1) IN ('1', '3', '5', '7', '9') AND plec = 'M' )  
22  ),
```

```

23 CONSTRAINT pesel_suma_kontrolna
24 CHECK (
25     CAST(SUBSTRING(PESSEL, 11, 1) AS UNSIGNED) = 10 -
26     MOD( (CAST(SUBSTRING(PESSEL, 1, 1) AS UNSIGNED) +
27         MOD(CAST(SUBSTRING(PESSEL, 2, 1) AS UNSIGNED) * 3, 10) +
28         MOD(CAST(SUBSTRING(PESSEL, 3, 1) AS UNSIGNED) * 7, 10) +
29         MOD(CAST(SUBSTRING(PESSEL, 4, 1) AS UNSIGNED) * 9, 10) +
30         CAST(SUBSTRING(PESSEL, 5, 1) AS UNSIGNED) +
31         MOD(CAST(SUBSTRING(PESSEL, 6, 1) AS UNSIGNED) * 3, 10) +
32         MOD(CAST(SUBSTRING(PESSEL, 7, 1) AS UNSIGNED) * 7, 10) +
33         MOD(CAST(SUBSTRING(PESSEL, 8, 1) AS UNSIGNED) * 9, 10) +
34         CAST(SUBSTRING(PESSEL, 9, 1) AS UNSIGNED) +
35         MOD(CAST(SUBSTRING(PESSEL, 10, 1) AS UNSIGNED) * 3, 10)), 10)
36 );
37

```

Tabela Zawody:

```

1 • CREATE TABLE IF NOT EXISTS Zawody (
2     zawod_id INT NOT NULL AUTO_INCREMENT,
3     nazwa VARCHAR(50),
4     pensja_min FLOAT,
5     pensja_max FLOAT,
6     PRIMARY KEY (zawod_id),
7     CONSTRAINT max_wieksze_od_min CHECK (pensja_max > pensja_min),
8     CONSTRAINT nieujemne CHECK (pensja_max >= 0 AND pensja_min >= 0));

```

Tabela Pracownicy:

```

1 • CREATE TABLE IF NOT EXISTS Pracownicy(
2     ludzie_id INT NOT NULL,
3     zawod_id INT NOT NULL,
4     pensja FLOAT,
5     FOREIGN KEY (ludzie_id)
6         REFERENCES Ludzie(id),
7     FOREIGN KEY (zawod_id)
8         REFERENCES Zawody(zawod_id),
9     CONSTRAINT nieujemna_pensja CHECK( pensja >= 0 ));
10

```

Wstawianie 5 osób niepełnoletnich:

```

1 DELIMITER $$
2 • CREATE PROCEDURE wstawNiepelnoletnich()
3 BEGIN
4     INSERT INTO ludzie(PESSEL, imie, nazwisko, data_urodzenia, plec) VALUES
5     ('06290499162', 'Anna', 'Nowak', STR_TO_DATE('04,09,2006', '%d,%m,%Y'), 'K'),
6     ('17240578972', 'Jan', 'Kowalski', STR_TO_DATE('05,04,2017', '%d,%m,%Y'), 'M'),
7     ('07221172426', 'Amelia', 'Wolska', STR_TO_DATE('11,02,2007', '%d,%m,%Y'), 'K'),
8     ('11303044848', 'Patrycja', 'Działowa', STR_TO_DATE('30,10,2011', '%d,%m,%Y'), 'K'),
9     ('17272988431', 'Mateusz', 'Grabacki', STR_TO_DATE('29,07,2017', '%d,%m,%Y'), 'M');
10 END$$
11 DELIMITER ;

```

```
1 • CALL wstawNiepełnoletnich();
```

Wstawianie 50 osób pełnoletnich (45 osób w wieku do 60 lat, 5 osób w wieku powyżej 60 lat):

```
1 DELIMITER $$
2 • CREATE PROCEDURE wstawPełnoletnich()
3 BEGIN
4     INSERT INTO ludzie VALUES
5     (6, '80050995217', 'Gniewomir', 'Cieślak', STR_TO_DATE('09,05,1980', '%d,%m,%Y'), 'M'),
6     (7, '83111333733', 'Miłosz', 'Andrzejewski', STR_TO_DATE('13,11,1983', '%d,%m,%Y'), 'M'),
7     (8, '73102025914', 'Aleksander', 'Walczak', STR_TO_DATE('20,10,1973', '%d,%m,%Y'), 'M'),
8     (9, '01261952718', 'Martin', 'Błaszczak', STR_TO_DATE('19,06,2001', '%d,%m,%Y'), 'M'),
9     (10, '81103046687', 'Agnieszka', 'Zalewska', STR_TO_DATE('30,10,1981', '%d,%m,%Y'), 'K'),
10    (11, '97022879686', 'Emilia', 'Szymańska', STR_TO_DATE('28,02,1997', '%d,%m,%Y'), 'K'),
11    (12, '63080985456', 'Daniel', 'Szewczyk', STR_TO_DATE('09,08,1963', '%d,%m,%Y'), 'M'),
12    (13, '89070497165', 'Agata', 'Sokołowska', STR_TO_DATE('04,07,1989', '%d,%m,%Y'), 'K'),
13    (14, '03261083672', 'Alfred', 'Przybylski', STR_TO_DATE('10,06,2003', '%d,%m,%Y'), 'M'),
14    (15, '74021058685', 'Dominika', 'Jasińska', STR_TO_DATE('10,02,1974', '%d,%m,%Y'), 'K'),
15    (16, '62071244349', 'Paula', 'Kubiak', STR_TO_DATE('12,07,1962', '%d,%m,%Y'), 'K'),
16    (17, '73101898939', 'Fabian', 'Cieślak', STR_TO_DATE('18,10,1973', '%d,%m,%Y'), 'M'),
17    (18, '98121029235', 'Hubert', 'Kwiatkowski', STR_TO_DATE('10,12,1998', '%d,%m,%Y'), 'M'),
18    (19, '67051995835', 'Norbert', 'Nowak', STR_TO_DATE('19,05,1967', '%d,%m,%Y'), 'M'),
19    (20, '91122535635', 'Kamil', 'Sawicki', STR_TO_DATE('25,12,1991', '%d,%m,%Y'), 'M'),
20    (21, '65080527586', 'Lara', 'Kamińska', STR_TO_DATE('05,08,1965', '%d,%m,%Y'), 'K'),
21    (22, '71070856417', 'Antoni', 'Czarnecki', STR_TO_DATE('08,07,1971', '%d,%m,%Y'), 'M'),
22    (23, '00250356799', 'Damian', 'Brzeziński', STR_TO_DATE('03,05,2000', '%d,%m,%Y'), 'M'),
23    (24, '79042366973', 'Ignacy', 'Krupa', STR_TO_DATE('23,04,1979', '%d,%m,%Y'), 'M'),
24    (25, '74051452949', 'Diana', 'Brzezińska', STR_TO_DATE('14,05,1974', '%d,%m,%Y'), 'K'),
25    (26, '85091458914', 'Emil', 'Jakubowski', STR_TO_DATE('14,09,1985', '%d,%m,%Y'), 'M'),
26    (27, '99071018643', 'Emilia', 'Kubiak', STR_TO_DATE('10,07,1999', '%d,%m,%Y'), 'K'),
27    (28, '76091788745', 'Amalia', 'Piotrowska', STR_TO_DATE('17,09,1976', '%d,%m,%Y'), 'K'),
28    (29, '76021725563', 'Kinga', 'Kubiak', STR_TO_DATE('17,02,1976', '%d,%m,%Y'), 'K'),
29    (30, '62010961575', 'Remigiusz', 'Sawicki', STR_TO_DATE('09,01,1962', '%d,%m,%Y'), 'M'),
30    (31, '74011121791', 'Miron', 'Błaszczak', STR_TO_DATE('11,01,1974', '%d,%m,%Y'), 'M'),
31    (32, '85040344558', 'Natan', 'Lewandowski', STR_TO_DATE('03,04,1985', '%d,%m,%Y'), 'M'),
32    (33, '87090872863', 'Marta', 'Szczepańska', STR_TO_DATE('08,09,1987', '%d,%m,%Y'), 'K'),
33    (34, '02212935969', 'Patrycja', 'Szulc', STR_TO_DATE('29,01,2002', '%d,%m,%Y'), 'K'),
34    (35, '66031345372', 'Piotr', 'Szczepański', STR_TO_DATE('13,03,1966', '%d,%m,%Y'), 'M'),
35    (36, '78041216319', 'Gustaw', 'Woźniak', STR_TO_DATE('12,04,1978', '%d,%m,%Y'), 'M'),
36    (37, '67062971781', 'Klaudia', 'Baran', STR_TO_DATE('29,06,1967', '%d,%m,%Y'), 'K'),
37    (38, '97061393839', 'Julian', 'Szymczak', STR_TO_DATE('13,06,1997', '%d,%m,%Y'), 'M'),
38    (39, '90042622386', 'Marlena', 'Górska', STR_TO_DATE('26,04,1990', '%d,%m,%Y'), 'K'),
39    (40, '65111865421', 'Adrianna', 'Piotrowska', STR_TO_DATE('18,11,1965', '%d,%m,%Y'), 'K'),
40    (41, '86011873639', 'Łukasz', 'Kozłowski', STR_TO_DATE('18,01,1986', '%d,%m,%Y'), 'M'),
41    (42, '92060379514', 'Adam', 'Zalewski', STR_TO_DATE('03,06,1992', '%d,%m,%Y'), 'M'),
42    (43, '79031268596', 'Amir', 'Mróz', STR_TO_DATE('12,03,1979', '%d,%m,%Y'), 'M'),
43    (44, '00210873434', 'Hubert', 'Brzeziński', STR_TO_DATE('08,01,2000', '%d,%m,%Y'), 'M'),
44    (45, '76061529888', 'Czesława', 'Lis', STR_TO_DATE('15,06,1976', '%d,%m,%Y'), 'K'),
45    (46, '94030713311', 'Natan', 'Szczepański', STR_TO_DATE('07,03,1994', '%d,%m,%Y'), 'M'),
46    (47, '66060585747', 'Jadwiga', 'Kubiak', STR_TO_DATE('05,06,1966', '%d,%m,%Y'), 'K'),
47    (48, '64040639617', 'Gracjan', 'Baran', STR_TO_DATE('06,04,1964', '%d,%m,%Y'), 'M'),
```



```

48      (49, '91080489984', 'Zofia', 'Ziółkowska', STR_TO_DATE('04,08,1991', '%d,%m,%Y'), 'K'),
49      (50, '88050624418', 'Arkadiusz', 'Wróblewski', STR_TO_DATE('06,05,1988', '%d,%m,%Y'), 'M'),
50      (51, '52100541437', 'Mirosław', 'Kalisz', STR_TO_DATE('05,10,1952', '%d,%m,%Y'), 'M'),
51      (52, '57093013167', 'Janina', 'Tucholska', STR_TO_DATE('30,09,1957', '%d,%m,%Y'), 'K'),
52      (53, '53061922248', 'Marianna', 'Wojciechowska', STR_TO_DATE('19,06,1953', '%d,%m,%Y'), 'K'),
53      (54, '59031917867', 'Józefa', 'Biernacka', STR_TO_DATE('19,03,1959', '%d,%m,%Y'), 'K'),
54      (55, '54112957482', 'Halina', 'Jarosz', STR_TO_DATE('29,11,1954', '%d,%m,%Y'), 'K');
55  END$$
56  DELIMITER ;

1 • call wstawPelnoletnich();

```

Wstawianie zawodów do tabeli Zawody:

```

1 • INSERT INTO zawody VALUES
2      (1, 'polityk', 3521.0, 9823.77),
3      (2, 'nauczyciel', 2949.53, 4046.89),
4      (3, 'lekarz', 3200.0, 8593.21),
5      (4, 'informatyk', 3700.23, 11234.21);

```

Tworzenie procedury, a w niej kursora do tworzenia rekordów tabeli Pracownicy:

```

1  DELIMITER $$
2  • CREATE PROCEDURE generatorPracownikow()
3  BEGIN
4      DECLARE done INT DEFAULT FALSE;
5      DECLARE l_id, z_id, z_lekarza INT;
6      DECLARE wiek DATE;
7      DECLARE gender enum('K', 'M');
8      DECLARE placa, placa_min, placa_max FLOAT;
9
10     DECLARE cur1 CURSOR FOR SELECT id, data_urodzenia, plec FROM ludzie;
11     DECLARE CONTINUE HANDLER
12     FOR NOT FOUND SET done = TRUE;
13     SET z_lekarza = (SELECT zawod_id FROM zawody WHERE nazwa='lekarz');
14     OPEN cur1;
15
16     read_loop: LOOP
17         FETCH cur1 INTO l_id, wiek, gender;
18         IF done THEN
19             LEAVE read_loop;
20         END IF;
21         SET z_id = FLOOR(RAND()*4 + 1);
22         IF (DATE_ADD(wiek, INTERVAL 65 YEAR) < CURDATE() OR (DATE_ADD(wiek, INTERVAL 60 YEAR) < CURDATE() AND gender = 'K')) THEN
23             WHILE (z_id=z_lekarza) DO
24                 SET z_id = FLOOR(RAND()*4 + 1);
25             END WHILE;
26         END IF;
27
28         SET placa_min = (SELECT zawody.pensja_min FROM zawody WHERE zawod_id = z_id);
29         SET placa_max = (SELECT zawody.pensja_max FROM zawody WHERE zawod_id = z_id);
30         INSERT INTO pracownicy VALUES
31         (l_id, z_id, ROUND( (placa_min + RAND()*(placa_max-placa_min)), 2) );
32     END LOOP;
33
34     CLOSE cur1;
35  END$$
36  DELIMITER ;

1 • call generatorPracownikow();

```

2.1.3. Rezultat

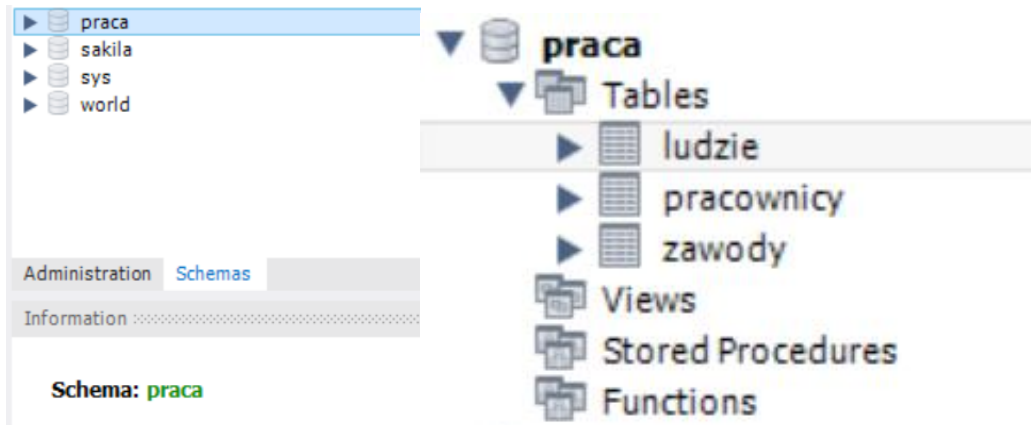


Table: **ludzie**

Columns:

id int AI PK
PESEL char(11)
imie varchar(30)
nazwisko varchar(30)
data_urodzenia date
plec enum('K','M')

Table: **pracownicy**

Columns:
ludzie_id int
zawod_id int
pensja float

Table: **zawody**

Columns:
zawod_id int AI PK
nazwa varchar(50)
pensja_min float
pensja_max float

id	PESEL	imie	nazwisko	data_urodzenia	plec
1	06290499162	Anna	Nowak	2006-09-04	K
2	17240578972	Jan	Kowalski	2017-04-05	M
3	07221172426	Amelia	Wolska	2007-02-11	K
4	11303044848	Patrycja	Działowa	2011-10-30	K
5	17272988431	Mateusz	Grabacki	2017-07-29	M
6	80050995217	Gniewomir	Cieślak	1980-05-09	M
7	83111333733	Mikołaj	Andrzejewski	1983-11-13	M
8	73102025914	Aleksander	Walczak	1973-10-20	M
9	01261952718	Martin	Błaszczak	2001-06-19	M
10	81103046687	Agnieszka	Zalewska	1981-10-30	K
11	97022879686	Emilia	Szymańska	1997-02-28	K
12	63080985456	Daniel	Szewczyk	1963-08-09	M
13	89070497165	Agata	Sokołowska	1989-07-04	K
14	03261083672	Alfred	Przybylski	2003-06-10	M
15	74021058685	Dominika	Jasińska	1974-02-10	K
16	62071244349	Paula	Kubiak	1962-07-12	K
17	73101898939	Fabian	Cieślak	1973-10-18	M
18	98121029235	Hubert	Kwiatkowski	1998-12-10	M
19	67051995835	Norbert	Nowak	1967-05-19	M
20	91122535635	Kamil	Sawicki	1991-12-25	M

zawod_id	nazwa	pensja_min	pensja_max
1	polityk	3521	9823.77
2	nauczyciel	2949.53	4046.89
3	lekarz	3200	8593.21
4	informatyk	3700.23	11234.2
NULL	NULL	NULL	NULL

ludzie_id	zawod_id	pensja
1	2	3615.85
2	3	3301.57
3	2	3929.44
4	2	3824.11
5	1	4188.86
6	1	8817.95
7	3	3276.85
8	3	7012.92
9	4	5994.2
10	4	6881.94
11	3	4729.66
12	4	8130.84
13	2	3630.94
14	2	3504.1
15	3	3271.91
16	4	9012.15
17	2	4600.45

Pensje są odpowiedniej wielkości (z zakresu [pensja_min, pensja_max]), brak wyników dla zapytania o pensje niezgodne:

```
1 • FROM praca.pracownicy INNER JOIN praca.zawody ON pracownicy.zawod_id = zawody.zawod_id WHERE pensja > pensja_max OR pensja < pensja_min;
```

ludzie_id	zawod_id	pensja	zawod_nazwa	pensja_min	pensja_max
-----------	----------	--------	-------------	------------	------------

Emeryci nie są lekarzami (mężczyźni powyżej 65 i kobiety powyżej 60 roku życia):

```
1 • SELECT data_urodzenia, plec, nazwa FROM praca.pracownicy INNER JOIN praca.ludzie ON pracownicy.ludzie_id=ludzie.id
2 INNER JOIN praca.zawody ON pracownicy.zawod_id=zawody.zawod_id
3 WHERE
4 (DATE_ADD(data_urodzenia, INTERVAL 65 YEAR) <= CURDATE() AND plec = 'M')
5 OR (DATE_ADD(data_urodzenia, INTERVAL 60 YEAR) <= CURDATE() AND plec = 'K');
```

data_urodzenia	plec	nazwa
1953-06-19	K	polityk
1959-03-19	K	polityk
1954-11-29	K	nauczyciel
1952-10-05	M	informatyk
1957-09-30	K	informatyk

2.1.4. Opis rozwiązania

Najpierw tworzę bazę danych Praca (CREATE DATABASE IF NOT EXISTS). Potem zajmuję się tworzeniem tabel wraz z jednoczesnym przypisaniem constraintów i oznaczeniem kluczy. W związku z tym:

- Dla Tabeli Ludzie tworzę pola zgodne z treścią zadania plus dodaję własny identyfikator, id. Zezwalam na wartość NULL w polu PESEL. Następnie oznaczam id kluczem głównym tabeli i dodaję constrainty (warunki dla rekordów), dotyczą one poprawności numeru PESEL: **pesel_poprawna_data** porównuje datę zamieszczoną w pierwszych 6 cyfrach numeru PESEL z polem data_urodzenia, **pesel_poprawna_plec** porównuje płeć zakodowaną w 10 cyfrze numeru PESEL z polem enum plec, **pesel_poprawna_suma_kontrolna** wykonuje obliczenia podane na stronie rządowej potrzebne do zweryfikowania ostatniej cyfry numeru PESEL przy pomocy pozostałych 10 cyfr. Każdy z warunków stosuje funkcję SUBSTRING() dzięki której pobiera odpowiedni fragment numeru PESEL; dodatkowo przy sprawdzaniu sumy kontrolnej korzystam z funkcji CAST(... AS UNSIGNED) aby móc skorzystać z operacji dzielenia modulo w funkcji MOD().
- Uważam, że numer PESEL sam w sobie jest złym kluczem głównym z paru względów: po pierwsze jest ograniczony (za setki lat nie będzie dostępnych numerów PESEL), po drugie w bazie danych możemy chcieć uwzględnić obcokrajowców z polem PESEL równym NULL, po trzecie często zdarzały się pomyłki przy nadawaniu numerów PESEL, przez co klucz główny byłby narażony na potrzeby modyfikacji. Ponadto numer ciąg numerów PESEL jest nieintuicyjny (brak automatycznej inkrementacji często stosowanej w kluczu głównym).
- Następnie ręcznie dodaję rekordy do tabel Ludzie i Zawody, zgodne z warunkami w treści zadania. Przy dodawaniu numerów PESEL korzystam z internetowego generatora poprawnych numerów PESEL.
- Tworzę procedurę (procedury omawiane były już na poprzedniej liście) a w niej tworzę kursor (DECLARE CURSOR nazwa FOR) i zapisuję po jakim zapytaniu ma przechodzić (tu precyzuję jakie dane otrzyma kursor z tabeli z pomocą SELECT dane FROM tabela_do_przejścia). Deklaruję zmienne pomocnicze pomocne do pobierania danych z kursora i zapisania id

zawodu lekarza, gdyż dla niego będą dodatkowe warunki. Ponadto tworzę zmienną `done` typu `INT` o wartości początkowej `FALSE` (u nas posłuży jako swego rodzaju boolean) oraz zmienne typu `float` do pobierania widełek płacowych danego zawodu.

- Deklaruję obsługę wyjątku `NOT FOUND` (czyli nie znaleziono już kolejnego rekordu), w którym ustawiam zmienną `done` na `TRUE`. Otwieram kursor słowem `OPEN`
- Tworzę pętlę, w której kursor będzie pobierał dane dopóki będą jakieś do pobrania (gdy nie ich nie będzie, zmienna `done` ustawi się na `TRUE` co spowoduje wejście do instrukcji warunkowej `IF`, w której opuszczona zostanie pętla). Dane są pobierane z użyciem słowa kluczowego `FETCH` i są zapisywane do pomocniczych zmiennych słowem `INTO`.
- Generuję losowe id zawodu korzystając z funkcji `FLOOR()` i `RAND()` oraz sprawdzam wiek i płeć osoby – jeśli jest to kobieta powyżej 60 lub mężczyzna powyżej 65 roku życia, sprawdzam czy zostało dla tej osoby przygotowane id zawodu równe id lekarza. Jeśli tak, losuję zawód aż zostanie on zmieniony na inny, dzięki temu zostaje zachowana losowość, co prawda kosztem czasu wykonania (jest szansa na wielokrotne wylosowywanie niechcianego id).
- Pobieram widełki płacowe dla zawodu wylosowanego w kolejnym kroku i tworzę rekord z `ludzie_id` pobranym z kursora, `zawod_id` wygenerowanym dwa kroki wcześniej i pensją generowaną losowo z zakresu [`pensja_min`, `pensja_max`].

2.2. Zadanie 2

2.2.1. Polecenie

W tabeli Ludzie utwórz indeks złożony na kolumnach plec oraz imie, natomiast w Pracownicy utwórz index na kolumnie pensja. Przyjrzyj się poleceniom SHOW INDEX. . . oraz EXPLAIN SELECT. . .

Za pomocą odpowiednich kwerend SQL wyciągnij z bazy dane dotyczące:

- wszystkich kobiet, których imię zaczyna się na 'A'
- wszystkich kobiet,
- wszystkich osób, których imię zaczyna się na 'K',
- wszystkich osób zarabiających poniżej 2000,
- wszystkich informatyków płci męskiej, zarabiających powyżej 10000.

Następnie odpowiedz na pytania:

- Jakie mamy obecnie indeksy założone dla obu tabel?
- W przypadku których zapytań optymalizator użyje indeksu/indeksów?

Do sprawozdania wpisz użyte polecenia SQL oraz odpowiedzi na pytania. (3pkt)

2.2.2. Kwerendy niezbędne do realizacji

Najpierw utworzenie podanych indeksów:

```
1 • CREATE INDEX indeks_ludzie ON ludzie(plec, imie);  
1 • CREATE INDEX indeks_pracownicy ON pracownicy(pensja);
```

Kwerendy wyciągające dane z baz:

```
1 • SELECT * FROM Ludzie WHERE plec='K' AND imie LIKE 'A%';  
1 • SELECT * FROM Ludzie WHERE plec='K';  
1 • SELECT * FROM Ludzie WHERE imie LIKE 'K%';  
1 • SELECT * FROM Ludzie INNER JOIN Pracownicy ON Ludzie.id=Pracownicy.ludzie_id WHERE pensja < 2000;  
1 • SELECT * FROM Ludzie INNER JOIN Pracownicy ON Ludzie.id=Pracownicy.ludzie_id  
2 INNER JOIN Zawody ON Pracownicy.zawod_id=Zawody.zawod_id WHERE pensja > 10000 AND Zawody.nazwa = 'informatyk' AND Ludzie.plec='M';
```

2.2.3. Rezultat

Indeksy występujące w tabeli Ludzie:

```
1 • SHOW INDEX FROM praca.ludzie;
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
ludzie	0	PRIMARY	1	id	A	0				BTREE			YES	
ludzie	1	indeks_ludzie	1	plec	A	2				BTREE			YES	
ludzie	1	indeks_ludzie	2	imie	A	51				BTREE			YES	

Indeksy występujące w tabeli Pracownicy:

1 • SHOW INDEX FROM praca.pracownicy;

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
pracownicy	1	ludzie_id	1	ludzie_id	A	0	NULL	NULL		BTREE			YES	NULL
pracownicy	1	zawod_id	1	zawod_id	A	0	NULL	NULL		BTREE			YES	NULL
pracownicy	1	indeks_pracownicy	1	pensja	A	55	NULL	NULL	YES	BTREE			YES	NULL

Sprawdzenie, w przypadku których zapytań optymalizator użyje indeksu/indeksów:

1 • EXPLAIN SELECT * FROM Ludzie WHERE plec='K' AND imie LIKE 'A%';

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Ludzie	NULL	range	indeks_ludzie	indeks_ludzie	123	NULL	6	100.00	Using index condition

1 • EXPLAIN SELECT * FROM Ludzie WHERE plec='K';

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Ludzie	NULL	ref	indeks_ludzie	indeks_ludzie	1	const	25	100.00	Using index condition

1 • EXPLAIN SELECT * FROM Ludzie WHERE imie LIKE 'K%';

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Ludzie	NULL	ALL	NULL	NULL	NULL	NULL	55	11.11	Using where

1 • EXPLAIN SELECT * FROM Ludzie INNER JOIN Pracownicy ON Ludzie.id=Pracownicy.ludzie_id WHERE pensja < 2000;

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Pracownicy	NULL	range	ludzie_id,indeks_pracownicy	indeks_pracownicy	5	NULL	1	100.00	Using index condition
1	SIMPLE	Ludzie	NULL	eq_ref	PRIMARY	PRIMARY	4	praca.Pracownicy.ludzie_id	1	100.00	NULL

1 • EXPLAIN SELECT * FROM Ludzie INNER JOIN Pracownicy ON Ludzie.id=Pracownicy.ludzie_id
2 INNER JOIN Zawody ON Pracownicy.zawod_id=Zawody.zawod_id WHERE pensja > 10000 AND Zawody.nazwa = 'informatyk' AND Ludzie.plec='M';

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	Pracownicy	NULL	range	ludzie_id,zawod_id,indeks_pracownicy	indeks_pracownicy	5	NULL	1	100.00	Using index condition
1	SIMPLE	Zawody	NULL	eq_ref	PRIMARY	PRIMARY	4	praca.Pracownicy.zawod_id	1	25.00	Using where
1	SIMPLE	Ludzie	NULL	eq_ref	PRIMARY,indeks_ludzie	PRIMARY	4	praca.Pracownicy.ludzie_id	1	54.55	Using where

2.2.4. Opis rozwiązania

W rezultacie zamieszczona została składnia potrzebna do sprawdzenia z jakich indeksów korzysta optymalizator dla poszczególnych zapytań oraz jakie indeksy znajdują się w obu tabelach:

Domyślnie tabele mają indeksy po kluczach głównych i obcych.

Po utworzeniu w tabeli Ludzie indeksu złożonego (na dwóch kolumnach, plec i imie) dodane zostały dwa indeksy składowe o tej samej nazwie (nazwie podanej przy tworzeniu klucza), stąd dla nich pole `non_unique` ustawione jest na jeden. Zatem dla tabeli Ludzie mamy 3 indeksy, z czego dwa składają się na jeden złożony.

W tabeli Pracownicy domyślnie znajdowały się dwa indeksy na kolumnach odpowiednio `ludzie_id` i `zawod_id` (klucze obce). Dodany został jeden indeks, o nazwie zadanej przy tworzeniu i na kolumnie `pensja`.

Wszystkie indeksy są typu BTREE (drzewo binarne).

Z pomocą EXPLAIN możemy się dowiedzieć, przy których zapytaniach wykorzystywane są indeksy (kolumna `key`). Można też zobaczyć jakie indeksy mogły zostać wykorzystane do danego zapytania (kolumna `possible keys`).

Dla pierwszych dwóch zapytań wykorzystany został indeks *indeks_ludzie* (utworzony w zadaniu). Przy obu zapytaniach pomaga przy filtrowaniu zapytań dla konkretnej płci. Indeksy nie pomagają przy bardziej złożonych warunkach, np. LIKE wzorzec, dlatego do trzeciego zapytania optymalizator nie stosuje żadnego indeksu.

Dla ostatnich dwóch zapytań wykorzystywane są indeksy PRIMARY (na kluczach głównych) przy tworzeniu złączenia INNER JOIN typu (klucz-klucz_obcy). Dla samego warunku WHERE wykorzystywany jest indeks *indeks_pracownicy* (usprawnia warunek `pensja < 2000 / pensja > 10000`) ale w ostatnim przykładzie mógłby również zostać wykorzystany indeks *indeks_ludzie* (znajduje się w `possible keys`, dla usprawnienia filtrowania płci).

2.3. Zadanie 3

2.3.1. Polecenie

Napisz procedurę, która jako parametr wejściowy przyjmuje nazwę zawodu, a następnie daje wszystkim wykonującym ten zawód 5% podwyżki przy zachowaniu ograniczeń wynikających z widełek płacowych w tabeli zawody. Operacja powinna wykonać się transakcyjnie, tzn. albo wszyscy pracownicy danego zawodu dostają podwyżkę albo, przy przekroczeniu widełek przez przynajmniej jedną osobę, nikt. Umieść procedur w sprawozdaniu.

2.3.2. Kwerendy niezbędne do realizacji

```
1 DELIMITER $$
2 CREATE PROCEDURE podwyzka(IN z_nazwa VARCHAR(50))
3 BEGIN
4     DECLARE z_id INT DEFAULT 0;
5     SET z_id = (SELECT DISTINCT zawod_id FROM zawody WHERE nazwa = z_nazwa);
6     START TRANSACTION;
7     UPDATE Pracownicy SET pensja=pensja*1.05 WHERE Pracownicy.zawod_id=z_id;
8     IF ( (SELECT COUNT(pensja) FROM Pracownicy
9         WHERE zawod_id=z_id AND pensja > (SELECT pensja_max FROM zawody WHERE zawod_id=z_id)) > 0) THEN
10         ROLLBACK;
11     END IF;
12     COMMIT;
13 END$$
14 DELIMITER ;
15
```

```
1 CALL podwyzka('polityk');
```

```
1 CALL podwyzka('informatyk');
```

2.3.3. Rezultat

Przed wywołaniem procedury sprawdzam dla których zawodów powinna się nie udać:

```
1 SELECT * FROM Pracownicy INNER JOIN Zawody ON Pracownicy.zawod_id=Zawody.zawod_id
2 WHERE pensja_max < pensja*1.05;
```

ludzie_id	zawod_id	pensja	zawod_id	nazwa	pensja_min	pensja_max
52	1	9377.16	1	polityk	3521	9823.77
10	2	3926.78	2	nauczyciel	2949.53	4046.89
54	2	3930.08	2	nauczyciel	2949.53	4046.89
47	3	8406	3	lekarz	3200	8593.21

W związku z tym podwyżka nie powiedzie się dla polityków, nauczycieli i lekarzy, wykonany zostanie rollback.

Próba podwyżki dla polityków (ID równe 1) się nie powiedzie (przed i po):

```
1 • SELECT * FROM praca.pracownicy WHERE zawod_id=1;
```

	ludzie_id	zawod_id	pensja
▶	4	1	4830.62
	12	1	9203.21
	16	1	6049.05
	18	1	9321.47
	19	1	3672.93
	25	1	6992.55
	30	1	6484.19
	31	1	9326.74
	33	1	8861.72
	38	1	6783.08
	43	1	5791.16
	45	1	5416.25
	46	1	8719.65
	52	1	9377.16

```
1 • SELECT * FROM praca.pracownicy WHERE zawod_id=1;
```

	ludzie_id	zawod_id	pensja
▶	4	1	4830.62
	12	1	9203.21
	16	1	6049.05
	18	1	9321.47
	19	1	3672.93
	25	1	6992.55
	30	1	6484.19
	31	1	9326.74
	33	1	8861.72
	38	1	6783.08
	43	1	5791.16
	45	1	5416.25
	46	1	8719.65
	52	1	9377.16

Udana podwyżka dla informatyków (przed i po):

```
1 • SELECT * FROM praca.pracownicy WHERE zawod_id=4;
```

	ludzie_id	zawod_id	pensja
▶	1	4	5554.68
	6	4	10439.9
	7	4	4912.79
	8	4	3751.86
	17	4	8272.25
	21	4	4551.77
	22	4	7995.67
	27	4	10259.3
	34	4	8719.76
	35	4	3945.55
	37	4	7712.74
	42	4	8788.28
	48	4	10609.5
	50	4	9330.22
	53	4	4015.07

```
1 • SELECT * FROM praca.pracownicy WHERE zawod_id=4;
```

	ludzie_id	zawod_id	pensja
▶	1	4	5832.41
	6	4	10961.9
	7	4	5158.43
	8	4	3939.45
	17	4	8685.86
	21	4	4779.36
	22	4	8395.45
	27	4	10772.3
	34	4	9155.75
	35	4	4142.83
	37	4	8098.38
	42	4	9227.69
	48	4	11139.9

pracownicy 5 ×

2.3.4. Opis rozwiązania

W procedurze tworzę deklaruję zmienną pomocniczą typu INT, w której zapisuję ID zawodu, którego nazwa została przekazana w parametrze (domyślnie równa zero, nikt nie dostanie podwyżki przy błędnej nazwie). Następnie tworzę transakcję, zaczynam ją słowami START TRANSACTION. Wewnątrz transakcji zwiększam wartości pensji pracowników danego zawodu o 5% i sprawdzam czy po wykonaniu podwyżki jakikolwiek pracownik z tego zawodu przekroczył widełki płacowe. Jeśli przynajmniej jeden przekroczył, cofam transakcję słowem kluczowym ROLLBACK. Jeśli wszyscy zmieścili się w widełkach, zatwierdzam transakcję słowem kluczowym COMMIT.

2.4. Zadanie 4

2.4.1. Polecenie

Za pomocą konstrukcji PREPARE statement przygotuj zapytanie zwracające liczbę kobiet, pracujących w zawodzie o podanej przy EXECUTE nazwie. Umieść konstrukcję i jej przykładowe wywołanie w sprawozdaniu.

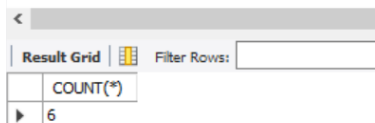
2.4.2. Kwerendy niezbędne do realizacji

```
1 • PREPARE statement FROM "SELECT COUNT(*) FROM Pracownicy INNER JOIN Zawody ON Pracownicy.zawod_id=Zawody.zawod_id  
2   INNER JOIN Ludzie ON Pracownicy.ludzie_id=Ludzie.id WHERE plec='K' AND nazwa=?";
```

2.4.3. Rezultat

Przykładowe wywołanie prepared statement przy pomocy EXECUTE:

```
1 • SET @a = 'lekarz';  
2 • EXECUTE statement USING @a;
```



COUNT(*)
6

2.4.4. Opis rozwiązania

Przy pomocy słowa kluczowego PREPARE tworzę prepared statement o nazwie *statement*. Spowoduje to, że polecenie wzięte w cudzysłów i zapisane po słowie FROM zostanie skompilowane i „przygotowane” w pamięci, dzięki czemu będzie szybciej dostępne (dlatego warto stosować je dla zapytań które często się powtarzają). W poleceniu można zawrzeć konieczność podania parametru poprzez użycie symbolu kluczowego ‘?’.

Aby wywołać sformułowanie, należy zadeklarować zmienną użytkownika (@nazwa_zmiennej) i przypisać do niej wartość, a następnie skorzystać ze słowa kluczowego EXECUTE nazwa_prepared_statement USING @nazwa_zmiennej;.

Prepared statements mogą też zabezpieczyć w pewnym stopniu przed SQL Injection, nie pozwalają na dopisanie w parametrze kodu szkodliwego dla bazy danych, oddzielając tym samym polecenie od bazy danych

2.5. Zadanie 5

2.5.1. Polecenie

Zrób backup bazy danych tej listy. Usuń bazę danych, a następnie ją przywróć z backupu. Do sprawozdania wrzuć krótki raport z wykonanych czynności. Jaka jest różnica między backupem pełnym a różnicowym?

2.5.2. Działania niezbędne do realizacji

```
C:\Users\Swmar>mysqldump -u root -p --databases praca > test_dump.sql
Enter password: *****

C:\Users\Swmar>
```

```
C:\Users\Swmar>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 24
Server version: 8.0.26 MySQL Community Server - GPL

Copyright (c) 2000, 2021, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> DROP SCHEMA praca;
Query OK, 3 rows affected (0.13 sec)

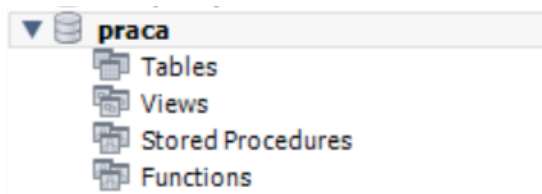
mysql> CREATE SCHEMA praca;
Query OK, 1 row affected (0.02 sec)
```

```
C:\Users\Swmar>mysql -u root -p praca < test_dump.sql
Enter password: *****

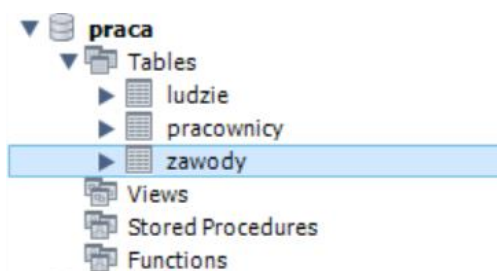
C:\Users\Swmar>
```

2.5.3. Rezultat

Pusta baza danych praca:



Po wczytaniu kopii zapasowej:



Jedna z tablic, rekordy zostały zachowane:

	zawod_id	nazwa	pensja_min	pensja_max
▶	1	polityk	3521	9823.77
	2	nauczyciel	2949.53	4046.89
	3	lekarz	3200	8593.21
	4	informatyk	3700.23	11234.2
•	NULL	NULL	NULL	NULL

2.5.4. Opis rozwiązania

Na wstępie odpowiem na pytanie zadane pod koniec zadania. Pełny backup zawiera wszystkie informacje dotyczące bazy danych (tablice z wszystkimi właściwościami). Backup różnicowy jest formą zapisu tylko różnic w bazie danych obecnych w danym momencie względem ostatniego pełnego backupu. Taka metoda ma swoje plusy i minusy. Zaletą jest znacznie mniejszy rozmiar takiego backupu (w szczególności w wielkich bazach danych, które nie są szczególnie dynamiczne). Do wad można zaliczyć konieczność uprzedniego wykonania pełnego backupu i konieczność zachowania go w celu odtwarzania bazy danych – bez niego backup różnicowy jest bezużyteczny. Co więcej, w przypadku wielu wykonanych kolejno backupów różnicowych, odtworzenie najbardziej aktualnej wersji bazy danych jest czasochłonne, trzeba przejść przez wszystkie poprzednie backupy różnicowe zrobione po ostatnim pełnym backupie.

Najpierw tworzę kopię zapasową z wykorzystaniem polecenia mysqldump (należy zadbać o to by ścieżka do programu była dodana do zmiennych środowiskowych systemu). Flaga -u specyfikuje nazwę użytkownika wykonującego operację (korzystam z roota, by mieć wszystkie przywileje, zwykle lepiej zrobić osobnego użytkownika/rolę do obsługi kopii zapasowych), flaga -p zaznacza, że chcemy wprowadzić hasło tego użytkownika, a --databases specyfikuje, że będzie mieć miejsce kopia baz danych. Po flagach wpisuję nazwa_bazy > nazwa_pliku.sql. Utworzona zostanie kopia zapasowa w domyślnym folderze użytkownika o nazwie podanej po znaku '>'.

Usuwa bazę danych (DROP SCHEMA praca;) i tworzę nową, pustą bazę o tej samej nazwie. Wgrywam kopię zapasową do pustej bazy danych poleceniem mysql -u root -p nazwa_bazy<nazwa_pliku.sql. Flagi w tym przypadku są analogiczne, a znak '<' wskazuje, że operacja zachodzi w drugą stronę (wczytujemy, nie zapisujemy). Po wykonaniu polecenia baza została odzyskana, konkretniej jej tabele z właściwościami (indeksy, triggerzy, itp.). Nie zostają odtworzone procedury i funkcje.

2.6. Zadanie 6

2.6.1. Polecenie

Zapoznaj się z celowo niezabezpieczoną aplikacją internetową WebGoat (<https://github.com/WebGoat/Webgoat/>). Uruchom ją i wykonaj lekcje dotyczące SQL Injection, znajdujące się w części A1:

- SQL Injection (introduction) – za 2 punkty
- SQL Injection (advanced) – za 4 punkty
- SQL Injection (mitigation) – za 4 punkty

W sprawozdaniu umieść raport z wykonanych ćwiczeń (rozwiązania ćwiczeń i wnioski)

2.6.2. Raport z SQL Injection (introduction)

Na początku znajduje się wprowadzenie do tego, co już poznaliśmy w sporej części na kursie: podstawy DQL, DML, DDL, DCL.

W lekcji 6 znajduje się wprowadzenie do SQL Injection, przedstawione jako przemycenie do interpretera aplikacji SQL szkodliwego kodu np. poprzez wprowadzenie go w pole tekstowe do wpisywania imienia. Wtedy wystarczy, żeby osoba z uprawnieniami wyświetliła dane zainfekowanej tablicy, a zostaną również wykonane niechciane polecenia. Przykład:

Username:

```
"SELECT * FROM users WHERE name = 'Smith'; DROP TABLE employees; SELECT * FROM users WHERE TRUE OR name = 'Smith';"
```

W lekcji 7 opisane są skutki dobrze wykonanego SQL Injection, potrafią być opłakane.

W lekcji 8 wymieniono ograniczenia SQL Injection; niektóre systemy bazodanowe nie zezwalają na tworzenie łańcuchów poleceń, przez co SQL Injection przestaje działać. Najbardziej podatne na SQL Injection są starsze systemy z PHP, Classic ASP, Cold Fusion. Znaczącym ograniczeniem może być również nieznanie dialektu używanego przez dany serwer.

Kolejne lekcje to ćwiczenie poprawnego wykonania SQL Injection.

2.6.2.1. Zadanie 1 (lekcja 2)

☒

SQL query

Submit

You have succeeded!

SELECT employees.department FROM employees WHERE first_name='Bob' AND last_name='Franco';

DEPARTMENT

Marketing

Podstawowe zadanie z wyszukiwania danych z tabeli, tego typu zadania były na liście 1.

2.6.2.2. Zadanie 2 (lekcja 3)

☒

SQL query

Submit

Congratulations. You have successfully completed the assignment.

UPDATE employees SET employees.department = 'Sales' WHERE first_name='Tobi' AND last_name='Barnett';

USERID FIRST_NAME LAST_NAME DEPARTMENT SALARY AUTH_TAN

89762 Tobi Barnett Sales 77000 TA9LL1

Podstawowe zadanie z aktualizacji danych w tabeli.

2.6.2.3. Zadanie 3 (lekcja 4)

✓

SQL query

SQL query

Submit

Congratulations. You have successfully completed the assignment.
`ALTER TABLE employees ADD COLUMN phone varchar(20);`

Podstawowe zadanie z modyfikacji struktury tabeli (DDL).

2.6.2.4. Zadanie 4 (lekcja 5)

✓

SQL query

GRANT ALL ON grant_rights TO unauthorized_user;

Submit

Congratulations. You have successfully completed the assignment.

Podstawowe zadanie z przyznawania praw użytkownikowi (DCL).

2.6.2.5. Zadanie 5 (lekcja 9)

✓

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

Smith' ▾

or ▾

'1' = '1' ▾

Get Account Info

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 first_name = 'John' and last_name = 'Smith' or '1' = '1'
Explanation: This injection works, because or '1' = '1' always evaluates to true (The string ending literal for '1' is closed by the query itself, so you should not inject it). So the injected query basically looks like this: SELECT * FROM user_data WHERE first_name = 'John' and last_name = ' or TRUE', which will always evaluate to true, no matter what came before it.

Poprawne wykonanie SQL Injection przy niezabezpieczonej zmiennej tekstowej

2.6.2.6. Zadanie 6 (lekcja 10)

✓

Login_Count:

User_Id:

Get Account Info

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 = 65656 and userid= 45567 OR '1' = '1'

Udane SQL Injection przy podawaniu dwóch pól numerycznych do wyszukiwania rekordów zgodnych z nimi. Wrażliwe na SQL Injection jest drugie pole, ponieważ w nim można wprowadzić po wartości userid OR '1' = '1' (równoważne z OR TRUE). Wprowadzony warunek zawsze prawdziwy oddzielony od poprzednich słowem OR spowoduje wyświetlenie wszystkich rekordów.

2.6.2.7. Zadanie 7 (lekcja 11)

You already found out that the query performing your request looks like this:

✓

Employee Name:

Authentication TAN:

Get department

You have succeeded! You successfully compromised the confidentiality of data by viewing internal information that you should not have access to. Well done!

USERID	FIRST_NAME	LAST_NAME	DEPARTMENT	SALARY	AUTH_TAN	PHONE
32147	Paulina	Travers	Accounting	46000	P45JSI	null
34477	Abraham	Holman	Development	50000	UU2ALK	null
37648	John	Smith	Marketing	64350	3SL99A	null
89762	Tobi	Barnett	Sales	77000	TA9LL1	null
96134	Bob	Franco	Marketing	83700	LO9S2V	null

Zadanie bardzo podobne do poprzedniego – znając kwerendę realizowaną w systemie możemy przemyścić (odpowiednio manipulując apostrofami) dodatkowy warunek zawsze prawdziwy, oddzielony od pozostałych alternatywą.

2.6.2.8. Zadanie 8 (lekcja 12)

Dane wprowadzone w pola:

cokolwiek

cokolwiek'; UPDATE employees SET salary = 999999 WHERE auth_tan = '3SL99A

✓

Employee Name:

Authentication 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	999999	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

Modyfikacja bazy danych z pomocą SQL Injection i SQL query chaining – kwerendy w jednej linii oddzielone średnikiem.

2.6.2.9. Zadanie 9 (lekcja 13)

Wprowadzone dane do pola: **cokolwiek'; DROP TABLE access_log; SELECT * FROM employees WHERE last_name = '**

✓

Action contains:

Success! You successfully deleted the access_log table and that way compromised the availability of the data.

Skuteczne przemycenie polecenia usunięcia całej tabeli.

2.6.3. Raport z SQL Injection (advanced)

Ponieważ do pełnego rozwiązania listy (uzyskania 20 punktów) brakuje mi dokładnie jednego punktu, to wykonałem kilka wstępnych lekcji oraz test końcowy.

2.6.3.1. Zadanie 1 (lekcja 3)

Pierwszy sposób na rozwiązanie zadania (przemycenie drugiego zapytania). Uwaga – znaki '--' oznaczają rozpoczęcie komentarza, dzięki czemu mogę zakomentować niechciany apostrof i zakończyć wtrącone zapytanie wcześniej (nie trzeba szukać sposobu na zakończenie tak, aby apostrofy były sparowane).

✓

Name:

Password:

You have succeeded:

USERID	USER_NAME	PASSWORD	COOKIE
101	jsnow	passwd1	,
102	jdoue	passwd2	,
103	jplane	passwd3	,
104	jeff	jeff	,
105	dave	passW0rD	,

Well done! Can you also figure out a solution, by using a UNION?

Your query was: SELECT * FROM user_data WHERE last_name = 'cokolwiek'; SELECT * FROM user_system_data; --'

Drugi sposób na rozwiązanie zadania (wykorzystanie UNION):

✓

Name:

Password:

You have succeeded:

USERID, FIRST_NAME, LAST_NAME, CC_NUMBER, CC_TYPE, COOKIE, LOGIN_COUNT,

101, jsnow, passwd1passwd1, , passwd1, passwd1passwd1, 101,

102, jdoe, passwd2passwd2, , passwd2, passwd2passwd2, 102,

103, jplane, passwd3passwd3, , passwd3, passwd3passwd3, 103,

104, jeff, jeffjeff, , jeff, jeffjeff, 104,

105, dave, passW0rDpasswdW0rD, , passW0rD, passW0rDpasswdW0rD, 105,

Well done! Can you also figure out a solution, by appending a new SQL Statement?

Your query was: SELECT * FROM user_data WHERE last_name = '' UNION SELECT userid, user_name, password||password, cookie, password, password||password, userid FROM user_system_data;--'

Przy wykorzystywaniu UNION trzeba uważać by obie tabele miały te same typy wartości parami dla każdej kolumny kolejno.

Hasło Dave'a to:

✓

Name:

Password:

Congratulations. You have successfully completed the assignment.

2.6.3.2. Zadanie 2 (lekcja 5)

2.6.3.3. Quiz

Poniżej zamieszczam odpowiedzi udzielone w quizie. Quiz wymagał wiedzy z działu SQL Injection (mitigation), więc najpierw przejrzałem materiały z tamtego działu:

+ 1 2 3 4 5 6

Now it is time for a quiz! It is recommended to do all SQL injection lessons before trying the quiz. Answer all questions correctly to complete the assignment.

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 separation 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;--'.

Submit answers

Congratulations. You have successfully completed the assignment.