
	Powiatowy Zespół Szkół nr 2 im. Bohaterskiej Załogi ORP „Orzeł” w Wejherowie ul. Strzelecka 9, 84-200 Wejherowo			
033	Projektowanie i Administrowanie Bazami Danych	Klasa	1 2 3 4 5	<input type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/> <input type="checkbox"/> <input checked="" type="checkbox"/>
Temat	Wykorzystanie konstrukcji zapytań SQL w celu wywołania procedury składowanej (ang. function) na przykładzie SZBD MariaDB. Zaprojektuj procedurę składowaną o nazwie code, która na podstawie składowanych danych w tabeli address skopiuje 52 wiersze do tabeli test z użyciem symetrycznego szyfru blokowego wywoływanego za pomocą funkcji AES_ENCRYPT.			
Opis szczegółowych zagadnień wynikających z tematu lekcji.	MD5, AES_DECRYPT, LONG VARCHAR, VARCHAR, MEDIUMTEXT			

Wskazówka dla prowadzącego zajęcia dydaktyczne:

L.P.	Nazwa skryptu BASH	Pliki pomocnicze	funkcja
1.	maria.sh	plik	zakładanie użytkowników w SZBD MariaDB na podstawie pliku tekstowego o nazwie plik.
2.	cleaning.sh	clean.sh, plik	usuwanie kont systemowych w systemie operacyjnym Debian 11.
3.	drop.sh	cancell.sh, plik	usuwanie wszystkich tabel z bazy danych użytkownika wraz jego procedurami i funkcjami.
4.	student.sh	mysql.user	usuwanie wszystkich użytkowników z SZBD MariaDB, którzy spełniają kryterium 6 znaków alfanumerycznych w nazwie loginu.
5.	user.sh	plik, skrypt.sh	przygotowanie kont systemowych zgodnie z nazwami loginów z pliku tekstowego o nazwie plik.
6.	delete.sh	plik	usuwanie baz danych dla każdego użytkownika zgodnie z nazwami loginów z pliku tekstowego o nazwie plik.

Zadanie nr 1

01

Od roku szkolnego 2022/2023 w pracowni Projektowania i Administracji Bazami Danych po zakończeniu roku szkolnego z komputerów przeznaczonych dla uczniów prowadzący zajęcia dydaktyczne z przedmiotu PiABD usuwa wszystkie konta systemowe w systemie operacyjnym Debian 11 oraz wszystkie konta użytkowników w SZBD MariaDB wraz z bazami danych dla tych użytkowników. Dlatego na pierwszych zajęciach w nowym roku szkolnym w pierwszej kolejności prowadzący zajęcia przygotowuje wraz z uczniami strukturę pliku tekstowego o nazwie **PLIK** dla grupy. Nazwy loginów muszą składać się z **6 znaków alfanumerycznych**. Aby uniknąć kojarzenia numerów ID z numeracją poszczególnych uczniów w elektronicznym dzienniku lekcyjnym, przyjmujemy następującą zasadę, że na komputerze oznaczonym jako **EXAM01** dla użytkownika zostanie przydzielone ID **01** i tak dalej aż do numeru ostatniej stacji roboczej. Należy przy tym uwzględnić fakt, że klasa na zajęciach jest podzielona na dwie grupy, co wiąże się z tym, że nie każdy uczeń, któremu zostanie np. przydzielony komputer o nazwie **EXAM01** otrzyma ID takie same jak ID komputera. Jeśli zajdzie taka sytuacja, uczeń z drugiej grupy w ramach tej samej klasy, otrzyma pierwszy wolny numer ID.



Poniżej zamieszczono kolejność znaków alfanumerycznych dla pliku tekstowego o nazwie **PLIK**.

omega@debianEXAM70-2ftp70

Po przygotowaniu i sprawdzeniu pliku tekstowego **PLIK** prowadzący zajęcia z konsoli komputera pełniącego funkcję serwera o nazwie debianEXAM00 uruchamia skryptu BASH w następującej kolejności: **user.sh**, **maria.sh**.

Wskazówka:

Proszę pamiętać, że zarówno przed pierwszym logowaniem do systemu operacyjnego Debian 11 oraz do SZBD MariaDB, użytkownik będzie zmuszony do zmiany hasła na inne według własnego uznania. W tym celu zachowaj się zgodnie z przyjętą ogólną polityką bezpieczeństwa i uwzględnij nie mniej niż 8 znaków alfanumerycznych. Wykorzystaj przy tworzeniu hasła małe i wielkie litery alfabetu, cyfry od 0 do 9 oraz znaki specjalne.

Na czas pierwszego logowania do systemu operacyjnego Debian 11 oraz SZBD MariaDB przyjęto hasło: **zaq1@WSX**

- | | |
|---|-------------|
| A | nie dotyczy |
| B | nie dotyczy |
| C | nie dotyczy |
| D | nie dotyczy |

Zadanie nr 2

Wykonaj następujące polecenie:

show databases;

Po zalogowaniu swoim loginem dla Twojego użytkownika do SZBD MariaDB po dowolnej składni SQL, którą w ramach swoich przydzielonych uprawnień możesz wykonać, otrzymasz następujący komunikat:

ERROR 1820 (HY000): You must SET PASSWORD before executing this statement



- 02 Powyższy komunikat oznacza, że nie wykonasz żadnego **dozwolonego** dla Ciebie polecenia języka SQL, dopóki nie zmienisz sam dla siebie hasła. Przypomnij sobie wiadomości z poprzedniego roku szkolnego i zmień hasło w SZBD MariaDB dla samego siebie. Po udanej zmianie hasła wprowadź ponownie instrukcję języka SQL i powinieneś otrzymać podobny wynik jak poniżej, pamiętając, że nie będziesz widział nazwy bazy danych **2ftp70**, gdyż jest to tylko przykład.

show databases;

```
+-----+
| Database           |
+-----+
| 2ftp70             |
| information_schema |
+-----+
```

Tabela pomocnicza.

A	nie dotyczy
B	nie dotyczy
C	nie dotyczy
D	nie dotyczy

Poniższe instrukcje wyświetlają przydatne zmienne do zobaczenia wszystkich zmiennych systemowych dla zestawu znaków:

Zadanie nr 3

Wykonaj poniższe polecenie:

```
SHOW SESSION VARIABLES LIKE 'collation\_%';
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| collation_connection | utf8_general_ci |
| collation_database | utf8mb4_general_ci |
| collation_server | utf8mb4_general_ci |
+-----+-----+
```

```
SHOW SESSION VARIABLES LIKE 'character\_set\_%';
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| character_set_client | utf8 |
| character_set_connection | utf8 |
| character_set_database | utf8mb4 |
| character_set_filesystem | binary |
| character_set_results | utf8 |
| character_set_server | utf8mb4 |
| character_set_system | utf8 |
+-----+-----+
```

Zatrzymajmy się na chwilę przy zmiennej oznaczonej jako `character_set_results`, która wskazuje zestaw znaków w jakim serwer zwraca klientowi wynik zapytania. Obejmuje to dane wynikowe, takie jak wartości kolumn, metadane wyników, takie jak nazwy kolumn i komunikaty o błędach.



03

A SET character_set_results = NULL;

B SET character_set_results = binary;

C nie dotyczy

D nie dotyczy

Zapoznaj się z poniższym tekstem.

MySQL/MariaDB obsługuje szereg operacji szyfrowania i kodowania bezpośrednio z języka SQL, wykorzystuje między innymi standardowy w branży 128-bitowy algorytm **AES**, który jest uważany za silne szyfrowanie i spełnia wymagania przepisów. W MySQL/MariaDB funkcja **AES_ENCRYPT** służy do szyfrowania ciągu znaków przy użyciu algorytmu Advanced Encryption Standard (AES). Funkcja MySQL **AES_ENCRYPT** koduje dane przy użyciu klucza o długości 128 bitów, ale można ją rozszerzyć do długości klucza 256 bitów. Szyfruje ciąg i zwraca ciąg binarny. Funkcja **AES_ENCRYPT** przyjmuje dwa parametry, którymi są zaszyfrowany ciąg znaków i ciąg klucza używany do zaszyfrowania ciągu.



Zadanie nr 4

Niezależnie od tego, czy programujesz głównie w systemie Linux, czy Windows, możesz osiągnąć zgodną implementację szyfrowania z bazą danych MySQL/MariaDB. Przyjrzymy się na początek algorytmowi **MD5**. Funkcja szyfrowania **MD5** zwracając ciąg znaków ASCII. Ich wartością zwracaną jest ciąg znaków, którego zestaw znaków jest określony przez zmienne systemowe **CHARACTER_SET_CONNECTION** i **COLLATION_CONNECTION**. Jest to ciąg niebinarny.

Wykonaj poniższe polecenie:

```
SELECT MD5('testing');
```

```
+-----+
| MD5('testing') |
+-----+
| ae2b1fca515949e5d54fb22b8ed95575 |
+-----+
```

Tabela pomocnicza.

Wykonaj poniższe polecenie:

```
SELECT AES_ENCRYPT('testing', 'zaq1@WSX')\G;
```

Otrzymasz wynik zapytania SQL powodujący pewien nieporządek, bowiem algorytm szyfrowania **AES_ENCRYPT** zwraca wynik w postaci danych binarnych **BLOB** lub **BINARY**. Co możemy zrobić, żeby otrzymać wynik czytelniejszy dla nas. Możemy przekonwertować go na format szesnastkowy za pomocą dodatkowej klauzuli **HEX**. Ponieważ AES jest algorytmem blokowym, do kodowania ciągów o nierównej długości stosuje się dopełnianie. Konwersję reprezentacji szesnastkowej na binarną można uzyskać za pomocą klauzuli **UNHEX**.

Wykonaj poniższe polecenie:

```
SELECT HEX(AES_ENCRYPT('testing', 'zaq1@WSX')) as result;
```

```
+-----+
| result |
+-----+
| A8A1D60A722E37FB477817209BC0505D |
+-----+
```

Tabela pomocnicza.

04

A nie dotyczy

B nie dotyczy

C nie dotyczy

D nie dotyczy

Związku z tym, iż informacje związane z danymi adresowymi kontrahentów będą pilnie strzeżone przez firmę **NORTHPACK** od dnia, w którym rozwiązujesz to zadanie, przekazano Tobie do wiadomości, że należy **niezwłocznie** zaszyfrować wszystkie wiersze w tabeli **ADDRESS**.

Kopiowanie tabel w MySQL to rutynowa operacja wykonywana przez administratorów baz danych, programistów i analityków dziesiątki razy dziennie z różnych powodów i w różnych celach.

Skorzystamy ze schematu tabeli **ADDRESS** poprzez skopiowanie tego schematu do nowej tabeli o nazwie **TEST**.

```
describe address;
```

```
SHOW CREATE TABLE address\G;
```

Field	Type	Null	Key	Default	Extra
id	int(11)	NO		0	
company	varchar(100)	YES		NULL	
location	varchar(100)	YES		NULL	
country	varchar(100)	YES		NULL	

Tabela pomocnicza.



05

Zadanie nr 5

Wykonaj poniższe polecenie:

```
CREATE TABLE test AS SELECT * FROM address;
```

Sprawdź jaki uzyskałeś wynik wykonując poniższe polecenia:

```
select * from test;
```

Uzyskaliśmy dokładnie kopię tabeli address, ale jak pamiętasz nie tego od Ciebie oczekują. Zatem skasujemy tabeli test i spróbujemy podejść do problemu jeszcze raz.

Wykonaj poniższe polecenia:

```
DROP TABLE test;
```

```
CREATE TABLE test LIKE address;
```

```
describe test;
```

```
SELECT * FROM test;
```

Przy zastosowaniu powyższego polecenia SQL otrzymaliśmy kopię tylko samej struktury tabeli **ADDRESS**, bez zawartych danych w tabeli **TEST**.



A nie dotyczy

B nie dotyczy

C nie dotyczy

D nie dotyczy

Zadanie nr 6**Wykonaj poniższe polecenie:**

```
select * from address where id = 1\G;
```

```

id: 1
company: AGC Packaging System LTD
location: 8 Lordsbury Field, South Wallington, SM6 9PE Surrey
country: England

```

Wykonaj następujące polecenia:

```
INSERT INTO test (id, company, location, country) VALUES (null, 'AGC Packaging System LTD', '8 Lordsbury Field, South Wallington, SM6 9PE Surrey', 'England');
```

```
select * from test;
```

id	company	location	country
1	AGC Packaging System LTD	8 Lordsbury Field, South Wallington, SM6 9PE Surrey	England

Tabela pomocnicza.

```
DELETE FROM test;
```

06

Wykonaj następujące polecenia:

```

INSERT INTO test (id, company, location, country) VALUES (null,
HEX(AES_ENCRYPT("AGC Packaging System LTD","zaq1@WSX")), HEX(AES_ENCRYPT("8
Lordsbury Field, South Wallington, SM6 9PE Surrey","zaq1@WSX")),
HEX(AES_ENCRYPT("England","zaq1@WSX")));

```

ERROR 1406 (22001): Data too long for column 'location' at row 1

Otrzymał komunikat, który informuje Ciebie, że próbowałeś wprowadzić do tabeli **TEST** dane, które SZBD MariaDB nie może przyjąć, gdyż po zaszyfrowaniu zajmują więcej pamięci, niż przewiduje typ danych **VARCHAR(100)**. Wartości w kolumnach **VARCHAR** są ciągami o zmiennej długości. Długość można określić jako wartość od 0 do 65535 bajtów, który jest wspólny dla wszystkich kolumn i użytego zestawu znaków. Efektywna maksymalna długość **VARCHAR** zależy od maksymalnego rozmiaru wiersza.



Możesz to naprawić w dwojaki sposób, albo zwiększyć maksymalną liczbę przechowywanych znaków do wartości np. 200, albo zastosować typ danych **LONG VARCHAR**. **LONG** i **LONG VARCHAR** są synonimami **MEDIUMTEXT**. Kolumna **TEXT** o maksymalnej długości 16777215 znaków. Efektywna maksymalna długość jest mniejsza, jeśli wartość zawiera znaki wielobajtowe. Każda wartość **MEDIUMTEXT** jest przechowywana przy użyciu trzybajtowego przedrostka, który wskazuje liczbę bajtów w wartości. Kolumny **VARCHAR** mogą być w pełni indeksowane.

Kolumny **TEXT** można indeksować tylko na określonej długości.**A** nie dotyczy**B** nie dotyczy**C** nie dotyczy**D** nie dotyczy

Zadanie nr 7

Wykonaj jeden z zestawów zapytań SQL według własnego uznania:

```
alter table test modify column company VARCHAR(200);
alter table test modify column location VARCHAR(200);
alter table test modify column country VARCHAR(200);
```

```
alter table test modify column company LONG VARCHAR;
alter table test modify column location LONG VARCHAR;
alter table test modify column country LONG VARCHAR;
```

Wykonaj następujące zapytania SQL:

```
INSERT INTO test (id, company, location, country) VALUES (null,
HEX(AES_ENCRYPT("AGC Packaging System LTD", "zaql@WSX")), HEX(AES_ENCRYPT("8
Lordsbury Field, South Wallington, SM6 9PE Surrey", "zaql@WSX")),
HEX(AES_ENCRYPT("England", "zaql@WSX")));
```

```
select AES_DECRYPT(UNHEX(location), 'zaql@WSX') as result from test;
```

```
+-----+
| result |
+-----+
| 8 Lordsbury Field, South Wallington, SM6 9PE Surrey |
+-----+
```

Tabela pomocnicza.

Po sprawdzeniu, czy jesteśmy w stanie odszyfrować pojedyncze składowane dane w tabeli **TEST** możemy zastanowić się, jak zautomatyzować proces skopiowania wszystkich wierszy do tabeli **TEST** z tabeli **ADDRESS**, ale już **zaszyfrowane** z użyciem symetrycznego szyfru blokowego z wykorzystaniem funkcji **AES_ENCRYPT**. W naszym przypadku mamy 52 kontrahentów. Najlepszym rozwiązaniem automatyzacji zapytań SQL w przypadku tego konkretnego zadania będzie procedura.

```
select count(*) from address;
```

```
+-----+
| count(*) |
+-----+
|          52 |
+-----+
```

tabela pomocnicza.



07

- | | |
|----------|-------------|
| A | nie dotyczy |
| B | nie dotyczy |
| C | nie dotyczy |
| D | nie dotyczy |

Zadanie nr 8

Przygotuj samodzielnie procedurę o nazwie CODE.

08

01	SELECT COUNT(*) FROM address into n;
02	WHILE i <= n DO
03	END //
04	set @haslo := 'zaq1@WSX';
05	set @country := (select country from address where id = i);
06	DEALLOCATE PREPARE stmp;
07	set @location := (select location from address where id = i);
08	DELIMITER ;
09	set @company := (select company from address where id = i);
10	SET i=1;
11	END WHILE;
12	PREPARE stmp FROM @syntax;
13	SET i = i+1;
14	EXECUTE stmp;
15	DECLARE i INT;
16	DELIMITER //
17	BEGIN
18	DECLARE n INT;
19	set @syntax := CONCAT('INSERT INTO test (id, company, location, country) VALUES (null, HEX(AES_ENCRYPT('',@company,'','','@haslo,'')), HEX(AES_ENCRYPT('',@location,'','','@haslo,'')), HEX(AES_ENCRYPT('',@country,'','','@haslo,''))));
20	CREATE PROCEDURE code()

Tabela pomocnicza z propozycjami kodu dla procedury w SZBD MariaDB.

Wykonaj następujące zapytania SQL:

call code;

select * from test\G;

Po wykonaniu wszystkich zadań wykonaj export swojej bazy danych do pliku tekstowego o rozszerzeniu SQL.

Rozwiązanie

Jeśli Twoje zadanie było związane z zaprojektowaniem procedury lub funkcji a wykonujesz to zadanie za pomocą terminala tekstowego wykonaj odpowiednie polecenie wybierając jedną z odpowiedzi poniżej, jeśli tego nie zrobisz podczas eksportu do pliku **SQL** zostaną pominięte **procedury i funkcje**.



- | | |
|----------|---|
| A | <code>mysqldump --routines=true -u [konto] -p [vama] > vama.sql</code> |
| B | <code>alter table staff add column pesel int after iban;</code> |
| C | <code>LOAD DATA LOCAL INFILE '/home/omega/Dokumenty/pesel.txt' into table pesel;</code> |
| D | <code>alter table pesel add id int primary key auto_increment first;</code> |