

# Лабораторна робота 1

## СТВОРЕННЯ БАЗИ ДАНИХ ЗАСОБАМИ МОВИ SQL

**Мета роботи:** навчитися створювати та зв'язувати таблиці бази даних з використанням СУБД MySQL.

### Хід роботи

#### 1. Встановити MySQL

Для спрощення установки та подальшого використання системи управління базами даних (СУБД) MySQL рекомендується встановити один з вільно розповсюджуваних WAMP (Windows, Apache, MySQL, PHP) або LAMP (Linux, Apache, MySQL, PHP) серверів, наприклад OpenServer або XAMPP. В подальших прикладах виконання лабораторних робіт буде використовуватися сервер XAMPP.

Після того, як сервер XAMPP буде встановлено, необхідно запустити панель управління компонентами серверу, запустити MySQL та відкрити командний рядок сервера (рисунок 1.1).

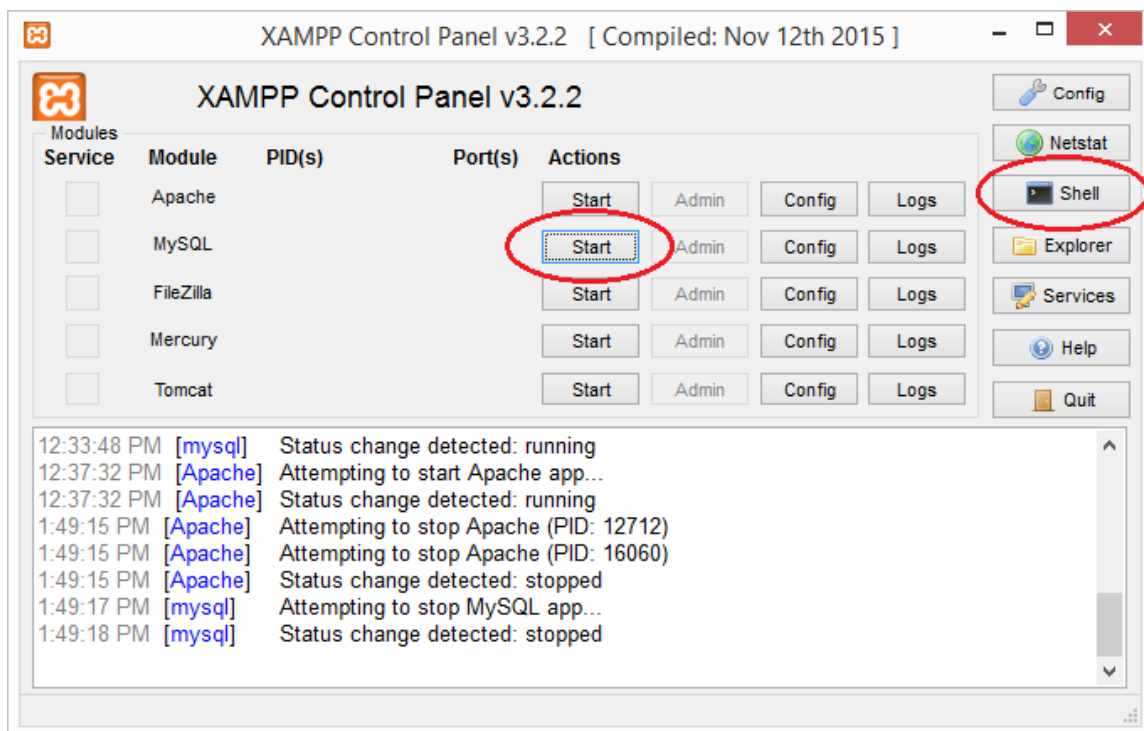


Рисунок 1.1

## 2. Підключитися до MySQL та створити базу даних

У командному рядку сервера MySQL необхідно ввести команду:

```
mysql -u root -p
```

Після цього необхідно ввести пароль (рисунок 1.2). Зазвичай пароль користувача root є порожнім, тому можна просто натиснути Enter.

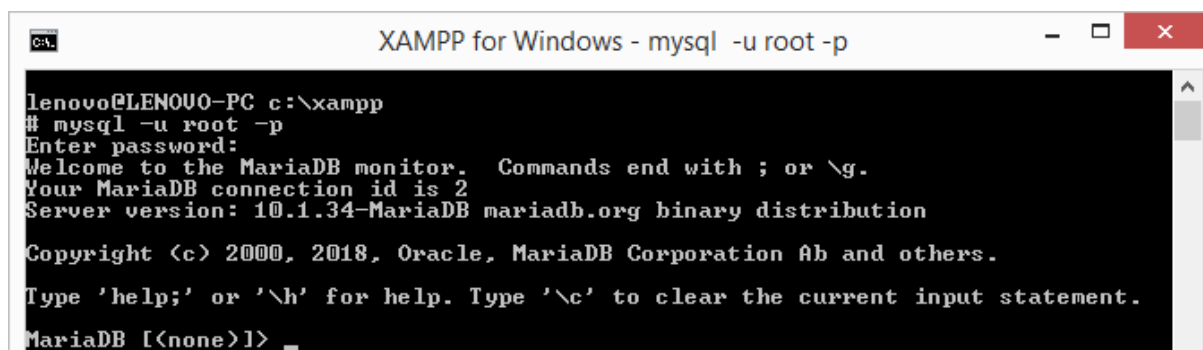


Рисунок 1.2

Для того, щоб відключитися від MySQL необхідно використати команду exit.

Основними командами, які будуть використовуватися час від часу при роботі з MySQL є наступні:

- 1) USE database – вибрати базу даних (БД) для подальшої роботи;
- 2) SHOW DATABASES – отримати перелік баз даних;
- 3) SHOW TABLES – отримати перелік таблиць для обраної БД;
- 4) SHOW COLUMNS FROM table – отримати інформацію про таблицю;
- 5) SHOW INDEX FROM table – отримати інформацію про індекси, визначені для таблиці.

Створення бази даних виконується за допомогою команди:

```
CREATE DATABASE supply;
```

Виконання даної команди дозволить створити базу даних, робота з якою буде розглядатися у лабораторному практикумі. Перевірити створення бази даних можна за допомогою команди SHOW DATABASES.

### 3. Створити таблиці бази даних та зв'язати їх

Для вивчення особливостей роботи з СУБД MySQL буде розглядатися база даних деякого підприємства, що придбає товари у різних постачальників. Придбання товарів виконується партіями та оформляється у вигляді договорів на поставку. Кожний договір має унікальний номер та укладається лише з одним постачальником. У документах по кожному договору вказується назва, розмір поставленої партії та ціна (у грн.).

Створення таблиць виконується за допомогою оператора CREATE TABLE. Таким чином, для бази даних, що розглядається, необхідно створити наступні таблиці:

```
CREATE TABLE supplier (  
  supplier_id int NOT NULL,  
  supplier_address varchar(100) NOT NULL,  
  supplier_phone varchar(20) NOT NULL,  
  PRIMARY KEY (supplier_id)  
) ENGINE=InnoDB;  
  
CREATE TABLE supplier_person (  
  supplier_id int NOT NULL,  
  supplier_last_name varchar(20) NOT NULL,  
  supplier_first_name varchar(20) NOT NULL,  
  supplier_middle_name varchar(20) NOT NULL,  
  PRIMARY KEY (supplier_id),  
  FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id)  
) ENGINE=InnoDB;  
  
CREATE TABLE supplier_org (  
  supplier_id int NOT NULL,  
  supplier_org_name varchar(20) NOT NULL,  
  PRIMARY KEY (supplier_id),  
  FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id)  
) ENGINE=InnoDB;  
  
CREATE TABLE contract (  
  contract_number int NOT NULL AUTO_INCREMENT,  
  contract_date timestamp NOT NULL,  
  supplier_id int NOT NULL,  
  contract_note varchar(100),  
  PRIMARY KEY (contract_number),  
  FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id)  
) ENGINE=InnoDB;
```

```
CREATE TABLE supplied (
contract_number int NOT NULL,
supplied_product varchar(20) NOT NULL,
supplied_amount decimal(4,0) NOT NULL,
supplied_cost decimal(8,2) NOT NULL,
PRIMARY KEY (contract_number, supplied_product),
FOREIGN KEY (contract_number) REFERENCES contract(contract_number)
) ENGINE=InnoDB;
```

Перевірити створені таблиці у базі даних supply (рисунок 1.3).

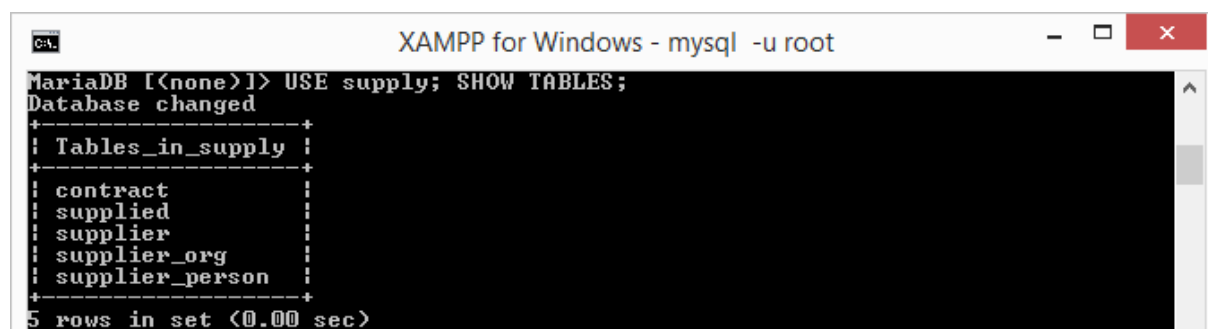
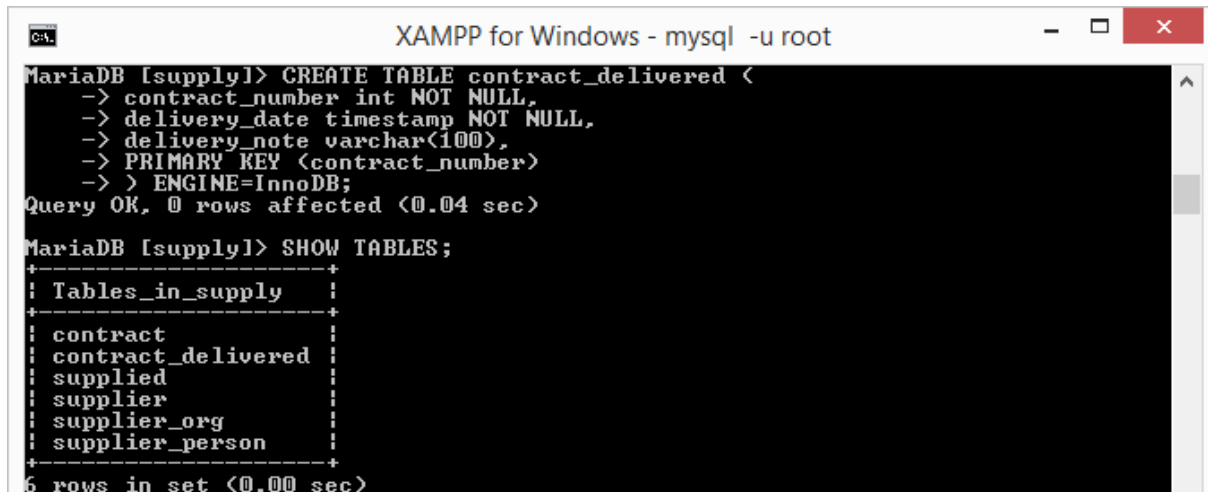


Рисунок 1.3

#### 4. Модифікація структури таблиць

Змінити структуру існуючої таблиці можна за допомогою оператора ALTER TABLE. Припустимо, що необхідно створити ще одну таблицю у базі даних supply, яка буде призначена для збереження даних про факти виконання поставок за договорами (рисунок 1.4).

```
CREATE TABLE contract_delivered (
contract_number int NOT NULL,
delivery_date timestamp NOT NULL,
delivery_note varchar(100),
PRIMARY KEY (contract_number)
) ENGINE=InnoDB;
```



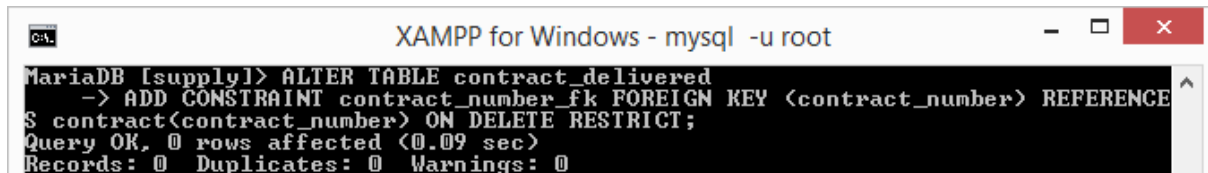
```
XAMPP for Windows - mysql -u root
MariaDB [supply]> CREATE TABLE contract_delivered (
-> contract_number int NOT NULL,
-> delivery_date timestamp NOT NULL,
-> delivery_note varchar(100),
-> PRIMARY KEY (contract_number)
-> ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.04 sec)

MariaDB [supply]> SHOW TABLES;
+-----+
| Tables_in_supply |
+-----+
| contract          |
| contract_delivered|
| supplied          |
| supplier          |
| supplier_org      |
| supplier_person   |
+-----+
6 rows in set (0.00 sec)
```

Рисунок 1.4

Для того, щоб зв'язати створену таблицю `contract_delivered` з таблицею `contract` необхідно застосувати команду `ALTER TABLE` (рисунок 1.5).

```
ALTER TABLE contract_delivered
ADD CONSTRAINT contract_number_fk FOREIGN KEY (contract_number)
REFERENCES contract(contract_number);
```

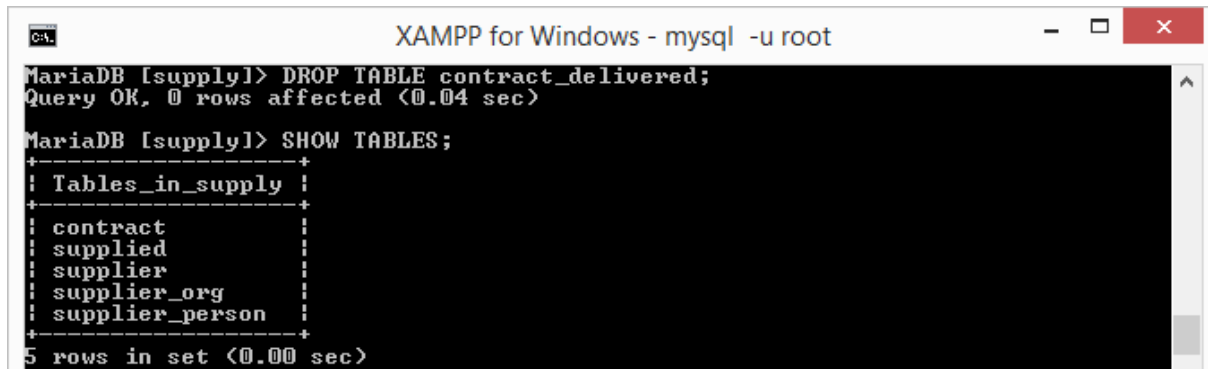


```
XAMPP for Windows - mysql -u root
MariaDB [supply]> ALTER TABLE contract_delivered
-> ADD CONSTRAINT contract_number_fk FOREIGN KEY (contract_number) REFERENCE
S contract(contract_number) ON DELETE RESTRICT;
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Рисунок 1.5

## 5. Видалення таблиць

Видалити таблицю можна за допомогою оператора `DROP TABLE`. Оскільки створена таблиця `contract_delivered` не буде використовуватися у подальшій роботі, її можна видалити за допомогою даної команди (рисунок 1.6).



```
CA. XAMPP for Windows - mysql -u root
MariaDB [supply]> DROP TABLE contract_delivered;
Query OK, 0 rows affected (0.04 sec)

MariaDB [supply]> SHOW TABLES;
+-----+
| Tables_in_supply |
+-----+
| contract          |
| supplied          |
| supplier          |
| supplier_org      |
| supplier_person   |
+-----+
5 rows in set (0.00 sec)
```

Рисунок 1.6

## 6. Оформити звіт з лабораторної роботи

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

## 7. Питання для самоконтролю

1. Як отримати доступ до командного рядка сервера MySQL?
2. Як встановити з'єднання з сервером MySQL, використовуючи ім'я та пароль певного користувача?
3. Назвіть основні команди адміністрування сервера БД MySQL та їх призначення.
4. За допомогою якої команди створюється база даних? Як можна перевірити створення БД?
5. За допомогою яких операторів мови SQL виконується створення та зв'язування таблиць?
6. За допомогою якого оператора мови SQL виконується модифікація структури таблиці?
7. За допомогою якого оператора мови SQL виконується видалення таблиці з бази даних?
8. Яким чином можна перевірити наявність або відсутність створених або видалених таблиць відповідно?
9. Яким чином можна задати ім'я зовнішнього ключа під час зв'язування таблиць?
10. Які недоліки містить розглянута структура БД? Як їх усунути?

## Лабораторна робота 2

### МАНІПУЛЮВАННЯ ДАНИМИ ЗАСОБАМИ МОВИ SQL: ДОДАВАННЯ, ООНОВЛЕННЯ ТА ВИДАЛЕННЯ ДАНИХ

**Мета роботи:** навчитися використовувати оператори мови SQL, призначені для додавання, оновлення та видалення даних, на прикладі СУБД MySQL.

#### Хід роботи

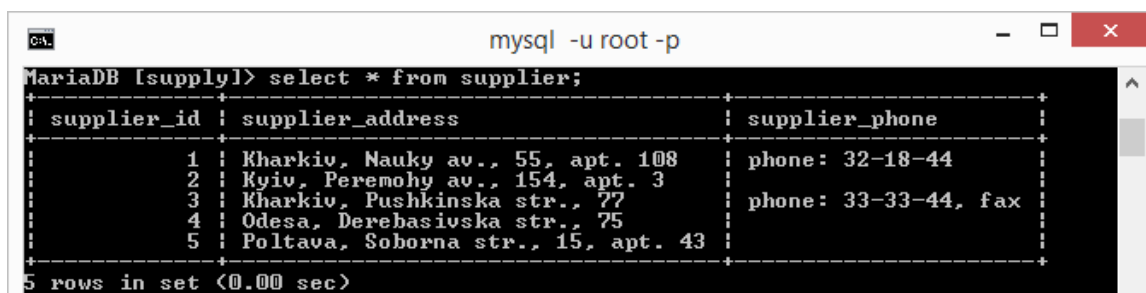
##### 1. Додавання даних до створеної бази даних

Для додавання даних використовується оператор INSERT.

Наступні команди дозволяють заповнити дані про постачальників у створеній базі даних:

```
INSERT INTO supplier (supplier_id, supplier_address, supplier_phone)
VALUES (1, 'Kharkiv, Nauky av., 55, apt. 108', 'phone: 32-18-44');
INSERT INTO supplier (supplier_id, supplier_address, supplier_phone)
VALUES (2, 'Kyiv, Peremohy av., 154, apt. 3', '');
INSERT INTO supplier (supplier_id, supplier_address, supplier_phone)
VALUES (3, 'Kharkiv, Pushkinska str., 77', 'phone: 33-33-44, fax: 22-12-33');
INSERT INTO supplier (supplier_id, supplier_address, supplier_phone)
VALUES (4, 'Odesa, Derebasivska str., 75', '');
INSERT INTO supplier (supplier_id, supplier_address, supplier_phone)
VALUES (5, 'Poltava, Soborna str., 15, apt. 43', '');
```

Перевірити записи, створені у таблиці supplier (рисунок 2.1).



The screenshot shows a terminal window titled 'mysql -u root -p'. The prompt is 'MariaDB [supply]>'. The command entered is 'select \* from supplier;'. The output is a table with 5 rows and 3 columns: 'supplier\_id', 'supplier\_address', and 'supplier\_phone'. The data is as follows:

supplier_id	supplier_address	supplier_phone
1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
2	Kyiv, Peremohy av., 154, apt. 3	
3	Kharkiv, Pushkinska str., 77	phone: 33-33-44, fax
4	Odesa, Derebasivska str., 75	
5	Poltava, Soborna str., 15, apt. 43	

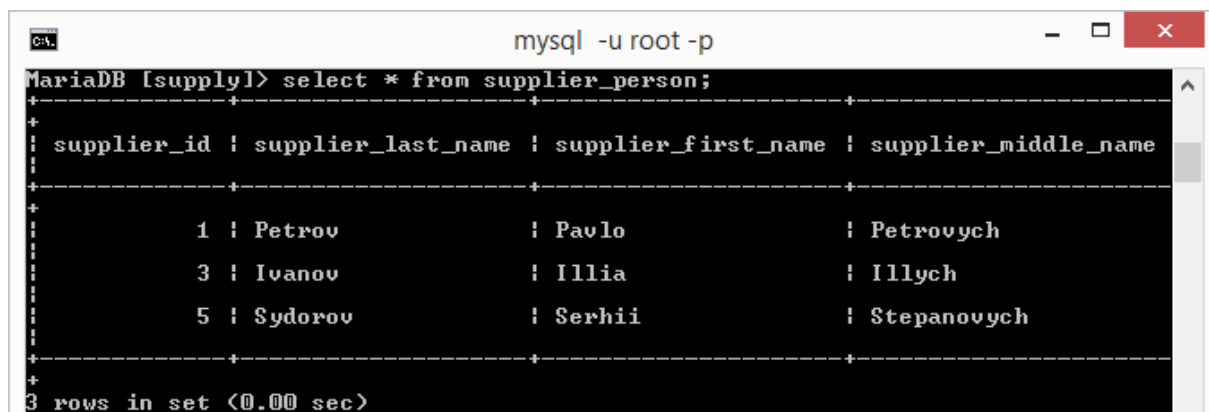
At the bottom of the terminal output, it says '5 rows in set (0.00 sec)'.

Рисунок 2.1

Наступні команди дозволяють заповнити дані про постачальників-фізичних осіб у створеній базі даних:

```
INSERT INTO supplier_person (supplier_id, supplier_last_name,  
supplier_first_name, supplier_middle_name) VALUES (1, 'Petrov', 'Pavlo',  
'Petrovych');  
INSERT INTO supplier_person (supplier_id, supplier_last_name,  
supplier_first_name, supplier_middle_name) VALUES (3, 'Ivanov', 'Illia',  
'Illych');  
INSERT INTO supplier_person (supplier_id, supplier_last_name,  
supplier_first_name, supplier_middle_name) VALUES (5, 'Sydorov', 'Serhii',  
'Stepanovych');
```

Перевірити записи, створені у таблиці `supplier_person` (рисунок 2.2).



The screenshot shows a MySQL terminal window titled 'mysql -u root -p'. The user is in the 'supply' database. The command 'select \* from supplier\_person;' has been executed, resulting in a table with 3 rows. The columns are 'supplier\_id', 'supplier\_last\_name', 'supplier\_first\_name', and 'supplier\_middle\_name'. The rows contain the following data: (1, 'Petrov', 'Pavlo', 'Petrovych'), (3, 'Ivanov', 'Illia', 'Illych'), and (5, 'Sydorov', 'Serhii', 'Stepanovych'). The terminal also shows '3 rows in set (0.00 sec)'.

supplier_id	supplier_last_name	supplier_first_name	supplier_middle_name
1	Petrov	Pavlo	Petrovych
3	Ivanov	Illia	Illych
5	Sydorov	Serhii	Stepanovych

Рисунок 2.2

Наступні команди дозволяють заповнити дані про постачальників-юридичних осіб у створеній базі даних:

```
INSERT INTO supplier_org (supplier_id, supplier_org_name) VALUES (2,  
'Interfruit Ltd.');
```

```
INSERT INTO supplier_org (supplier_id, supplier_org_name) VALUES (4,  
'Transservice LLC');
```

Перевірити записи, створені у таблиці `supplier_org` (рисунок 2.3).



```
mysql -u root -p
MariaDB [supply]> select * from supplier_org;
+-----+-----+
| supplier_id | supplier_org_name |
+-----+-----+
| 2 | Interfruit Ltd. |
| 4 | Transservice LLC |
+-----+-----+
2 rows in set <0.00 sec>
```

Рисунок 2.3

Наступні команди дозволяють заповнити дані про укладені договори у створеній базі даних:

```
INSERT INTO contract (contract_date, supplier_id, contract_note) VALUES
('2018-09-01', 1, 'Order 34 on 30.08.2018');
INSERT INTO contract (contract_date, supplier_id, contract_note) VALUES
('2018-09-10', 1, 'Invoice 08-78 on 28.08.2018');
INSERT INTO contract (contract_date, supplier_id, contract_note) VALUES
('2018-09-23', 3, 'Order 56 on 28.08.2018');
INSERT INTO contract (contract_date, supplier_id, contract_note) VALUES
('2018-09-24', 2, 'Order 74 on 11.09.2018');
INSERT INTO contract (contract_date, supplier_id, contract_note) VALUES
('2018-10-02', 2, 'Invoice 09-12 on 21.09.2018');
```

Перевірити записи, створені у таблиці contract (рисунок 2.4).

```
mysql -u root -p
MariaDB [supply]> select * from contract;
+-----+-----+-----+-----+
| contract_number | contract_date | supplier_id | contract_note |
+-----+-----+-----+-----+
| 1 | 2018-09-01 00:00:00 | 1 | Order 34 on 30.08.2018 |
| 2 | 2018-09-10 00:00:00 | 1 | Invoice 08-78 on 28.08.2018 |
| 3 | 2018-09-23 00:00:00 | 3 | Order 56 on 28.08.2018 |
| 4 | 2018-09-24 00:00:00 | 2 | Order 74 on 11.09.2018 |
| 5 | 2018-10-02 00:00:00 | 2 | Invoice 09-12 on 21.09.2018 |
+-----+-----+-----+-----+
5 rows in set <0.00 sec>
```

Рисунок 2.4

Наступні команди дозволяють заповнити дані про поставлені товари у створеній базі даних:

```
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (1, 'TV', 10, 1300);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (1, 'Audio Player', 25, 700);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (1, 'Video Player', 12, 750);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (2, 'Stereo System', 11, 500);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (2, 'Audio Player', 5, 450);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (2, 'Video Player', 8, 450);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (3, 'TV', 52, 900);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (3, 'Audio Player', 11, 550);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (3, 'Monitor', 85, 550);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (4, 'TV', 56, 990);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (4, 'Audio Player', 22, 320);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (4, 'Printer', 41, 350);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (5, 'TV', 14, 860);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (5, 'Audio Player', 33, 580);
INSERT INTO supplied (contract_number, supplied_product, supplied_amount,
supplied_cost) VALUES (5, 'Video Player', 17, 850);
```

Перевірити записи, створені у таблиці supplied (рисунок 2.5).

```

mysql -u root -p
MariaDB [supply]> select * from supplied;
+-----+-----+-----+-----+
| contract_number | supplied_product | supplied_amount | supplied_cost |
+-----+-----+-----+-----+
| 1 | Audio Player | 25 | 700.00 |
| 1 | TV | 10 | 1300.00 |
| 1 | Video Player | 12 | 750.00 |
| 2 | Audio Player | 5 | 450.00 |
| 2 | Stereo System | 11 | 500.00 |
| 2 | Video Player | 8 | 450.00 |
| 3 | Audio Player | 11 | 550.00 |
| 3 | Monitor | 85 | 550.00 |
| 3 | TV | 52 | 900.00 |
| 4 | Audio Player | 22 | 320.00 |
| 4 | Printer | 41 | 350.00 |
| 4 | TV | 56 | 990.00 |
| 5 | Audio Player | 33 | 580.00 |
| 5 | TV | 14 | 860.00 |
| 5 | Video Player | 17 | 850.00 |
+-----+-----+-----+-----+
15 rows in set (0.00 sec)

```

Рисунок 2.5

## 2. Оновлення даних у базі даних

Оновлення даних (зміна значень полів у існуючих записах) у БД виконується за допомогою оператора UPDATE.

Наприклад, якщо необхідно зменшити вартість принтеру який був поставлений за договором з номером 4, на 5%, команда буде наступною (рисунок 2.6):

```

UPDATE supplied
SET supplied_cost = supplied_cost * 0.95
WHERE contract_number = 4 AND supplied_product = 'Printer';

```

```

mysql -u root -p
MariaDB [supply]> update supplied set supplied_cost = supplied_cost * 0.95 where
contract_number = 4 AND supplied_product = 'Printer';
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [supply]> select * from supplied where contract_number = 4;
+-----+-----+-----+-----+
| contract_number | supplied_product | supplied_amount | supplied_cost |
+-----+-----+-----+-----+
| 4 | Audio Player | 22 | 320.00 |
| 4 | Printer | 41 | 332.50 |
| 4 | TV | 56 | 990.00 |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

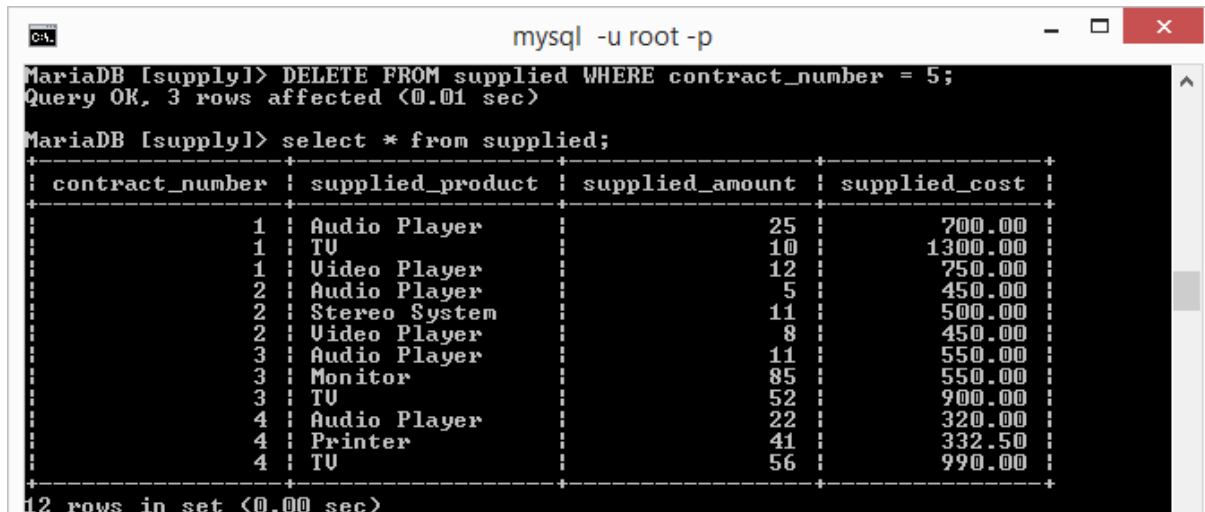
Рисунок 2.6

## 3. Видалення даних з бази даних

Для видалення даних з таблиць бази даних застосовується оператор DELETE.

Наприклад, для видалення поставлених товарів, які відповідають договору з номером 5, необхідно виконати наступну команду (Рисунок 2.7):

```
DELETE FROM supplied WHERE contract_number = 5;
```



The screenshot shows a MySQL terminal window titled 'mysql -u root -p'. The user is in the 'supply' database. The first command executed is 'DELETE FROM supplied WHERE contract\_number = 5;', which returns 'Query OK, 3 rows affected (0.01 sec)'. The second command is 'select \* from supplied;', which returns a table with 12 rows. The table has four columns: 'contract\_number', 'supplied\_product', 'supplied\_amount', and 'supplied\_cost'.

contract_number	supplied_product	supplied_amount	supplied_cost
1	Audio Player	25	700.00
1	TU	10	1300.00
1	Video Player	12	750.00
2	Audio Player	5	450.00
2	Stereo System	11	500.00
2	Video Player	8	450.00
3	Audio Player	11	550.00
3	Monitor	85	550.00
3	TU	52	900.00
4	Audio Player	22	320.00
4	Printer	41	332.50
4	TU	56	990.00

Рисунок 2.7

Відновіть видалені записи за допомогою команд INSERT.

#### 4. Оформити звіт з лабораторної роботи

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

#### 5. Питання для самоконтролю

1. Наведіть структуру та приклади використання команди INSERT.
2. Наведіть структуру та приклади використання команди UPDATE.
3. Наведіть структуру та приклади використання команди DELETE.
4. Яким чином можна оновити усі записи у таблиці БД?
5. Яким чином можна видалити усі записи з таблиці БД?
6. Яким чином можна видалити останні 20 укладених договорів?
7. Яким чином можна збільшити ціну на 15% для 5 найдешевших товарів, які були поставлені за певним договором?
8. Якою повинна бути структура команди INSERT для того, щоб нові записи з дублюючим ключем відкидалися без генерації помилки?

## Лабораторна робота 3

### МАНІПУЛЮВАННЯ ДАНИМИ ЗАСОБАМИ МОВИ SQL: ЗАПИТИ SELECT ТА ЇХ ОСНОВНІ ОСОБЛИВОСТІ

**Мета роботи:** навчитися використовувати оператор SELECT мови SQL, призначений для вибірки даних, на прикладі СУБД MySQL.

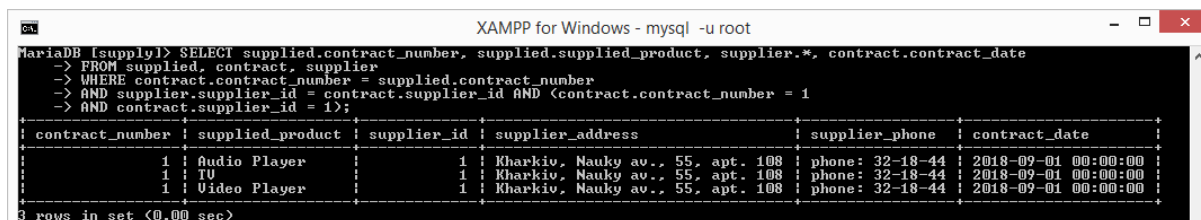
#### Хід роботи

##### 1. Створити та виконати запити SQL SELECT

###### Запит 1

Сформуувати список товарів, поставлених постачальником 1 (Петров П. П.) за договором 1 (рисунок 3.1).

```
SELECT supplied.contract_number, supplied.supplied_product, supplier.*, contract.contract_date
FROM supplied, contract, supplier
WHERE contract.contract_number = supplied.contract_number
AND supplier.supplier_id = contract.supplier_id AND (contract.contract_number = 1
AND contract.supplier_id = 1);
```



contract_number	supplied_product	supplier_id	supplier_address	supplier_phone	contract_date
1	Audio Player	1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44	2018-09-01 00:00:00
1	TU	1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44	2018-09-01 00:00:00
1	Video Player	1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44	2018-09-01 00:00:00

3 rows in set (0.00 sec)

Рисунок 3.1

###### Запит 2

Сформуувати список товарів, які були поставлені постачальником 1 (Петров П. П.) у період з 2018-09-05 по 2018-09-12 (рисунок 3.2).

```
SELECT contract.contract_number, contract.contract_date, supplied.supplied_product,
       supplied.supplied_cost, supplier.*
FROM (supplier INNER JOIN contract ON supplier.supplier_id = contract.supplier_id)
     INNER JOIN supplied ON contract.contract_number = supplied.contract_number
WHERE contract.contract_date BETWEEN '2018-09-05' AND '2018-09-12' AND
       supplier.supplier_id = 1;
```

```

XAMPP for Windows - mysql -u root
MariaDB [supply]> SELECT contract.contract_number, contract.contract_date, supplied.supplied_product,
-> supplied.supplied_cost, supplier.*
-> FROM (supplier INNER JOIN contract ON supplier.supplier_id = contract.supplier_id)
-> INNER JOIN supplied ON contract.contract_number = supplied.contract_number
-> WHERE contract.contract_date BETWEEN '2018-09-05' AND '2018-09-12' AND
-> supplier.supplier_id = 1;
+-----+-----+-----+-----+-----+-----+
| contract_number | contract_date | supplied_product | supplied_cost | supplier_id | supplier_address | supp
lier_phone |
+-----+-----+-----+-----+-----+-----+
| 32-18-44 | 2018-09-10 00:00:00 | Audio Player | 450.00 | 1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
| 32-18-44 | 2018-09-10 00:00:00 | Stereo System | 500.00 | 1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
| 32-18-44 | 2018-09-10 00:00:00 | Video Player | 450.00 | 1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)

```

Рисунок 3.2

### Запит 3

Сформувати список товарів, які були поставлені у 9 місяці 2018 року з виводом назви постачальника та дати поставки (рисунок 3.3).

```

SELECT contract.contract_number, contract.contract_date, supplied.supplied_product,
supplied.supplied_cost, supplier.*
FROM (supplier INNER JOIN contract ON supplier.supplier_id = contract.supplier_id) INNER JOIN
supplied ON contract.contract_number = supplied.contract_number
WHERE MONTH(contract.contract_date) = 9 AND YEAR(contract.contract_date) = 2018;

```

```

XAMPP for Windows - mysql -u root
MariaDB [supply]> SELECT contract.contract_number, contract.contract_date, supplied.supplied_product,
-> supplied.supplied_cost, supplier.*
-> FROM (supplier INNER JOIN contract ON supplier.supplier_id = contract.supplier_id) INNER JOIN
-> supplied ON contract.contract_number = supplied.contract_number
-> WHERE MONTH(contract.contract_date) = 9 AND YEAR(contract.contract_date) = 2018;
+-----+-----+-----+-----+-----+-----+
| contract_number | contract_date | supplied_product | supplied_cost | supplier_id | supplier_address | supp
lier_phone |
+-----+-----+-----+-----+-----+-----+
| 32-18-44 | 2018-09-01 00:00:00 | Audio Player | 700.00 | 1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
| 32-18-44 | 2018-09-01 00:00:00 | TV | 1300.00 | 1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
| 32-18-44 | 2018-09-01 00:00:00 | Video Player | 750.00 | 1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
| 32-18-44 | 2018-09-10 00:00:00 | Audio Player | 450.00 | 1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
| 32-18-44 | 2018-09-10 00:00:00 | Stereo System | 500.00 | 1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
| 32-18-44 | 2018-09-10 00:00:00 | Video Player | 450.00 | 1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
| 32-18-44 | 2018-09-24 00:00:00 | Audio Player | 320.00 | 2 | Kyiv, Peremohy av., 154, apt. 3 |
| 32-18-44 | 2018-09-24 00:00:00 | Printer | 332.50 | 2 | Kyiv, Peremohy av., 154, apt. 3 |
| 32-18-44 | 2018-09-24 00:00:00 | TV | 990.00 | 2 | Kyiv, Peremohy av., 154, apt. 3 |
| 32-18-44 | 2018-09-23 00:00:00 | Audio Player | 550.00 | 3 | Kharkiv, Pushkinska str., 77 | phon
e: 33-33-44, fax |
| 32-18-44 | 2018-09-23 00:00:00 | Monitor | 550.00 | 3 | Kharkiv, Pushkinska str., 77 | phon
e: 33-33-44, fax |
| 32-18-44 | 2018-09-23 00:00:00 | TV | 900.00 | 3 | Kharkiv, Pushkinska str., 77 | phon
e: 33-33-44, fax |
+-----+-----+-----+-----+-----+-----+
12 rows in set (0.00 sec)

```

Рисунок 3.3

### Запит 4

Сформувати список договорів (номер, дата, назва) та загальну суму за кожним договором (розмір партії помножити на ціну за одиницю та просумувати за договором). Список повинен бути відсортований за зростанням номерів договорів (рисунок 3.4).

```
SELECT contract.contract_number, contract.contract_date, contract.supplier_id,
       SUM(supplied.supplied_amount * supplied.supplied_cost) AS `Sum`
FROM contract INNER JOIN supplied ON contract.contract_number = supplied.contract_number
GROUP BY contract.contract_number, contract.contract_date, contract.supplier_id
ORDER BY contract.contract_number;
```

XAMPP for Windows - mysql -u root

```
MariaDB [supply]> SELECT contract.contract_number, contract.contract_date, contract.supplier_id,
-> SUM(supplied.supplied_amount * supplied.supplied_cost) AS `Sum`
-> FROM contract INNER JOIN supplied ON contract.contract_number = supplied.contract_number
-> GROUP BY contract.contract_number, contract.contract_date, contract.supplier_id
-> ORDER BY contract.contract_number;
```

contract_number	contract_date	supplier_id	Sum
1	2018-09-01 00:00:00	1	39500.00
2	2018-09-10 00:00:00	1	11350.00
3	2018-09-23 00:00:00	3	99600.00
4	2018-09-24 00:00:00	2	76112.50
5	2018-10-02 00:00:00	2	45630.00

5 rows in set (0.00 sec)

Рисунок 3.4

### Запит 5

Сформувати список договорів (номер, дата, назва) та загальну суму за кожним договором (розмір партії помножити на ціну за одиницю та просумувати за договором). Список повинен бути відсортований за зростанням загальних сум за кожним договором. Після цього на список повинна бути накладена умова фільтрації, яка полягає у виключенні з результату запиту записів, для яких номер договору менший 4 (рисунок 3.5).

```
SELECT contract.contract_number, contract.contract_date, contract.supplier_id,
       SUM(supplied.supplied_amount * supplied.supplied_cost) AS `Sum`
FROM contract INNER JOIN supplied ON contract.contract_number = supplied.contract_number
WHERE contract.contract_number < 4
GROUP BY contract.contract_number, contract.contract_date, contract.supplier_id
ORDER BY contract.contract_number;
```

XAMPP for Windows - mysql -u root

```
MariaDB [supply]> SELECT contract.contract_number, contract.contract_date, contract.supplier_id,
-> SUM(supplied.supplied_amount * supplied.supplied_cost) AS `Sum`
-> FROM contract INNER JOIN supplied ON contract.contract_number = supplied.contract_number
-> WHERE contract.contract_number < 4
-> GROUP BY contract.contract_number, contract.contract_date, contract.supplier_id
-> ORDER BY contract.contract_number;
```

contract_number	contract_date	supplier_id	Sum
1	2018-09-01 00:00:00	1	39500.00
2	2018-09-10 00:00:00	1	11350.00
3	2018-09-23 00:00:00	3	99600.00

3 rows in set (0.00 sec)

Рисунок 3.5



### Запит 6

Вивести інформацію про найбільшу за розміром партію товару в усіх договорах із зазначенням постачальника, а також номер та дати договору (рисунок 3.6).

```
SELECT contract.contract_number, contract.contract_date, contract.contract_note,  
       supplier.*, supplied.supplied_amount  
FROM contract, supplied, supplier  
WHERE contract.contract_number = supplied.contract_number AND  
       contract.supplier_id = supplier.supplier_id AND  
       supplied.supplied_amount = (SELECT MAX(supplied.supplied_amount) FROM supplied);
```

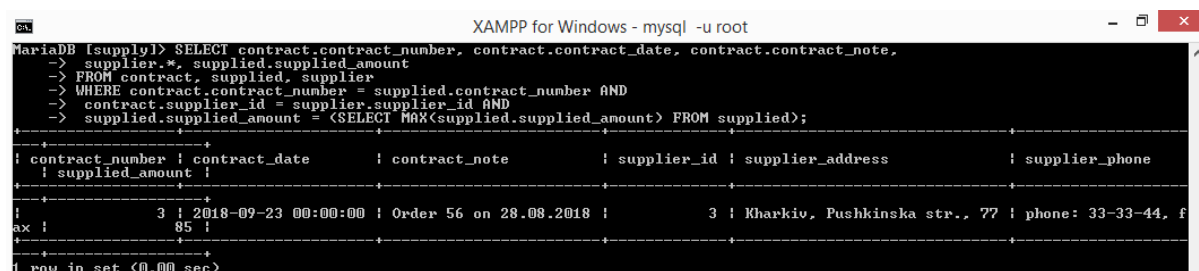


Рисунок 3.6

### Запит 7

Сформувати список постачальників (назва та код), з якими не було укладено жодного договору (рисунок 3.7).

```
SELECT * FROM supplier  
WHERE supplier_id NOT IN (SELECT supplier_id FROM supplier);
```



Рисунок 3.7

### Запит 8

Сформувати список назв поставлених товарів із зазначенням середньої ціни поставки за одиницю (незалежно від постачальника) (рисунок 3.8).

```
SELECT supplied_product, AVG(supplied_cost) AS `Average cost`  
FROM supplied  
GROUP BY supplied_product;
```



```

XAMPP for Windows - mysql -u root
MariaDB [supply]> SELECT supplied_product, AVG(supplied_cost) AS 'Average cost'
-> FROM supplied
-> GROUP BY supplied_product;
+-----+-----+
| supplied_product | Average cost |
+-----+-----+
| Audio Player    | 520.000000   |
| Monitor         | 550.000000   |
| Printer         | 332.500000   |
| Stereo System   | 500.000000   |
| TV              | 1012.500000  |
| Video Player    | 683.333333   |
+-----+-----+
6 rows in set (0.00 sec)

```

Рисунок 3.8

### Запит 9

Сформувати список товарів (назва, кількість та ціна, постачальник), для яких ціна за одиницю більше середньої (рисунок 3.9).

```

SELECT supplied_product, supplied_amount, supplied_cost, supplier.*
FROM (supplier INNER JOIN contract ON supplier.supplier_id = contract.supplier_id)
     INNER JOIN supplied ON contract.contract_number = supplied.contract_number
WHERE supplied_cost > (SELECT AVG(supplied_cost) FROM supplied);

```

```

XAMPP for Windows - mysql -u root
MariaDB [supply]> SELECT supplied_product, supplied_amount, supplied_cost, supplier.*
-> FROM (supplier INNER JOIN contract ON supplier.supplier_id = contract.supplier_id)
-> INNER JOIN supplied ON contract.contract_number = supplied.contract_number
-> WHERE supplied_cost > (SELECT AVG(supplied_cost) FROM supplied);
+-----+-----+-----+-----+-----+-----+
| supplied_product | supplied_amount | supplied_cost | supplier_id | supplier_address | supplier_phone |
+-----+-----+-----+-----+-----+-----+
| Audio Player    | 25              | 700.00       | 1           | Kharkiv, Nauky av., 55, apt. 108 | phone: 32-18-44 |
| TV              | 10              | 1300.00      | 1           | Kharkiv, Nauky av., 55, apt. 108 | phone: 32-18-44 |
| Video Player    | 12              | 750.00       | 1           | Kharkiv, Nauky av., 55, apt. 108 | phone: 32-18-44 |
| TV              | 56              | 990.00       | 2           | Kyiv, Peremohy av., 154, apt. 3   |                 |
| TV              | 14              | 860.00       | 2           | Kyiv, Peremohy av., 154, apt. 3   |                 |
| Video Player    | 17              | 850.00       | 2           | Kyiv, Peremohy av., 154, apt. 3   |                 |
| TV              | 52              | 900.00       | 3           | Kharkiv, Pushkinska str., 77     | phone: 33-33-44, fax |
+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

Рисунок 3.9

### Запит 10

Вивести інформацію про п'ять найдорожчих товарів (назва, ціна за одиницю, постачальник) (рисунок 3.10).

```

SELECT supplied_product, supplied_cost, supplier.*
FROM (supplier INNER JOIN contract ON supplier.supplier_id = contract.supplier_id)
     INNER JOIN supplied ON contract.contract_number = supplied.contract_number
ORDER BY supplied_cost DESC
LIMIT 5;

```

```

XAMPP for Windows - mysql -u root
MariaDB [supply]> SELECT supplied_product, supplied_cost, supplier.*
-> FROM (supplier INNER JOIN contract ON supplier.supplier_id = contract.supplier_id)
-> INNER JOIN supplied ON contract.contract_number = supplied.contract_number
-> ORDER BY supplied_cost DESC
-> LIMIT 1;
+-----+-----+-----+-----+-----+
| supplied_product | supplied_cost | supplier_id | supplier_address | supplier_phone |
+-----+-----+-----+-----+-----+
| TV               | 1300.00      | 1          | Kharkiv, Nauky av., 55, apt. 108 | phone: 32-18-44 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Рисунок 3.10

### Запит 11

Сформувати список постачальників із зазначенням коду, адреси та даних постачальника. При формуванні даних постачальника для фізичних осіб вивести прізвище та ініціали, а для юридичних осіб – назву (рисунок 3.11).

```

SELECT supplier.supplier_id, supplier.supplier_address,
       IFNULL(supplier_org.supplier_org_name, CONCAT(RTRIM(supplier_person.supplier_last_name), ' ',
       SUBSTRING(supplier_person.supplier_first_name, 1, 1), '. ',
       SUBSTRING(supplier_person.supplier_middle_name, 1, 1), '. ')) AS `Supplier`
FROM (supplier LEFT JOIN supplier_person ON supplier.supplier_id = supplier_person.supplier_id)
LEFT JOIN supplier_org ON supplier.supplier_id = supplier_org.supplier_id;

```

```

XAMPP for Windows - mysql -u root
MariaDB [supply]> SELECT supplier.supplier_id, supplier.supplier_address,
-> IFNULL(supplier_org.supplier_org_name, CONCAT(RTRIM(supplier_person.supplier_last_name), ' ',
-> SUBSTRING(supplier_person.supplier_first_name, 1, 1), '. ',
-> SUBSTRING(supplier_person.supplier_middle_name, 1, 1), '. ')) AS `Supplier`
-> FROM (supplier LEFT JOIN supplier_person ON supplier.supplier_id = supplier_person.supplier_id)
-> LEFT JOIN supplier_org ON supplier.supplier_id = supplier_org.supplier_id;
+-----+-----+-----+
| supplier_id | supplier_address | Supplier |
+-----+-----+-----+
| 1           | Kharkiv, Nauky av., 55, apt. 108 | Petrov P. P. |
| 2           | Kyiv, Peremohy av., 154, apt. 3   | Interfruit Ltd. |
| 3           | Kharkiv, Pushkinska str., 77     | Ivanov I. I. |
| 4           | Odesa, Derebasivska str., 75     | Transservice LLC |
| 5           | Poltava, Soborna str., 15, apt. 43 | Sydorov S. S. |
+-----+-----+-----+
5 rows in set (0.00 sec)

```

Рисунок 3.11

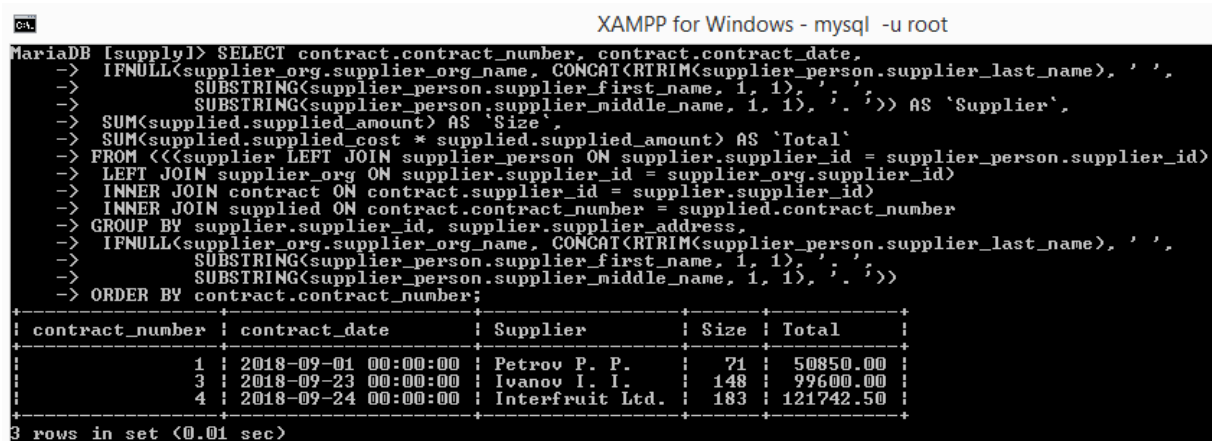
### Запит 12

Сформувати список договорів (із зазначенням номеру, дати постачання та даних про постачальника), загальну кількість поставлених товарів та загальну суму за кожним договором. При формуванні даних постачальника для фізичних осіб вивести прізвище та ініціали, а для юридичних осіб – назву. В результат повинні бути включені тільки ті договори, на основі яких товари дійсно поставлялись (тобто в результат запиту не повинні потрапити так звані «пусті» договори) (рисунок 3.12).

```

SELECT contract.contract_number, contract.contract_date,
    IFNULL(supplier_org.supplier_org_name, CONCAT(RTRIM(supplier_person.supplier_last_name), ' ',
        SUBSTRING(supplier_person.supplier_first_name, 1, 1), '. ',
        SUBSTRING(supplier_person.supplier_middle_name, 1, 1), '. ')) AS `Supplier`,
    SUM(supplied.supplied_amount) AS `Size`,
    SUM(supplied.supplied_cost * supplied.supplied_amount) AS `Total`
FROM (((supplier LEFT JOIN supplier_person ON supplier.supplier_id = supplier_person.supplier_id)
LEFT JOIN supplier_org ON supplier.supplier_id = supplier_org.supplier_id)
INNER JOIN contract ON contract.supplier_id = supplier.supplier_id)
INNER JOIN supplied ON contract.contract_number = supplied.contract_number
GROUP BY supplier.supplier_id, supplier.supplier_address,
    IFNULL(supplier_org.supplier_org_name, CONCAT(RTRIM(supplier_person.supplier_last_name), ' ',
        SUBSTRING(supplier_person.supplier_first_name, 1, 1), '. ',
        SUBSTRING(supplier_person.supplier_middle_name, 1, 1), '. '))
ORDER BY contract.contract_number;

```



```

XAMPP for Windows - mysql -u root
MariaDB [supply]> SELECT contract.contract_number, contract.contract_date,
-> IFNULL(supplier_org.supplier_org_name, CONCAT(RTRIM(supplier_person.supplier_last_name), ' ',
-> SUBSTRING(supplier_person.supplier_first_name, 1, 1), '. ',
-> SUBSTRING(supplier_person.supplier_middle_name, 1, 1), '. ')) AS `Supplier`,
-> SUM(supplied.supplied_amount) AS `Size`,
-> SUM(supplied.supplied_cost * supplied.supplied_amount) AS `Total`
-> FROM (((supplier LEFT JOIN supplier_person ON supplier.supplier_id = supplier_person.supplier_id)
-> LEFT JOIN supplier_org ON supplier.supplier_id = supplier_org.supplier_id)
-> INNER JOIN contract ON contract.supplier_id = supplier.supplier_id)
-> INNER JOIN supplied ON contract.contract_number = supplied.contract_number
-> GROUP BY supplier.supplier_id, supplier.supplier_address,
-> IFNULL(supplier_org.supplier_org_name, CONCAT(RTRIM(supplier_person.supplier_last_name), ' ',
-> SUBSTRING(supplier_person.supplier_first_name, 1, 1), '. ',
-> SUBSTRING(supplier_person.supplier_middle_name, 1, 1), '. '))
-> ORDER BY contract.contract_number;
+-----+-----+-----+-----+-----+
| contract_number | contract_date | Supplier | Size | Total |
+-----+-----+-----+-----+-----+
| 1 | 2018-09-01 00:00:00 | Petrov P. P. | 71 | 50850.00 |
| 3 | 2018-09-23 00:00:00 | Ivanov I. I. | 148 | 99600.00 |
| 4 | 2018-09-24 00:00:00 | Interfruit Ltd. | 183 | 121742.50 |
+-----+-----+-----+-----+-----+
3 rows in set (0.01 sec)

```

Рисунок 3.12

### Запит 13

Сформувати список товарів (із зазначенням номеру договору та дати постачання), поставлених постачальниками 1 (Петров П. П.) та 2 («Інтерфрут») (рисунок 3.13).

```

SELECT supplied.contract_number, contract.contract_date,
    supplied.supplied_product, supplier.supplier_id
FROM supplied, contract, supplier
WHERE contract.contract_number = supplied.contract_number
    AND supplier.supplier_id = contract.supplier_id AND contract.supplier_id = 1
UNION
SELECT supplied.contract_number, contract.contract_date,
    supplied.supplied_product, supplier.supplier_id
FROM supplied, contract, supplier
WHERE contract.contract_number = supplied.contract_number
    AND supplier.supplier_id = contract.supplier_id AND contract.supplier_id = 2
ORDER BY supplier_id, contract_number;

```

```

XAMPP for Windows - mysql -u root
MariaDB [supply]> SELECT supplied.contract_number, contract.contract_date,
-> supplied.supplied_product, supplier.supplier_id
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number
-> AND supplier.supplier_id = contract.supplier_id AND contract.supplier_id = 1
-> UNION
-> SELECT supplied.contract_number, contract.contract_date,
-> supplied.supplied_product, supplier.supplier_id
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number
-> AND supplier.supplier_id = contract.supplier_id AND contract.supplier_id = 2
-> ORDER BY supplier_id, contract_number;

```

contract_number	contract_date	supplied_product	supplier_id
1	2018-09-01 00:00:00	Audio Player	1
1	2018-09-01 00:00:00	TU	1
1	2018-09-01 00:00:00	Video Player	1
2	2018-09-10 00:00:00	Audio Player	1
2	2018-09-10 00:00:00	Stereo System	1
2	2018-09-10 00:00:00	Video Player	1
4	2018-09-24 00:00:00	Printer	2
4	2018-09-24 00:00:00	TU	2
4	2018-09-24 00:00:00	Audio Player	2
5	2018-10-02 00:00:00	Audio Player	2
5	2018-10-02 00:00:00	TU	2
5	2018-10-02 00:00:00	Video Player	2

```

12 rows in set (0.00 sec)

```

Рисунок 3.13

#### Запит 14

Сформувати номенклатуру товарів (тобто список назв товарів), які поставлялись тільки постачальником 1 (Петров П. П.), або тільки постачальником 2 («Інтерфрут»), або і постачальником 1, і постачальником 2 (рисунок 3.14).

```

SELECT DISTINCT supplied.supplied_product
FROM supplied, contract
WHERE contract.contract_number = supplied.contract_number AND contract.supplier_id = 1
UNION
SELECT DISTINCT supplied.supplied_product
FROM supplied, contract
WHERE contract.contract_number = supplied.contract_number AND contract.supplier_id = 2
ORDER BY supplied_product;

```

```

XAMPP for Windows - mysql -u root
MariaDB [supply]> SELECT DISTINCT supplied.supplied_product
-> FROM supplied, contract
-> WHERE contract.contract_number = supplied.contract_number AND contract.supplier_id = 1
-> UNION
-> SELECT DISTINCT supplied.supplied_product
-> FROM supplied, contract
-> WHERE contract.contract_number = supplied.contract_number AND contract.supplier_id = 2
-> ORDER BY supplied_product;

```

supplied_product
Audio Player
Printer
Stereo System
TU
Video Player

```

5 rows in set (0.00 sec)

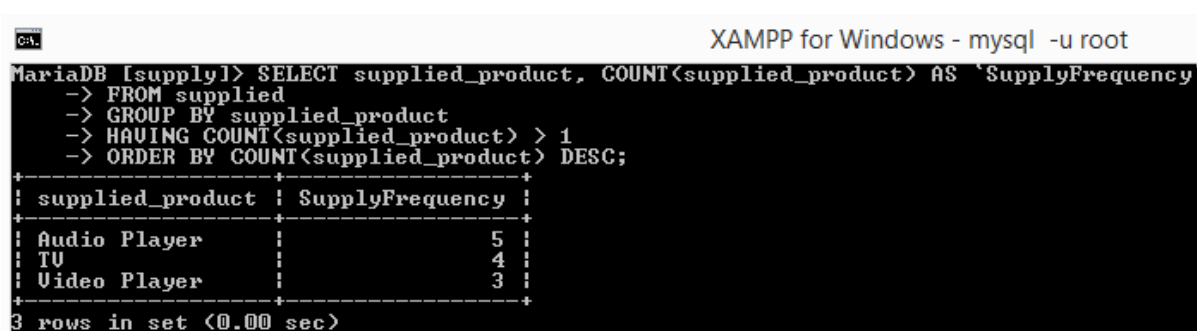
```

Рисунок 3.14

### Запит 15

Сформувати список товарів, який повинен відображати частоту поставок. У список включити тільки товари, які поставлялись більше одного разу. Список повинен бути відсортований за зменшенням частоти поставок (рисунок 3.15).

```
SELECT supplied_product, COUNT(supplied_product) AS `SupplyFrequency`
FROM supplied
GROUP BY supplied_product
HAVING COUNT(supplied_product) > 1
ORDER BY COUNT(supplied_product) DESC;
```



The screenshot shows a terminal window titled "XAMPP for Windows - mysql -u root". The prompt is "MariaDB [supply]>". The user has entered the following SQL query:

```
SELECT supplied_product, COUNT(supplied_product) AS `SupplyFrequency`
FROM supplied
GROUP BY supplied_product
HAVING COUNT(supplied_product) > 1
ORDER BY COUNT(supplied_product) DESC;
```

The output of the query is displayed in a table format:

supplied_product	SupplyFrequency
Audio Player	5
TU	4
Video Player	3

At the bottom of the terminal, it says "3 rows in set (0.00 sec)".

Рисунок 3.15

### Запит 16

Сформувати дані про кількісну динаміку поставок товарів протягом 2018 року. Дані повинні бути агреговані за місяцями та представлені у вигляді таблиці, рядками якої є назви товарів, а стовпчиками – номери місяців 2018 року. На перетині рядка і стовпчика повинна відображатися кількість даного товару, поставленого у даному місяці (рисунок 3.16).

```
SELECT supplied_product, SUM(IF(MONTH(contract_date) = 1, supplied_amount, 0)) AS `Jan`,
SUM(IF(MONTH(contract_date) = 2, supplied_amount, 0)) AS `Feb`,
SUM(IF(MONTH(contract_date) = 3, supplied_amount, 0)) AS `Mar`,
SUM(IF(MONTH(contract_date) = 4, supplied_amount, 0)) AS `Apr`,
SUM(IF(MONTH(contract_date) = 5, supplied_amount, 0)) AS `May`,
SUM(IF(MONTH(contract_date) = 6, supplied_amount, 0)) AS `Jun`,
SUM(IF(MONTH(contract_date) = 7, supplied_amount, 0)) AS `Jul`,
SUM(IF(MONTH(contract_date) = 8, supplied_amount, 0)) AS `Aug`,
SUM(IF(MONTH(contract_date) = 9, supplied_amount, 0)) AS `Sep`,
SUM(IF(MONTH(contract_date) = 10, supplied_amount, 0)) AS `Oct`,
SUM(IF(MONTH(contract_date) = 11, supplied_amount, 0)) AS `Nov`,
SUM(IF(MONTH(contract_date) = 12, supplied_amount, 0)) AS `Dec`
FROM contract, supplied
WHERE contract.contract_number = supplied.contract_number AND YEAR(contract_date) = 2018
GROUP BY supplied_product
ORDER BY supplied_product;
```

XAMPP for Windows - mysql -u root

```

MariaDB [supply]> SELECT supplied_product, SUM(IF(MONTH(contract_date) = 1, supplied_amount, 0)) AS `Jan`,
-> SUM(IF(MONTH(contract_date) = 2, supplied_amount, 0)) AS `Feb`,
-> SUM(IF(MONTH(contract_date) = 3, supplied_amount, 0)) AS `Mar`,
-> SUM(IF(MONTH(contract_date) = 4, supplied_amount, 0)) AS `Apr`,
-> SUM(IF(MONTH(contract_date) = 5, supplied_amount, 0)) AS `May`,
-> SUM(IF(MONTH(contract_date) = 6, supplied_amount, 0)) AS `Jun`,
-> SUM(IF(MONTH(contract_date) = 7, supplied_amount, 0)) AS `Jul`,
-> SUM(IF(MONTH(contract_date) = 8, supplied_amount, 0)) AS `Aug`,
-> SUM(IF(MONTH(contract_date) = 9, supplied_amount, 0)) AS `Sep`,
-> SUM(IF(MONTH(contract_date) = 10, supplied_amount, 0)) AS `Oct`,
-> SUM(IF(MONTH(contract_date) = 11, supplied_amount, 0)) AS `Nov`,
-> SUM(IF(MONTH(contract_date) = 12, supplied_amount, 0)) AS `Dec`
-> FROM contract, supplied
-> WHERE contract.contract_number = supplied.contract_number AND YEAR(contract_date) = 2018
-> GROUP BY supplied_product
-> ORDER BY supplied_product;

```

supplied_product	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Audio Player	0	0	0	0	0	0	0	0	63	33	0	0
Monitor	0	0	0	0	0	0	0	0	85	0	0	0
Printer	0	0	0	0	0	0	0	0	41	0	0	0
Stereo System	0	0	0	0	0	0	0	0	11	0	0	0
TU	0	0	0	0	0	0	0	0	118	14	0	0
Video Player	0	0	0	0	0	0	0	0	20	17	0	0

6 rows in set (0.00 sec)

Рисунок 3.16

### Запит 17

Сформувати список поставлених товарів. Для кожного товару у цьому списку повинні бути вказані наступні дані: номер договору, назва товару, кількість одиниць, ціна за одиницю, дата поставки, назва місяця та номер року (рисунок 3.17).

```

SELECT supplied.contract_number, supplied.supplied_product,
       supplied.supplied_amount, supplied.supplied_cost,
       contract.contract_date,
       MONTHNAME(contract.contract_date) AS `Month`,
       YEAR(contract.contract_date) AS `Year`
FROM supplied, contract
WHERE contract.contract_number = supplied.contract_number;

```

XAMPP for Windows - mysql -u root

```

MariaDB [supply]> SELECT supplied.contract_number, supplied.supplied_product,
-> supplied.supplied_amount, supplied.supplied_cost,
-> contract.contract_date,
-> MONTHNAME(contract.contract_date) AS `Month`,
-> YEAR(contract.contract_date) AS `Year`
-> FROM supplied, contract
-> WHERE contract.contract_number = supplied.contract_number;

```

contract_number	supplied_product	supplied_amount	supplied_cost	contract_date	Month	Year
1	Audio Player	25	700.00	2018-09-01 00:00:00	September	2018
1	TU	10	1300.00	2018-09-01 00:00:00	September	2018
1	Video Player	12	750.00	2018-09-01 00:00:00	September	2018
2N	Audio Player	5	450.00	2018-09-10 00:00:00	September	2018
2N	Stereo System	11	500.00	2018-09-10 00:00:00	September	2018
2N	Video Player	8	450.00	2018-09-10 00:00:00	September	2018
3	Audio Player	11	550.00	2018-09-23 00:00:00	September	2018
3	Monitor	85	550.00	2018-09-23 00:00:00	September	2018
3	TU	52	900.00	2018-09-23 00:00:00	September	2018
4	Audio Player	22	320.00	2018-09-24 00:00:00	September	2018
4	Printer	41	332.50	2018-09-24 00:00:00	September	2018
4	TU	56	990.00	2018-09-24 00:00:00	September	2018
5N	Audio Player	33	580.00	2018-10-02 00:00:00	October	2018
5N	TU	14	860.00	2018-10-02 00:00:00	October	2018
5	Video Player	17	850.00	2018-10-02 00:00:00	October	2018

15 rows in set (0.00 sec)

Рисунок 3.17

## **2. Оформити звіт з лабораторної роботи**

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

## **3. Питання для самоконтролю**

1. Який оператор мови SQL використовується для вибору даних з однієї або декількох таблиць?
2. Навести загальну структуру оператора SELECT.
3. Як можна записати SQL SELECT запит, якщо необхідно вивести усі стовпці таблиці?
4. Яка конструкція використовується для вибору записів, що задовольняють критеріям пошуку?
5. Яке ключове слово використовується для виключення повторень?
6. Яка конструкція використовується для сортування значень за одним або кількома стовпцями?
7. Яким чином реалізується зворотний порядок сортування?
8. За допомогою якого ключового слова виконується обмеження вибірки?
9. За допомогою якої конструкції виконується групування рядків, що вилучаються?
10. Назвати функції агрегації, їх призначення та основні особливості.
11. Яким чином стовпцю можна призначити нове ім'я?
12. Назвіть призначення та відмінність ключового слова HAVING від WHERE?
13. Назвати основні арифметичні, логічні та оператори порівняння, їх призначення та приклади використання.
14. Призначення функції MONTH та приклади її використання.
15. Призначення функції YEAR та приклади її використання.
16. Призначення функції IFNULL та приклади її використання.
17. Призначення функції CONCAT та приклади її використання.
18. Призначення функції RTRIM та приклади її використання.
19. Призначення функції SUBSTRING та приклади її використання.
20. Призначення функції IF та приклади її використання.
21. Який оператор використовується для об'єднання результатів двох запитів.



## Лабораторна робота 4

### СТВОРЕННЯ ТА ВИКОРИСТАННЯ ПРЕДСТАВЛЕНЬ

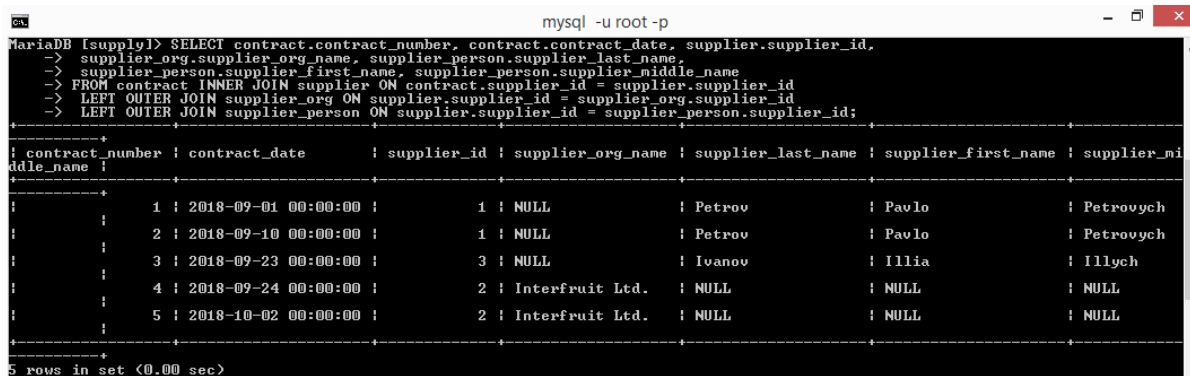
**Мета роботи:** навчитися створювати та застосовувати представлення (view) у базі даних на прикладі СУБД MySQL.

#### Хід роботи

##### 1. Створити представлення, що дозволяє при перегляді списку договорів бачити назву постачальника

Створення представлень здійснюється за допомогою оператора CREATE VIEW. Таким чином, створити представлення, яке дозволить переглядати список договорів із зазначенням назви постачальника, можна на основі наступного запиту (рисунок 4.1).

```
SELECT contract.contract_number, contract.contract_date, supplier.supplier_id,  
       supplier_org.supplier_org_name, supplier_person.supplier_last_name,  
       supplier_person.supplier_first_name, supplier_person.supplier_middle_name  
FROM contract INNER JOIN supplier ON contract.supplier_id = supplier.supplier_id  
     LEFT OUTER JOIN supplier_org ON supplier.supplier_id = supplier_org.supplier_id  
     LEFT OUTER JOIN supplier_person ON supplier.supplier_id = supplier_person.supplier_id;
```



The screenshot shows a MySQL terminal window with the following content:

```
mysql -u root -p  
MariaDB [supply]> SELECT contract.contract_number, contract.contract_date, supplier.supplier_id,  
-> supplier_org.supplier_org_name, supplier_person.supplier_last_name,  
-> supplier_person.supplier_first_name, supplier_person.supplier_middle_name  
-> FROM contract INNER JOIN supplier ON contract.supplier_id = supplier.supplier_id  
-> LEFT OUTER JOIN supplier_org ON supplier.supplier_id = supplier_org.supplier_id  
-> LEFT OUTER JOIN supplier_person ON supplier.supplier_id = supplier_person.supplier_id;
```

contract_number	contract_date	supplier_id	supplier_org_name	supplier_last_name	supplier_first_name	supplier_middle_name
1	2018-09-01 00:00:00	1	NULL	Petrov	Pavlo	Petrovych
2	2018-09-10 00:00:00	1	NULL	Petrov	Pavlo	Petrovych
3	2018-09-23 00:00:00	3	NULL	Ivanov	Illia	Illych
4	2018-09-24 00:00:00	2	Interfruit Ltd.	NULL	NULL	NULL
5	2018-10-02 00:00:00	2	Interfruit Ltd.	NULL	NULL	NULL

5 rows in set (0.00 sec)

Рисунок 4.1

Результат такого запиту має певний недолік – дані постачальників – юридичних та фізичних осіб знаходяться у різних стовпцях, а також присутні значення NULL. Цей недолік можна виправити за допомогою наступного запиту (рисунок 4.2).



```
SELECT contract.contract_number, contract.contract_date, supplier.supplier_id,
IFNULL(supplier_org.supplier_org_name, CONCAT(supplier_person.supplier_last_name, ' ',
supplier_person.supplier_first_name, ' ', supplier_person.supplier_middle_name)) AS `Supplier`
FROM contract INNER JOIN supplier ON contract.supplier_id = supplier.supplier_id
LEFT OUTER JOIN supplier_org ON supplier.supplier_id = supplier_org.supplier_id
LEFT OUTER JOIN supplier_person ON supplier.supplier_id = supplier_person.supplier_id;
```

```
mysql -u root -p
MariaDB [supply]> SELECT contract.contract_number, contract.contract_date, supplier.supplier_id,
-> IFNULL(supplier_org.supplier_org_name, CONCAT(supplier_person.supplier_last_name, ' ',
-> supplier_person.supplier_first_name, ' ', supplier_person.supplier_middle_name)) AS `Supplier`
-> FROM contract INNER JOIN supplier ON contract.supplier_id = supplier.supplier_id
-> LEFT OUTER JOIN supplier_org ON supplier.supplier_id = supplier_org.supplier_id
-> LEFT OUTER JOIN supplier_person ON supplier.supplier_id = supplier_person.supplier_id;
```

contract_number	contract_date	supplier_id	Supplier
1	2018-09-01 00:00:00	1	Petrov Pavlo Petrovych
2	2018-09-10 00:00:00	1	Petrov Pavlo Petrovych
3	2018-09-23 00:00:00	3	Ivanov Illia Illych
4	2018-09-24 00:00:00	2	Interfruit Ltd.
5	2018-10-02 00:00:00	2	Interfruit Ltd.

5 rows in set (0.00 sec)

Рисунок 4.2

Тепер можна створити дане представлення з назвою `contract_supplier` за допомогою відповідної команди мови SQL (рисунок 4.3).

```
mysql -u root -p
MariaDB [supply]> SHOW TABLES;
```

Tables_in_supply
contract
contract_supplier
supplied
supplier
supplier_org
supplier_person

6 rows in set (0.00 sec)

```
MariaDB [supply]> SELECT * FROM contract_supplier;
```

contract_number	contract_date	supplier_id	Supplier
1	2018-09-01 00:00:00	1	Petrov Pavlo Petrovych
2	2018-09-10 00:00:00	1	Petrov Pavlo Petrovych
3	2018-09-23 00:00:00	3	Ivanov Illia Illych
4	2018-09-24 00:00:00	2	Interfruit Ltd.
5	2018-10-02 00:00:00	2	Interfruit Ltd.

5 rows in set (0.01 sec)

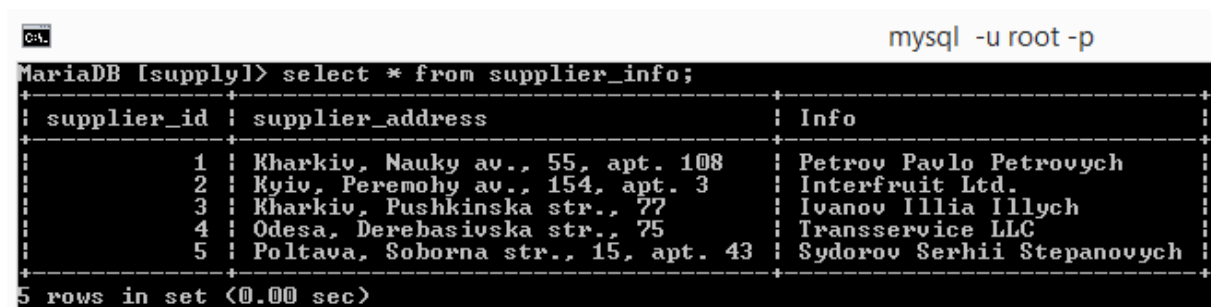
Рисунок 4.3

## 2. Створити представлення, що дозволяє користувачу працювати з обмеженими даними про постачальників

Припустимо, що для певних користувачів повинна бути доступна не уся загальна інформація про постачальників (збережена у таблиці `supplier`), а лише інформація про код та адресу постачальника. При цьому

користувач повинен мати можливість бачити дані постачальника як суб'єкта підприємницької діяльності (для юридичних осіб – назва, для фізичних – прізвище, ім'я, по батькові) (рисунок 4.4).

```
CREATE VIEW supplier_info AS
SELECT supplier.supplier_id, supplier.supplier_address,
    IFNULL(supplier_org.supplier_org_name, CONCAT(supplier_person.supplier_last_name, ' ',
        supplier_person.supplier_first_name, ' ', supplier_person.supplier_middle_name)) AS `Info`
FROM supplier LEFT OUTER JOIN supplier_org ON supplier.supplier_id = supplier_org.supplier_id
LEFT OUTER JOIN supplier_person ON supplier.supplier_id = supplier_person.supplier_id;
```



supplier_id	supplier_address	Info
1	Kharkiv, Nauky av., 55, apt. 108	Petrov Pavlo Petrovych
2	Kyiv, Peremohy av., 154, apt. 3	Interfruit Ltd.
3	Kharkiv, Pushkinska str., 77	Ivanov Illia Illych
4	Odesa, Derebasivska str., 75	Transservice LLC
5	Poltava, Soborna str., 15, apt. 43	Sydorov Serhii Stepanovych

5 rows in set (0.00 sec)

Рисунок 4.4

В разі виникнення необхідності, видалити представлення можна за допомогою оператора DROP VIEW.

### 3. Оформити звіт з лабораторної роботи

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

### 4. Питання для самоконтролю

1. Що таке представлення?
2. Назвати переваги та недоліки представлень.
3. Який оператор мови SQL використовується для створення представлень?
4. Який оператор мови SQL використовується для видалення представлень?
5. Яким чином можна перевірити наявність представлення у базі даних?
6. Як вказати список стовпців при створенні представлення?
7. Що таке вертикальне представлення?
8. Що таке горизонтальне представлення?

## Лабораторна робота 5

### СТВОРЕННЯ ТА ВИКОРИСТАННЯ ЗБЕРЕЖЕНИХ ПРОЦЕДУР ТА ТРИГЕРІВ

**Мета роботи:** навчитися створювати та застосовувати програмні об'єкти бази даних – збережені процедури та тригери, на прикладі СУБД MySQL.

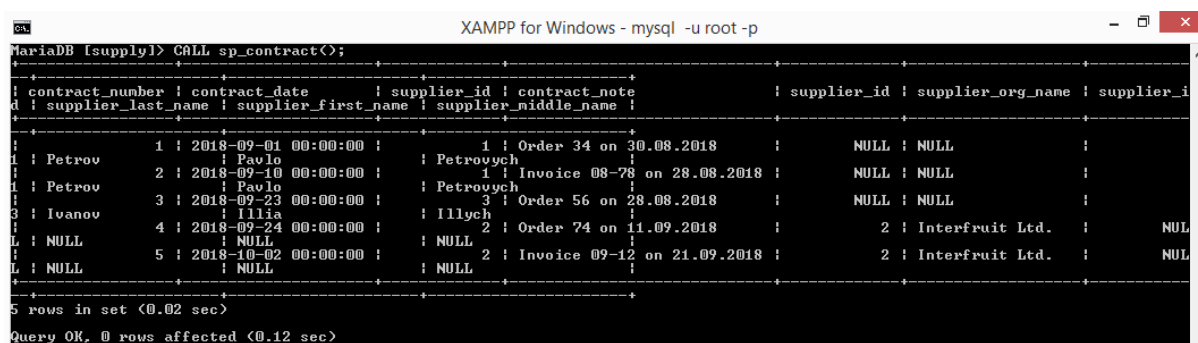
#### Хід роботи

##### 1. Створення та використання збережених процедур

Створення збережених процедур реалізується оператором CREATE PROCEDURE. Таким чином, створити збережену процедуру, яка реалізує вибірку даних з таблиць contract, supplier\_org, supplier\_person, можна за допомогою наступної команди (рисунок 5.1).

```
DELIMITER //
CREATE PROCEDURE sp_contract()
BEGIN
    SELECT *
    FROM (contract LEFT JOIN supplier_org ON
        contract.supplier_id = supplier_org.supplier_id)
    LEFT JOIN supplier_person ON
        contract.supplier_id = supplier_person.supplier_id;
END //
```

Виклик процедури здійснюється за допомогою оператора CALL.



contract_number	contract_date	supplier_id	contract_note	supplier_last_name	supplier_first_name	supplier_middle_name	supplier_id	supplier_org_name	supplier_i
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018	Petrov	Paulo	Petrovych	NULL	NULL	
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018	Petrov	Paulo	Petrovych	NULL	NULL	
3	2018-09-23 00:00:00	3	Order 56 on 28.08.2018	Ivanov	Illia	Illych	NULL	NULL	
4	2018-09-24 00:00:00	2	Order 74 on 11.09.2018	NULL	NULL	NULL	2	Interfruit Ltd.	NUL
5	2018-10-02 00:00:00	2	Invoice 09-12 on 21.09.2018	NULL	NULL	NULL	2	Interfruit Ltd.	NUL

5 rows in set (0.02 sec)  
Query OK, 0 rows affected (0.12 sec)

Рисунок 5.1

Для знайомства з особливостями створення та використання процедур з параметрами, необхідно створити збережену процедуру, яка забезпечує формування агрегатних даних за поставками для вказаного інтервалу календарних дат (рисунок 5.2).

```
DELIMITER //
CREATE PROCEDURE sp_contract_total(IN date_from timestamp,
                                   IN date_to timestamp)
BEGIN
    SELECT contract.contract_number, contract.contract_date,
           SUM(supplied.supplied_amount), SUM(supplied.supplied_amount * supplied.supplied_cost)
    FROM contract LEFT JOIN supplied ON contract.contract_number = supplied.contract_number
    WHERE contract.contract_date BETWEEN date_from AND date_to
    GROUP BY contract.contract_number, contract.contract_date;
END //
```

Здійснити виклик створеної процедури можна за допомогою наступного запиту.

```
CALL sp_contract_total('2018-09-01', '2018-10-31');
```

contract_number	contract_date	SUM(supplied.supplied_amount)	SUM(supplied.supplied_amount * supplied.supplied_cost)
1	2018-09-01 00:00:00	47	39500.00
2	2018-09-10 00:00:00	24	11350.00
3	2018-09-23 00:00:00	148	99600.00
4	2018-09-24 00:00:00	119	76112.50
5	2018-10-02 00:00:00	64	45630.00

5 rows in set (0.01 sec)  
Query OK, 0 rows affected (0.06 sec)

Рисунок 5.2

Наступна збережена процедура призначена для виконання різних операцій модифікації даних для таблиці contract. Дана процедура використовує оператор умови IF, призначений для управління потоком даних.

```

DELIMITER //
CREATE PROCEDURE sp_contract_ops(IN op CHAR(1), IN c_num INT, IN c_date TIMESTAMP,
                                IN s_id INT, IN c_note VARCHAR(100))
BEGIN
    IF op = 'i' THEN
        INSERT INTO contract(contract_date, supplier_id, contract_note)
            VALUES(CURRENT_TIMESTAMP(), s_id, c_note);
    ELSEIF op = 'u' THEN
        UPDATE contract SET contract_date = c_date,
                            supplier_id = s_id,
                            contract_note = c_note
        WHERE contract_number = c_num;
    ELSE
        DELETE FROM contract WHERE contract_number = c_num;
    END IF;
END //

```

Наступний запит дозволяє створювати договір (рисунок 5.3).

```
CALL sp_contract_ops('i', 0, '2018-12-16', 2, 'contract inserted');
```

XAMPP for Windows - mysql -u root -p

```

MariaDB [supply]> CALL sp_contract_ops('i', 0, '2018-12-16', 2, 'contract inserted');
Query OK, 1 row affected (0.01 sec)

MariaDB [supply]> select * from contract;

```

contract_number	contract_date	supplier_id	contract_note
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018
3	2018-09-23 00:00:00	3	Order 56 on 28.08.2018
4	2018-09-24 00:00:00	2	Order 74 on 11.09.2018
5	2018-10-02 00:00:00	2	Invoice 09-12 on 21.09.2018
6	2018-12-27 13:10:43	2	contract inserted

6 rows in set (0.00 sec)

Рисунок 5.3

Наступний запит дозволяє модифікувати договір (рисунок 5.4).

```
CALL sp_contract_ops('u', 6, '2018-12-31', 2, 'contract updated');
```

XAMPP for Windows - mysql -u root -p

```

MariaDB [supply]> CALL sp_contract_ops('u', 6, '2018-12-31', 2, 'contract updated');
Query OK, 1 row affected (0.01 sec)

MariaDB [supply]> select * from contract;

```

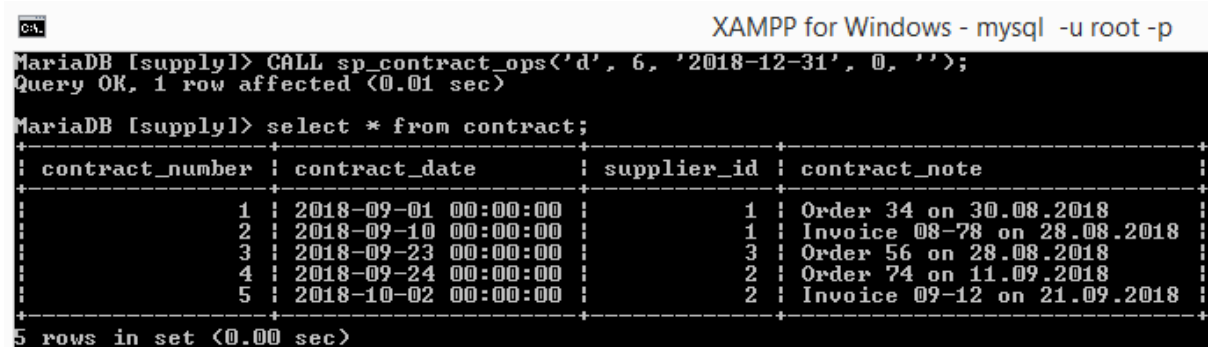
contract_number	contract_date	supplier_id	contract_note
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018
3	2018-09-23 00:00:00	3	Order 56 on 28.08.2018
4	2018-09-24 00:00:00	2	Order 74 on 11.09.2018
5	2018-10-02 00:00:00	2	Invoice 09-12 on 21.09.2018
6	2018-12-31 00:00:00	2	contract updated

6 rows in set (0.00 sec)

Рисунок 5.4

Наступний запит дозволяє видаляти договір (рисунк 5.5).

```
CALL sp_contract_ops('d', 6, '2018-12-31', 0, '');
```



The screenshot shows a MySQL command prompt window titled "XAMPP for Windows - mysql -u root -p". The user is logged in as root. The prompt shows the following commands and results:

```
MariaDB [supply]> CALL sp_contract_ops('d', 6, '2018-12-31', 0, '');
Query OK, 1 row affected (0.01 sec)

MariaDB [supply]> select * from contract;
```

contract_number	contract_date	supplier_id	contract_note
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018
3	2018-09-23 00:00:00	3	Order 56 on 28.08.2018
4	2018-09-24 00:00:00	2	Order 74 on 11.09.2018
5	2018-10-02 00:00:00	2	Invoice 09-12 on 21.09.2018

5 rows in set (0.00 sec)

Рисунок 5.5

## 2. Створення та використання тригерів

Припустимо, що при вводі даних у таблицю contract, у якій зберігається інформація про договори на постачання продукції, поле contract\_date, у якому зберігається дата укладення договору, повинне бути обов'язково заповнене. При чому у випадку, якщо при вводі нового договору дане поле залишається незаповненим, в нього повинна бути автоматично записана поточна дата. Дану задачу можна вирішити за допомогою створення певного тригера, використовуючи відповідну команду CREATE TRIGGER (рисунк 5.6).

```
DELIMITER //
CREATE TRIGGER not_null_date BEFORE INSERT ON contract
FOR EACH ROW
BEGIN
    IF NEW.contract_date IS NULL THEN
        SET NEW.contract_date = CURRENT_TIMESTAMP();
    END IF;
END //
```

Для перевірки роботи тригера необхідно додати новий договір за допомогою наступного запиту.

```
INSERT INTO contract (supplier_id, contract_note) VALUES (1, '');
```

```

XAMPP for Windows - mysql -u root -p
MariaDB [supply]> INSERT INTO contract (supplier_id, contract_note) VALUES (1, '');
Query OK, 1 row affected (0.01 sec)

MariaDB [supply]> select * from contract;
+-----+-----+-----+-----+
| contract_number | contract_date | supplier_id | contract_note |
+-----+-----+-----+-----+
| 1 | 2018-09-01 00:00:00 | 1 | Order 34 on 30.08.2018 |
| 2 | 2018-09-10 00:00:00 | 1 | Invoice 08-78 on 28.08.2018 |
| 3 | 2018-09-23 00:00:00 | 3 | Order 56 on 28.08.2018 |
| 4 | 2018-09-24 00:00:00 | 2 | Order 74 on 11.09.2018 |
| 5 | 2018-10-02 00:00:00 | 2 | Invoice 09-12 on 21.09.2018 |
| 7 | 2018-12-27 13:30:04 | 1 | |
+-----+-----+-----+-----+
6 rows in set (0.00 sec)

```

Рисунок 5.6

В базі даних зберігається як загальна інформація про постачальників, так і інформація, яка відноситься тільки до фізичних або юридичних осіб. Одночасна наявність даних про постачальника у таблицях `supplier_org` та `supplier_person` не допускається з точки зору логіки управління бізнесом. Таким чином, виникає необхідність складного контролю відношень посилкової цілісності. Для вирішення даної задачі створимо тригер, який при введенні інформації у таблицю `supplier_person` буде контролювати наявність коду відповідного постачальника у таблиці `supplier_org` та блокувати введення даних про постачальника як про фізичну особу у тому випадку, якщо вже наявні дані про даного постачальника як про юридичну особу (рисунок 5.7).

```

DELIMITER //
CREATE TRIGGER check_supplier_org BEFORE INSERT ON supplier_person
FOR EACH ROW
BEGIN
    IF NEW.supplier_id IN (SELECT supplier_id FROM supplier_org) THEN
        SET @message = CONCAT('The person with id ', NEW.supplier_id,
            ' is already stored as the organization!');
        SIGNAL SQLSTATE '45001';
        SET MESSAGE_TEXT = @message;
    END IF;
END //

```

Для перевірки роботи тригеру необхідно спробувати додати дані про постачальника 2 (який вже зберігається у БД в якості юридичної особи) як про фізичну особу.

```

INSERT INTO supplier_person VALUES (2, 'Makarov', 'Oleg', 'Petrovych');

```

```
XAMPP for Windows - mysql -u root -p
MariaDB [supplyl]> INSERT INTO supplier_person VALUES (2, 'Makarov', 'Oleg', 'Petrovych');
ERROR 1644 (45001): The person with id 2 is already stored as the organization!
MariaDB [supplyl]> select * from supplier_person;
+-----+-----+-----+-----+
| supplier_id | supplier_last_name | supplier_first_name | supplier_middle_name |
+-----+-----+-----+-----+
| 1 | Petrov | Pavlo | Petrovych |
| 3 | Ivanov | Illia | Illych |
| 5 | Sydorov | Serhii | Stepanovych |
+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

Рисунок 5.7

Для видалення збережених процедур та тригерів необхідно скористатися операторами DROP PROCEDURE та DROP TRIGGER відповідно.

### 3. Оформити звіт з лабораторної роботи

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

### 4. Питання для самоконтролю

1. Що таке збережена процедура?
2. Назвати переваги використання збережених процедур.
3. Який оператор використовується для створення збереженої процедури?
4. Яким чином можна визначити вхідні або вихідні параметри збереженої процедури?
5. Для чого використовується оператор IF?
6. Яке призначення операторів BEGIN та END?
7. Що таке тригер?
8. Назвати переваги використання тригерів.
9. За допомогою якого оператора тригер зв'язується з таблицею?
10. До яких подій, пов'язаних зі зміною вмісту таблиці, можна прив'язати тригер?
11. Яким чином можна визначити до чи після операції зміни вмісту таблиці повинен спрацьовувати тригер?
12. Для чого використовуються префікси NEW та OLD?
13. Яке призначення оператора SET?
14. За допомогою яких операторів виконується видалення процедур та тригерів?



## Лабораторна робота 6

### ОСНОВИ ВИКОРИСТАННЯ ЗАСОБІВ КОНТРОЛЮ ЦІЛІСНОСТІ ДАНИХ

**Мета роботи:** вивчити основи роботи із засобами контролю посилкової цілісності даних на прикладі СУБД MySQL.

#### Хід роботи

**Увага!** Перш ніж перейти до виконання лабораторної роботи, необхідно створити тимчасову базу даних, використовуючи запити, використані у лабораторній роботі 2. В усіх подальших пунктах даної лабораторної роботи передбачається використання тимчасової бази даних.

#### 1. Вивчити особливості роботи механізму посилкової цілісності NO ACTION

Особливості роботи механізму посилкової цілісності NO ACTION розглянемо на прикладі відношень між таблицями supplier та contract, supplier та supplier\_person, supplier та supplier\_org. Дані таблиці пов'язані між собою за полем supplier\_id. У цьому зв'язку таблиця supplier є батьківською, а таблиці contract, supplier\_org, supplier\_person – дочірніми. Для вивчення особливостей роботи механізму посилкової цілісності необхідно виконати наступну послідовність дій.

Встановити параметри ON DELETE та ON UPDATE, що визначають поведінку під час видалення та оновлення записів з таблиці-предка.

```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE supplier_org
DROP FOREIGN KEY supplier_org_ibfk_1;

ALTER TABLE supplier_org
ADD CONSTRAINT supplier_org_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE supplier_person
DROP FOREIGN KEY supplier_person_ibfk_1;

ALTER TABLE supplier_person
ADD CONSTRAINT supplier_person_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

Припустимо, що в силу певних причин необхідно видалили постачальника з кодом 4 (рисунок 6.1).

```
DELETE FROM supplier WHERE supplier_id = 4;
```

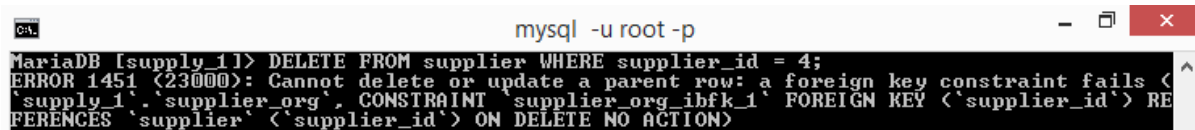
A screenshot of a MySQL terminal window titled 'mysql -u root -p'. The prompt is 'MariaDB [supply\_1]'. The user has entered the command 'DELETE FROM supplier WHERE supplier\_id = 4;'. The terminal displays an error: 'ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (<supply\_1>.`supplier\_org`, CONSTRAINT `supplier\_org\_ibfk\_1` FOREIGN KEY (<supplier\_id>) REFERENCES `supplier` (<supplier\_id>) ON DELETE NO ACTION)'.

Рисунок 6.1

Таким чином, для того, щоб видалити даного постачальника, необхідно попередньо видалити усі пов'язані з ним дані. Для цього потрібно видалити відповідний запис з таблиці `supplier_org` та перевірити наявність договорів з даним постачальником у таблиці `contract`. Якщо такі договори є, їх також потрібно видалити (при цьому необхідно мати на увазі, що може виникнути потреба видалення й вмісту даних договорів). Після цього необхідно спробувати видалити постачальника з кодом 4 знову. Якщо зв'язаних з ним даних немає, постачальник буде видалений.

Припустимо, що в силу певних причин виникла необхідність для постачальника з кодом 5 змінити код на 7 (рисунок 6.2).

```
UPDATE supplier SET supplier_id = 7 WHERE supplier_id = 5;
```

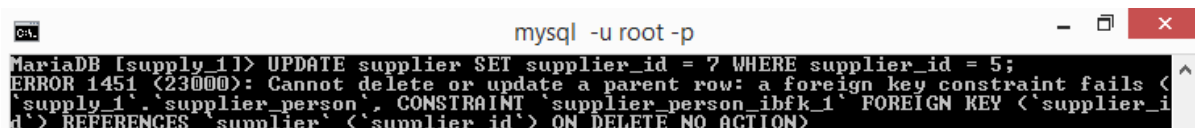
A screenshot of a MySQL terminal window titled 'mysql -u root -p'. The prompt is 'MariaDB [supply\_1]'. The user has entered the command 'UPDATE supplier SET supplier\_id = 7 WHERE supplier\_id = 5;'. The terminal displays an error: 'ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (<supply\_1>.`supplier\_person`, CONSTRAINT `supplier\_person\_ibfk\_1` FOREIGN KEY (<supplier\_id>) REFERENCES `supplier` (<supplier\_id>) ON DELETE NO ACTION)'.

Рисунок 6.2

Оскільки договори з даним постачальником відсутні, посилання на нього є лише в таблиці `supplier_person`. Видаливши цей запис, необхідно повторити зміну коду постачальника з 5 на 7. Тепер ця операція повинна пройти успішно. Після цього необхідно перевірити вміст таблиць.

## 2. Вивчити особливості роботи механізму посилкової цілісності CASCADE

Змінимо механізми посилкової цілісності для зв'язків між усіма розглянутими вище таблицями на CASCADE.

```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE supplier_org
DROP FOREIGN KEY supplier_org_ibfk_1;

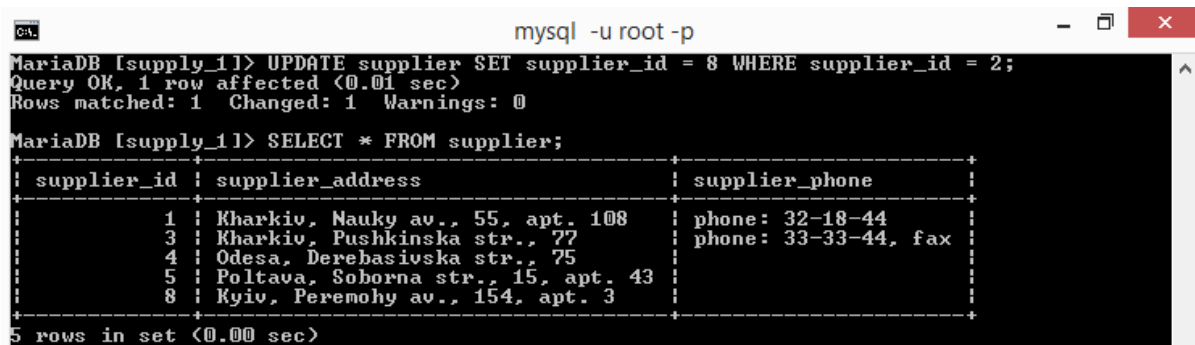
ALTER TABLE supplier_org
ADD CONSTRAINT supplier_org_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;

ALTER TABLE supplier_person
DROP FOREIGN KEY supplier_person_ibfk_1;

ALTER TABLE supplier_person
ADD CONSTRAINT supplier_person_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;
```

Припустимо, що в силу певних причин виникла необхідність для постачальника з кодом 2 змінити код на 8 (рисунок 6.3).

```
UPDATE supplier SET supplier_id = 8 WHERE supplier_id = 2;
```



```
mysql -u root -p
MariaDB [supply_1]> UPDATE supplier SET supplier_id = 8 WHERE supplier_id = 2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [supply_1]> SELECT * FROM supplier;
```

supplier_id	supplier_address	supplier_phone
1	Kharkiv, Nauky av., 55, apt. 108	phone: 32-18-44
3	Kharkiv, Pushkinska str., 77	phone: 33-33-44, fax
4	Odesa, Derebasivska str., 75	
5	Poltava, Soborna str., 15, apt. 43	
8	Kyiv, Peremohy av., 154, apt. 3	

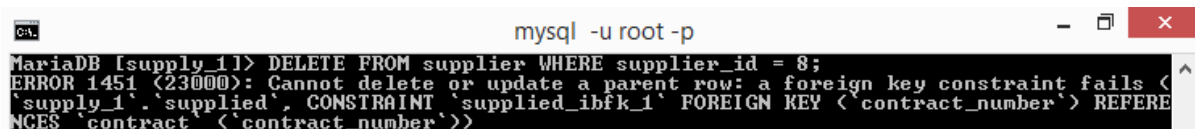
```
5 rows in set (0.00 sec)
```

Рисунок 6.3

Перевірити наявність відповідних змін у таблиці supplier\_org.

Тепер припустимо, що даного постачальника (який зараз має код 8) треба видалити (рисунок 6.4).

```
DELETE FROM supplier WHERE supplier_id = 8;
```



```
mysql -u root -p
MariaDB [supply_1]> DELETE FROM supplier WHERE supplier_id = 8;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (<supply_1>.`supplier`, CONSTRAINT `supplier_ibfk_1` FOREIGN KEY (<contract_number>) REFERENCES <contract> (<contract_number>))
```

Рисунок 6.4

Визначити причину, через яку записи не були видалені. Внести необхідні зміни у механізми посилкової цілісності необхідних таблиць для того, щоб необхідні дані все ж були видалені.

### 3. Вивчити особливості роботи механізму посилкової цілісності SET NULL

Особливості механізму посилкової цілісності SET NULL розглянемо на прикладі таблиць supplier та contract.

Змінимо механізми посилкової цілісності для зв'язків між усіма розглянутими вище таблицями на SET NULL.

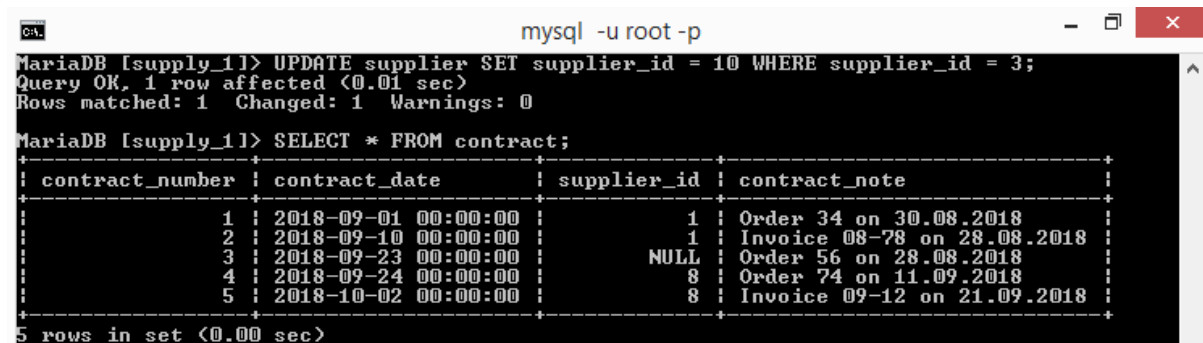
```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

ALTER TABLE contract
MODIFY supplier_id INT NULL;

ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE SET NULL ON UPDATE SET NULL;
```

В таблиці supplier змінити код постачальника 3 на 10. Перевірити дані в таблиці contract (рисунок 6.5).

```
UPDATE supplier SET supplier_id = 10 WHERE supplier_id = 3;
```



```
mysql -u root -p
MariaDB [supply_1]> UPDATE supplier SET supplier_id = 10 WHERE supplier_id = 3;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [supply_1]> SELECT * FROM contract;
```

contract_number	contract_date	supplier_id	contract_note
1	2018-09-01 00:00:00	1	Order 34 on 30.08.2018
2	2018-09-10 00:00:00	1	Invoice 08-78 on 28.08.2018
3	2018-09-23 00:00:00	NULL	Order 56 on 28.08.2018
4	2018-09-24 00:00:00	8	Order 74 on 11.09.2018
5	2018-10-02 00:00:00	8	Invoice 09-12 on 21.09.2018

5 rows in set (0.00 sec)

Рисунок 6.5

Замість NULL встановити значення коду постачальника 10 для договору з номером 3.

### 4. Оформити звіт з лабораторної роботи

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

## **5. Питання для самоконтролю**

1. Чи є конструкції ON DELETE та ON UPDATE обов'язковими при формуванні команди CREATE TABLE або ALTER TABLE?
2. Яку поведінку СУБД задає конструкція ON DELETE?
3. Яку поведінку СУБД задає конструкція ON UPDATE?
4. Які параметри можна встановити після конструкцій ON DELETE та ON UPDATE?
5. Назвати особливості механізму посилкової цілісності CASCADE.
6. Назвати особливості механізму посилкової цілісності SET NULL.
7. Назвати особливості механізму посилкової цілісності NO ACTION.
8. Назвати особливості механізму посилкової цілісності SET DEFAULT.
9. Назвати особливості механізму посилкової цілісності RESTRICT.
10. Чому у даній лабораторній роботі не було розглянуто роботу з механізмом посилкової цілісності SET DEFAULT?
11. Яким чином можна встановити той чи інший механізм посилкової цілісності для зовнішнього ключа таблиці?
12. Навіщо перед тим, як встановити механізм SET NULL, була виконана модифікація поля supplier\_id таблиці contract?
13. В яких випадках не рекомендується використання механізму посилкової цілісності CASCADE?
14. Який механізм посилкової цілісності завжди використовується за замовченням в СУБД MySQL у випадку, якщо конструкції ON DELETE та ON UPDATE не були визначені?

## Лабораторна робота 7

### РОБОТА З ТРАНЗАКЦІЯМИ

**Мета роботи:** вивчити основи роботи з механізмом транзакцій на прикладі СУБД MySQL.

#### Хід роботи

**Увага!** Перш ніж перейти до виконання лабораторної роботи, необхідно створити тимчасову базу даних, використовуючи запити, використані у лабораторній роботі 2. В усіх подальших пунктах даної лабораторної роботи передбачається використання тимчасової бази даних.

#### 1. Створити запит, що ілюструє роботу механізму транзакцій при додаванні даних в одну таблицю

Розглянемо послідовність дій при створенні та використанні запиту, за допомогою якого запускається транзакція, до таблиці `supplied` додається новий запис, а потім імітується ситуація некоректного або коректного завершення транзакції. Стан таблиці контролюється до початку транзакції, під час виконання транзакції та після її завершення. Для цього необхідно виконати наступну послідовність дій.

```
SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
       supplier.supplier_address, contract.contract_date
FROM supplied, contract, supplier
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
AND contract.contract_number = 1;

SET AUTOCOMMIT = 0;
START TRANSACTION;
INSERT INTO supplied VALUES (1, 'Vacuum cleaner', 22, 390);

SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
       supplier.supplier_address, contract.contract_date
FROM supplied, contract, supplier
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
AND contract.contract_number = 1;

ROLLBACK;

SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
       supplier.supplier_address, contract.contract_date
FROM supplied, contract, supplier
WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
AND contract.contract_number = 1;
```

Запити SELECT дозволяють вивести дані, які ілюструють стан таблиці до початку транзакції (рисунок 7.1), в процесі виконання транзакції та після завершення транзакції.

```
mysql -u root -p
MariaDB [supply_11] SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
-> supplier.supplier_address, contract.contract_date
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
-> AND contract.contract_number = 1;
```

contract_number	supplied_product	supplied_cost	supplied_amount	supplier_address	contract_date
1	Audio Player	700.00	25	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	TU	1300.00	10	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Video Player	750.00	12	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00

3 rows in set (0.00 sec)

Рисунок 7.1

Як видно з наведених даних, новий запис у таблиці з'являється (рисунок 7.2), а потім зникає (рисунок 7.3).

```
mysql -u root -p
MariaDB [supply_11] SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
-> supplier.supplier_address, contract.contract_date
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
-> AND contract.contract_number = 1;
```

contract_number	supplied_product	supplied_cost	supplied_amount	supplier_address	contract_date
1	Audio Player	700.00	25	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	TU	1300.00	10	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Vacuum cleaner	390.00	22	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Video Player	750.00	12	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00

4 rows in set (0.00 sec)

Рисунок 7.2

```
mysql -u root -p
MariaDB [supply_11] ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

MariaDB [supply_11]
MariaDB [supply_11] SELECT supplied.contract_number, supplied.supplied_product, supplied.supplied_cost, supplied.supplied_amount,
-> supplier.supplier_address, contract.contract_date
-> FROM supplied, contract, supplier
-> WHERE contract.contract_number = supplied.contract_number AND supplier.supplier_id = contract.supplier_id
-> AND contract.contract_number = 1;
```

contract_number	supplied_product	supplied_cost	supplied_amount	supplier_address	contract_date
1	Audio Player	700.00	25	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	TU	1300.00	10	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00
1	Video Player	750.00	12	Kharkiv, Nauky av., 55, apt. 108	2018-09-01 00:00:00

3 rows in set (0.00 sec)

Рисунок 7.3

Тепер необхідно розглянути ситуацію коректного завершення транзакції. Для цього у наведеному тексті запиту необхідно змінити оператор ROLLBACK на COMMIT. Виконати запит та проаналізувати отримані результати.

## 2. Створити запит, що ілюструє роботу механізму транзакцій при додаванні даних в декілька таблиць

Розглянемо послідовність дій при створенні та використанні запиту, за допомогою якого запускається транзакція, а потім створюється новий постачальник, з цим постачальником укладається договір на постачання, за цим договором поставляється продукція. Імітується ситуація некоректного або коректного завершення транзакції. Стан таблиць контролюється до початку транзакції, в процесі виконання транзакції та після завершення транзакції. Для цього необхідно виконати наступну послідовність дій.

```
SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

SET AUTOCOMMIT = 0;
START TRANSACTION;
INSERT INTO supplier (supplier_id, supplier_address, supplier_phone)
VALUES (6, 'Kyiv, Velyka Vasylkivska st., 55', '');
INSERT INTO contract (contract_date, supplier_id, contract_note)
VALUES ('2018-12-12', 6, '');
INSERT INTO supplied VALUES (6, 'Vacuum cleaner', 22, 390);
INSERT INTO supplied VALUES (6, 'Coffee machine', 33, 90);

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

ROLLBACK;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;
```

Запити SELECT дозволяють вивести дані, які ілюструють стан таблиць до початку транзакції, в процесі виконання транзакції та після завершення транзакції. Як видно з наведених даних, нові записи у таблицях з'являється, а потім зникають.

Тепер необхідно розглянути ситуацію коректного завершення транзакції. Для цього у наведеному тексті запиту необхідно змінити оператор ROLLBACK на COMMIT. Виконати запит та проаналізувати отримані результати.



### 3. Створити запит, що ілюструє роботу механізму транзакцій при зміні даних в декількох таблицях

Розглянемо послідовність дій при створенні та використанні запиту, за допомогою якого запускається транзакція, потім змінюються дані, введені у таблиці при виконанні попереднього запиту. Імітується ситуація некоректного або коректного завершення транзакції. Стан таблиць контролюється до початку транзакції, в процесі виконання транзакції та після завершення транзакції. Для цього необхідно виконати наступну послідовність дій.

```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

ALTER TABLE contract
ADD CONSTRAINT contract_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

SET AUTOCOMMIT = 0;
START TRANSACTION;
UPDATE supplier SET supplier_id = 22 WHERE supplier_id = 6;
UPDATE supplied SET supplied_cost = supplied_cost * 1.1 WHERE contract_number = 8;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied WHERE contract_number = 8;

ROLLBACK;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied WHERE contract_number = 8;
```

Запити SELECT дозволяють вивести дані, які ілюструють стан таблиць до початку транзакції, в процесі виконання транзакції та після завершення транзакції. Як видно з наведених даних, нові записи у таблицях з'являється, а потім зникають.

Тепер необхідно розглянути ситуацію коректного завершення транзакції. Для цього у наведеному тексті запиту необхідно змінити оператор ROLLBACK на COMMIT. Виконати запит та проаналізувати отримані результати.

### 4. Створити запит, що ілюструє роботу механізму транзакцій при видаленні даних з декількох таблиць

Розглянемо послідовність дій при створенні та використанні запиту, за допомогою якого запускається транзакція, в рамках якої видаляється постачальник, який був створений при виконанні запиту 2 та дані якого були змінені при виконанні запиту 3. З урахуванням механізму контролю

посилкової цілісності, що використовується (CASCADE), дані будуть видалені у декількох таблицях. Імітується ситуація некоректного або коректного завершення транзакції. Стан таблиць контролюється до початку транзакції, в процесі виконання транзакції та після завершення транзакції. Для цього необхідно виконати наступну послідовність дій.

```
ALTER TABLE supplied
DROP FOREIGN KEY supplied_ibfk_1;

ALTER TABLE supplied
ADD CONSTRAINT supplied_ibfk_1 FOREIGN KEY (contract_number) REFERENCES contract(contract_number) ON DELETE CASCADE ON UPDATE CASCADE;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

SET AUTOCOMMIT = 0;
START TRANSACTION;
DELETE FROM supplier WHERE supplier_id = 22;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;

ROLLBACK;

SELECT * FROM supplier;
SELECT * FROM contract;
SELECT * FROM supplied;
```

Запити SELECT дозволяють вивести дані, які ілюструють стан таблиць до початку транзакції, в процесі виконання транзакції та після завершення транзакції. Як видно з наведених даних, нові записи у таблицях з'являється, а потім зникають.

Тепер необхідно розглянути ситуацію коректного завершення транзакції. Для цього у наведеному тексті запиту необхідно змінити оператор ROLLBACK на COMMIT. Виконати запит та проаналізувати отримані результати.

## 5. Оформити звіт з лабораторної роботи

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

## 6. Питання для самоконтролю

1. Що таке транзакція?
2. Таблиці яких типів у СУБД MySQL підтримують транзакції?
3. Таблиці яких типів у СУБД MySQL не підтримують транзакції?
4. Яким чином у СУБД MySQL можна відключити режим автоматичного завершення транзакцій?
5. Який оператор використовується для завершення транзакції?

6. Який оператор використовується для відкату змін, виконаних транзакцією?

7. За допомогою якої команди у СУБД MySQL можна включити режим автоматичного завершення транзакцій для окремої послідовності операторів?

8. З таблицями якого типу можуть бути використані оператори SAVEPOINT та ROLLBACK TO SAVEPOINT?

9. Яке призначення операторів SAVEPOINT та ROLLBACK TO SAVEPOINT?

10. З якими проблемами пов'язане паралельне виконання транзакцій?

11. Які існують рівні ізоляції транзакцій та які проблеми кожен з цих рівнів дозволяє вирішити?

12. Який тип таблиць використовується у MySQL за замовченням (починаючи з версії 5.5)?

13. Які рівні ізоляції транзакцій підтримує InnoDB?

14. Який рівень ізоляції транзакцій за замовченням використовується у InnoDB?

## Лабораторна робота 8

### УПРАВЛІННЯ ПРАВАМИ КОРИСТУВАЧІВ

**Мета роботи:** вивчити основи роботи з обліковими записами та привілеями користувачів на прикладі СУБД MySQL.

#### Хід роботи

##### 1. Створити нові облікові записи користувачів

Система управління базами даних MySQL є багатокористувацьким середовищем, тому для доступу до таблиць бази даних supply можуть бути створені різні облікові записи з різним рівнем привілеїв.

Обліковому запису менеджера із закупівель можна надати привілеї на перегляд таблиць supplier, supplier\_org, supplier\_person та contract, додавання нових записів, видалення та оновлення вже існуючих записів у даних таблицях.

Адміністратору бази даних supply можна надати більш широкі повноваження (можливість створення таблиць, редагування та видалення вже існуючих, створення та редагування облікових записів користувачів тощо).

Для працівника складу достатньо лише перегляду таблиць contract та supplied, а також додавання нових записів, видалення та оновлення вже існуючих записів у таблиці supplied.

Розглянемо створення облікових записів для різних користувачів бази даних.

```
CREATE USER 'admin'@'localhost' IDENTIFIED BY 'admin123';  
CREATE USER 'manager'@'localhost' IDENTIFIED BY 'manager123';  
CREATE USER 'storekeeper'@'localhost' IDENTIFIED BY 'storekeeper123';
```

Даний запит дозволяє створити облікові записи для наступних користувачів:

- 1) адміністратора з паролем «admin123»;
- 2) менеджера з закупівель з паролем «manager123»;
- 3) працівника складу з паролем «storekeeper123».

Для видалення облікового запису використовується оператор DROP USER. Зміна імені користувача в обліковому записі виконується за допомогою оператора RENAME USER %old\_name% TO %new\_name%.

Оскільки усі облікові записи користувачів зберігаються у таблиці user системної бази даних mysql, перевірити створення розглянутих облікових записів можна за допомогою наступного запита (рисунок 8.1):

```
SELECT Host, User, Password FROM mysql.user;
```

XAMPP for Windows

Host	User	Password
localhost	root	
127.0.0.1	root	
::1	root	
localhost	pma	
%	supply_manager	*D3EA2B50EA2CDB63852452342425A884B6C6A8DC
localhost	supply_manager	*D3EA2B50EA2CDB63852452342425A884B6C6A8DC
localhost	manager	*1B2333B70420F3DB5F4F164A9B89E21810F06840
localhost	admin	*01A6717B58FF5C7EAF66CB7C96F7428EA65FE4C
localhost	storekeeper	*6A8DA8D9B9189005A0B1791874632DFD2DDD7DFA

10 rows in set (0.00 sec)

Рисунок 8.1

## 2. Призначити привілеї для створених облікових записів

Розглянуті вище оператори дозволяють створювати, видаляти та редагувати облікові записи, однак вони не дозволяють змінювати привілеї користувача – повідомляти MySQL, який користувач має право тільки на читання інформації, який на читання та редагування, а кому надані права змінювати структуру БД та створювати облікові записи.

Необхідно призначити привілеї для створених облікових записів.

```
GRANT ALL ON supply.* TO 'admin'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE ON supply.supplier TO 'manager'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON supply.supplier_org TO 'manager'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON supply.supplier_person TO 'manager'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON supply.contract TO 'manager'@'localhost';
GRANT SELECT ON supply.supplied TO 'manager'@'localhost';
GRANT EXECUTE ON supply.* TO 'manager'@'localhost';

GRANT SELECT, INSERT, UPDATE, DELETE ON supply.supplied TO 'storekeeper'@'localhost';
GRANT SELECT ON supply.contract TO 'storekeeper'@'localhost';
GRANT EXECUTE ON supply.* TO 'storekeeper'@'localhost';
```

Для позбавлення облікового запису користувача певних привілеїв використовується оператор REVOKE. Даний оператор не видаляє облікові записи, а лише віднімає надані раніше привілеї. Тому для остаточного видалення облікового запису необхідно скористатися оператором DROP USER.

Перевірити привілеї облікового запису admin, якому були надані усі права на рівні бази даних supply, можна за допомогою наступного запити (рисунок 8.2).

```
SELECT * FROM mysql.db
WHERE Db = 'supply';
```

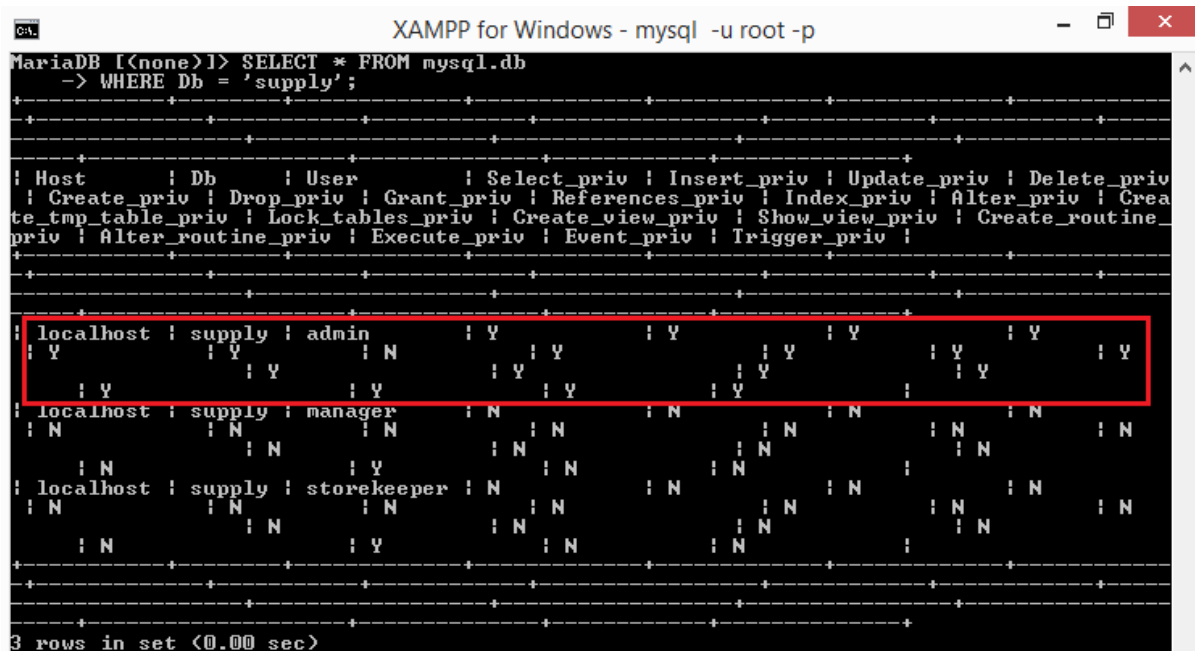


Рисунок 8.2

Аналогічно можна перевірити привілеї облікових записів manager та storekeeper, для яких були визначені певні обмеження у роботі з таблицями бази даних supply (рисунок 8.3).

```
SELECT Db, User, Table_name, Table_priv FROM mysql.tables_priv
WHERE Db = 'supply';
```

```

XAMPP for Windows - mysql -u root -p
MariaDB [(none)]> SELECT Db, User, Table_name, Table_priv FROM mysql.tables_priv
-> WHERE Db = 'supply';
+-----+-----+-----+-----+
| Db      | User      | Table_name      | Table_priv      |
+-----+-----+-----+-----+
| supply  | manager   | supplier        | Select,Insert,Update,Delete |
| supply  | manager   | supplier_org    | Select,Insert,Update,Delete |
| supply  | manager   | supplier_person | Select,Insert,Update,Delete |
| supply  | manager   | contract        | Select,Insert,Update,Delete |
| supply  | manager   | supplied        | Select           |
| supply  | storekeeper | supplied        | Select,Insert,Update,Delete |
| supply  | storekeeper | contract        | Select           |
+-----+-----+-----+-----+
7 rows in set (0.00 sec)

```

Рисунок 8.3

Крім того, певним користувачам необхідно надати також привілеї, які дозволять їм використовувати представлення, що містяться у базі даних supply. Наприклад, користувачу manager повинні бути надані права для перегляду представлень contract\_supplier та supplier\_info, тоді як для користувача storekeeper повинне бути доступним лише представлення contract\_supplier.

### 3. Оформити звіт з лабораторної роботи

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

### 4. Питання для самоконтролю

1. Який вигляд має обліковий запис користувача у СУБД MySQL?
2. З яких складових формується обліковий запис?
3. Яке призначення у складових облікового запису?
4. Яким чином можна переглянути усі облікові записи?
5. Яка команда використовується для створення облікового запису?
6. Яка команда використовується для видалення облікового запису?
7. Яким чином можна змінити ім'я користувача в обліковому записі?
8. За допомогою якого оператора можна визначити певні привілеї для необхідного облікового запису?
9. Який оператор може бути використаний для відміни привілеїв?
10. Які привілеї можуть бути визначені для облікового запису?
11. Які існують рівні призначення привілеїв?
12. Яким чином можна перевірити глобальні привілеї, привілеї бази даних та привілеї таблиць?

## Лабораторна робота 9

### РОЗРОБКА ПРИКЛАДНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ РОБОТИ З БАЗОЮ ДАНИХ

**Мета роботи:** вивчити основи розробки прикладного програмного забезпечення, призначеного для роботи з СУБД MySQL, з використанням мови PHP.

#### Хід роботи

**Увага!** У лабораторній роботі буде продемонстровано створення лише спрощеного фрагменту застосунку для роботи з базою даних.

#### 1. Визначити основні функціональні можливості програмного забезпечення

Основні функціональні можливості фрагменту web-застосунку, який призначений для роботи з базою даних supply, наведені у вигляді UML діаграми прецедентів (рисунок 9.1).

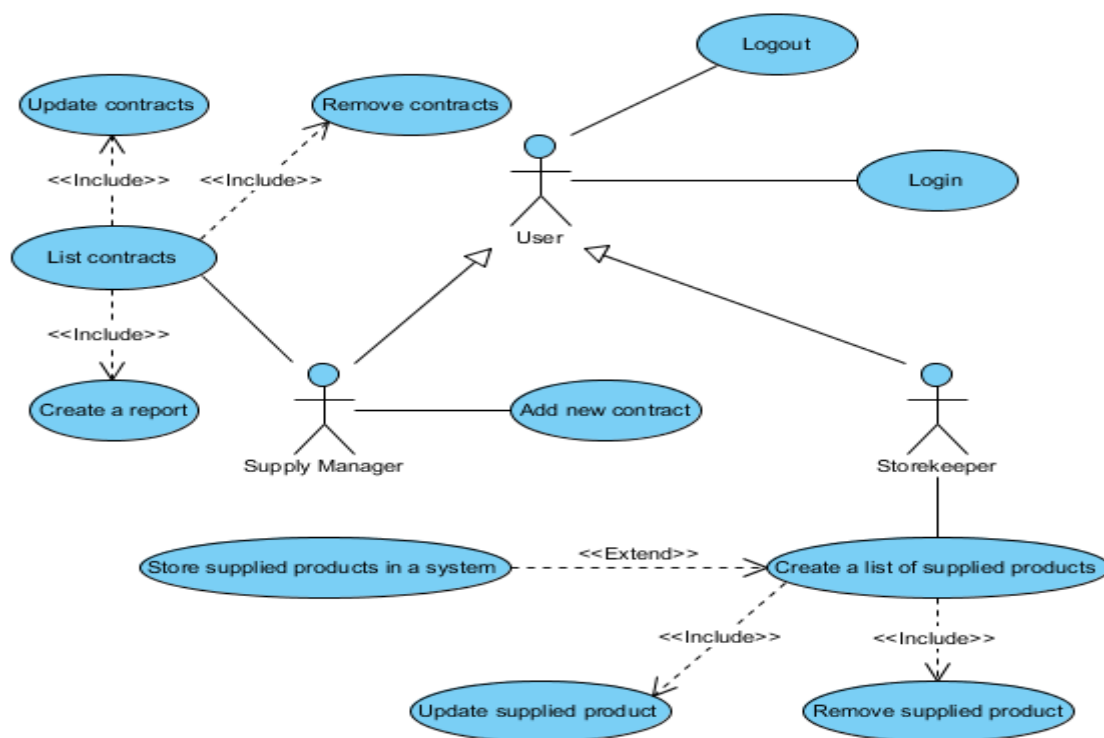


Рисунок 9.1



## 2. Розробити сторінку для авторизації користувачів програмного забезпечення

Усі сторінки web-застосунку необхідно розмістити у каталозі `xampp/htdocs/supply`.

Перш ніж приступати до створення сторінки авторизації, необхідно розробити функціональність програмного забезпечення щодо встановлення з'єднання з базою даних. Для цього створити файл `connect.php` з наступним вмістом.

```
<?php
function db_conn() {
    $server = "localhost";
    $user = $_SESSION["user"];
    $pass = $_SESSION["pass"];
    $db = "supply";

    $conn = @mysqli_connect($server, $user, $pass, $db);

    if (!$conn) {
        session_unset();
        session_destroy();

        die("Connection failed: " . mysqli_connect_error());
    }

    return $conn;
}
?>
```

Крім того, необхідно розробити основну сторінку web-застосунку, для чого створити файл `index.php` з наступним вмістом.

```
1 <?php
2 session_start();
3
4 require_once("connect.php");
5
6 $conn = NULL;
7
8 # check for a user session
9 if (isset($_SESSION["user"])) {
10     $conn = db_conn();
11     include("action.php");
12 } else {
13     # redirected to login page if the user is not set
14     header("location: login.php");
15 }
16 ?>
```

Рядки 2 – 4 містять початок користувацького сеансу та підключення файлу, що містить функцію `db_conn()` для встановлення з'єднання з базою

даних. Рядки 9 – 15 містять перевірку наявності користувацького сеансу та підключення до бази даних. В разі, якщо користувач не був авторизований, він буде перенаправлений на сторінку авторизації (рядок 14). Рядок 11 визначає підключення файлу, який містить обробку форм додавання, оновлення та видалення даних, він буде створений пізніше.

```
17 <!DOCTYPE html>
18 <html>
19 <head>
20 <title>Supply</title>
21 </head>
22 <body>
23 <p>
24 <b>User:</b> <i><?=$_SESSION["user"] ?></i> | <a href="logout.php">Logout</a>
25 </p>
26 <?php
27 # display content depending on the user type
28 if ($_SESSION["user"] == "manager") {
29     include("manager.php");
30 }
31
32 if ($_SESSION["user"] == "storekeeper") {
33     include("storekeeper.php");
34 }
35 ?>
36 </body>
37 </html>
38 <?php
39 mysqli_close($conn);
```

Наступні рядки (17 – 39) визначають вигляд головної сторінки: інформацію про поточного користувача (рисунок 9.2), вміст сторінки відповідно до типу користувача, відключення з'єднання з базою даних (рядок 39). У рядку 24 визначене посилання, яке дозволяє видалити усі змінні сеансу та завершити сеанс використання застосунку користувачем. Для цього використовується файл `logout.php`.

User: *manager* | [Logout](#)

Рисунок 9.2

```
<?php
session_start();

# remove session variables and destroy a session
session_unset();
session_destroy();

header("location: login.php");
```

Сторінка авторизації користувача зберігається у файлі login.php.

```
1 <?php
2     session_start();
3
4     # process login form
5     if (isset($_POST["login"])) {
6         session_unset();
7
8         # set user session variables
9         $_SESSION["user"] = $_POST["user"];
10        $_SESSION["pass"] = $_POST["pass"];
11
12        header("location: index.php");
13    } else {
14        # redirect to a home page if user is already signed in
15        if (isset($_SESSION["user"])) {
16            header("location: index.php");
17        }
18    }
19 ?>
```

Рядки 9 та 10 визначають запис змінних сеансу, які містять дані про обліковий запис користувача. Саме ці змінні використовуються у файлі connect.php для встановлення з'єднання з базою даних за допомогою функції mysql\_connect(). Якщо сеанс користувача вже був встановлений, він буде перенаправлений на головну сторінку index.php (рядки 15 – 17).

```
20 <!DOCTYPE html>
21 <html>
22 <head>
23     <title>Login</title>
24 </head>
25 <body>
26     <h3>Supply Application Login</h3>
27     <form method="post" action="login.php">
28         <p>
29             <b>User name</b>
30         </p>
31         <p>
32             <input type="text" name="user" required />
33         </p>
34         <p>
35             <b>Password</b>
36         </p>
37         <p>
38             <input type="password" name="pass" required />
39         </p>
40         <p>
41             <input type="submit" name="login" value="Login" />
42         </p>
43     </form>
44 </body>
45 </html>
```

У рядках 20 – 45 визначена статична структура сторінки авторизації користувача, яка містить відповідну форму з необхідними елементами користувацького інтерфейсу (рисунок 9.3).

### Supply Application Login

User name

Password

Login

Рисунок 9.3

### 3. Розробити функціональність програмного забезпечення для роботи менеджера з закупівель

Сторінка, що містить функціональність програмного забезпечення для роботи менеджера з закупівель, міститься у файлі `manager.php`.

```
1  <?php
2  # check for a user session
3  if (!isset($_SESSION["user"])) {
4      header("location: login.php");
5  }
6  ?>
7
8  <h3>Contracts</h3>
9  <p>
10     <?php
11     # if the page is in record's create/update or delete mode (action parameter is set) - show 'back' link
12     if (isset($_GET["action"]) && ($_GET["action"] == "create" || $_GET["action"] == "update"
13         || $_GET["action"] == "delete")) {
14     }
15     <a href="index.php">Back</a>
16     <?php
17     # otherwise - show 'new record' link
18     } else {
19     }
20     <a href="index.php?action=create">New contract</a>
21     <?php
22     }
23     ?>
24 </p>
```

Рядки 1 – 24 містять перевірку наявності користувацького сеансу, а також режиму роботи з даними про договори (створення, оновлення або видалення), від чого залежить елемент інтерфейсу – посилання New

contract, призначене для створення нового договору (рисунок 9.4), або Back – для повернення до перегляду даних про усі договори (рисунок 9.5).

## Contracts

[New contract](#)

Contract number	Contract date	Supplier	Note	Action
<a href="#">1</a>	2018-09-01 00:00:00	Petrov Pavlo Petrovych	Order 34 on 30.08.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">2</a>	2018-09-10 00:00:00	Petrov Pavlo Petrovych	Invoice 08-78 on 28.08.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">3</a>	2018-09-23 00:00:00	Ivanov Illia Illych	Order 56 on 28.08.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">4</a>	2018-09-24 00:00:00	Interfruit Ltd.	Order 74 on 11.09.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">5</a>	2018-10-02 00:00:00	Interfruit Ltd.	Invoice 09-12 on 21.09.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">7</a>	2018-12-27 13:30:04	Petrov Pavlo Petrovych		<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">13</a>	2019-01-10 13:20:48	Transservice LLC	Order #9876	<a href="#">Update</a> <a href="#">Delete</a>

Рисунок 9.4

[Back](#)

**Supplier**

Petrov Pavlo Petrovych ▼

**Note**

Save

Рисунок 9.5

Рядки 26 – 99 містять перевірку режимів створення нового запису (рисунок 9.5), оновлення (рисунок 9.6) або видалення існуючого запису (рисунок 9.7) та відображення відповідних форм з певними елементами інтерфейсу користувача.

```

26 <?php
27 # check for action parameter
28 # show create/update or delete form if it is set
29 if (isset($_GET["action"]) && ($_GET["action"] == "create" || $_GET["action"] == "update"
30 || $_GET["action"] == "delete")) {
31     ?>
32     <form method="post" action="index.php">
33         <input type="hidden" value="<?=$_GET["id"] ?>" name="contract_number" />
34         <?php
35         # if the current mode is create/update
36         # show corresponding form with the required fields and buttons
37         if ($_GET["action"] == "create" || $_GET["action"] == "update") {
38             ?>
39             <p>
40                 <b>Supplier</b>
41             </p>
42             <p>
43                 <select name="supplier_id">
44                     <?php
45                     # retrieve suppliers ids/info to display select control
46                     $sql = "SELECT * FROM supplier_info";
47                     $result = mysqli_query($conn, $sql);
48
49                     while ($row = mysqli_fetch_assoc($result)) {
50                         ?><option value="<?=$row["supplier_id"] ?>"><?=$row["Info"] ?></option><?php
51                     }
52                     ?>
53                 </select>
54             </p>
55
56             <p>
57                 <b>Note</b>
58             </p>
59             <p>
60                 <?php
61                 # retrieve and display contract note of the updated contract
62                 if (isset($_GET["action"]) && $_GET["action"] == "update") {
63                     $contract_number = $_GET["id"];
64
65                     $sql = "SELECT contract_note FROM contract WHERE contract_number = {$contract_number}";
66                     $result = mysqli_query($conn, $sql);
67                     $row = mysqli_fetch_assoc($result);
68                     ?>
69                     <textarea name="contract_note" rows="5" cols="50"><?=$row["contract_note"] ?></textarea>
70                 </p>
71             <p>
72                 <?php
73                 # set proper names for create/update buttons
74                 if (isset($_GET["action"]) && $_GET["action"] == "create") {
75                     ?>
76                     <input type="submit" name="create_contract" value="Save" />
77                 <?php
78                 } else if (isset($_GET["action"]) && $_GET["action"] == "update") {
79                     ?>
80                     <input type="submit" name="update_contract" value="Save" />
81                 <?php
82                 }
83                 ?>
84             </p>

```

```

85 <?php
86 # if the current mode is delete
87 # display the corresponding question and button
88 } else if ($_GET["action"] == "delete") {
89 ?>
90     <b>Delete the contract #<?= $_GET["id"] ?>?</b>
91     <p>
92         <input type="submit" name="delete_contract" value="Continue" />
93     </p>
94 <?php
95 }
96 ?>
97 </form>
98 <?php
99 } else {

```

### Supplier

Transservice LLC ▼

### Note

Order #9876

Save

Рисунок 9.6

[Back](#)

**Delete the contract #13?**

Continue

Рисунок 9.7

Рядки 100 – 133, в свою чергу, визначають таблицю з даними про договори та відповідними посиланнями (стовпчик Action), призначеними для маніпулювання цими даними (рисунок 9.4).

Рядки 135 – 179 містять визначення додаткової таблиці, призначеної для відображення переліку поставлених товарів за певним договором (рисунок 9.8). Для демонстрації даної таблиці виконується необхідна перевірка режиму перегляду даних про договори (рядки 137 – 138).

```

100  <?>
101  <table border="1">
102      <tr>
103          <th>Contract number</th>
104          <th>Contract date</th>
105          <th>Supplier</th>
106          <th>Note</th>
107          <th>Action</th>
108      </tr>
109  <?php
110      # retrieve and display data about contracts
111      $sql = "SELECT contract_supplier.*,
112             (SELECT contract_note FROM contract WHERE contract_number = contract_supplier.contract_number) AS `note`
113             FROM contract_supplier";
114      $result = mysqli_query($conn, $sql);
115
116      while ($row = mysqli_fetch_assoc($result)) {
117          <?>
118          <tr>
119              <td><a href="index.php?action=info&id=<?=$row["contract_number"] ?>"><?=$row["contract_number"] ?></a></td>
120              <td><?=$row["contract_date"] ?></td>
121              <td><?=$row["Supplier"] ?></td>
122              <td><?=$row["note"] ?></td>
123              <td>
124                  <a href="index.php?action=update&id=<?=$row["contract_number"] ?>">Update</a>
125                  <a href="index.php?action=delete&id=<?=$row["contract_number"] ?>">Delete</a>
126              </td>
127          </tr>
128      <?php
129      }
130  <?>
131  </table>
132  <?php
133  }

```

```

135  # if the action mode is info
136  # display data about supplied products for a selected contract
137  if (isset($_GET["action"]) && $_GET["action"] == "info") {
138      $contract_number = $_GET["id"];
139      <?>
140      <h3>Supplied products by contract #<?=$contract_number ?></h3>
141      <p>
142          <a href="index.php">Hide</a>
143      </p>
144      <?php
145          # retrieve data about selected products
146          $sql = "SELECT supplied_product, supplied_amount, supplied_cost
147                  FROM supplied
148                  WHERE contract_number = {$contract_number}";
149          $result = mysqli_query($conn, $sql);
150
151          # check the size of a result set
152          if (mysqli_num_rows($result) > 0) {
153              <?>
154              <table border="1">
155                  <tr>
156                      <th>Product</th>
157                      <th>Amount</th>
158                      <th>Cost</th>
159                  </tr>
160              <?php

```



```

161      # display products if the contract is not empty
162      while ($row = mysqli_fetch_assoc($result)) {
163          ?>
164          <tr>
165              <td><?=$row["supplied_product"] ?></td>
166              <td><?=$row["supplied_amount"] ?></td>
167              <td><?=$row["supplied_cost"] ?></td>
168          </tr>
169          <?php
170      }
171  } else {
172      # if the result set is empty print the following message
173      echo "Contract is empty";
174  }
175  ?>
176  </table>
177  <?php
178  }
179  ?>

```

Contract number	Contract date	Supplier	Note	Action
<a href="#">1</a>	2018-09-01 00:00:00	Petrov Pavlo Petrovych	Order 34 on 30.08.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">2</a>	2018-09-10 00:00:00	Petrov Pavlo Petrovych	Invoice 08-78 on 28.08.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">3</a>	2018-09-23 00:00:00	Ivanov Illia Illych	Order 56 on 28.08.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">4</a>	2018-09-24 00:00:00	Interfruit Ltd.	Order 74 on 11.09.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">5</a>	2018-10-02 00:00:00	Interfruit Ltd.	Invoice 09-12 on 21.09.2018	<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">7</a>	2018-12-27 13:30:04	Petrov Pavlo Petrovych		<a href="#">Update</a> <a href="#">Delete</a>
<a href="#">13</a>	2019-01-10 13:20:48	Transservice LLC	Order #9876	<a href="#">Update</a> <a href="#">Delete</a>

#### Supplied products by contract #4

[Hide](#)

Product	Amount	Cost
Audio Player	22	320.00
Printer	41	332.50
TV	56	990.00

Рисунок 9.8

#### 4. Розробити функціональність програмного забезпечення для роботи працівника складу

Сторінка, що містить функціональність програмного забезпечення для роботи працівника складу, міститься у файлі storekeeper.php.

Рядки 1 – 14 містять перевірку наявності користувацького сеансу, а також наявності змінної сеансу – масиву, до якого будуть записані товари, поставлені на склад, але ще не збережені у базі даних.

```

1 <?php
2 # check for a user session
3 if (!isset($_SESSION["user"])) {
4     header("location: login.php");
5 }
6
7 # initialize array of delivered but not stored products
8 # such array is implemented as the session variable
9 if (!isset($_SESSION["supplied_products"])) {
10     $_SESSION["supplied_products"] = array();
11 }
12 ?>
13
14 <h3>Supplied products</h3>

```

У рядках 16 – 72 визначається таблиця поставлених на склад товарів.

```

16 <?php
17 # check for awaiting deliveries (is there any empty contracts)
18 $sql = "SELECT * FROM contract_supplier
19     WHERE contract_number NOT IN (SELECT contract_number FROM supplied)";
20 $result = mysqli_query($conn, $sql);
21
22 # if awaiting deliveries exist
23 # display a corresponding form
24 if (mysqli_num_rows($result) > 0) {
25     # check session array of delivered but not stored products
26     # if there are any products - display the form used to store supplied products
27     if (sizeof($_SESSION["supplied_products"]) > 0) {
28         ?>
29         <form method="post" action="index.php">
30             <p>
31                 <b>by contract</b>
32                 <select name="contract_number">
33                     <?php
34                     # display the combo box with awaiting orders
35                     while ($row = mysqli_fetch_assoc($result)) {
36                         ?><option value="<?=$row["contract_number"] ?>">
37                             <?=$row["contract_number"] . " - " . $row["Supplier"] .
38                             " (" . $row["contract_date"] . ")" ?></option><?php
39                     }
40                     ?>
41                 </select>
42             </p>
43             <table border="1">
44                 <tr>
45                     <th>Product</th>
46                     <th>Amount</th>
47                     <th>Cost</th>
48                     <th>Action</th>
49                 </tr>

```

При цьому, виконується перевірка наявності товарів у масиві (змінна сеансу) та вивід форми (рисунок 9.9), що дозволяє записати прийняті товари у базу даних (рядки 27 – 67).

```

50      <?php
51      # display the session array of delivered products
52      foreach ($_SESSION["supplied_products"] as $key => $value) {
53      ?>
54      <tr>
55          <td><?= $key ?></td>
56          <td><?= $value["amount"] ?></td>
57          <td><?= $value["cost"] ?></td>
58          <td><a href="index.php?supplied=remove&product=<?= $key ?>">Remove</a></td>
59      </tr>
60      <?php
61      }
62      ?>
63      </table>
64      <p>
65          <input type="submit" name="save_products" value="Store products" />
66      </p>
67  </form>
68  <?php
69  } else {
70      echo "Add supplied products";
71  }
72  ?>

```

Також перевіряється наявність очікуваних поставок (якщо є так звані «пусті» договори, що були укладені, але за якими ще не були поставлені товари) у рядках 17 – 24. В разі наявності таких договорів, демонструється форма (рисунок 9.10) для додавання прийнятого товару (рядки 73 – 103).

```

73      <p>
74          <b>New product</b>
75      </p>
76      <form method="post" action="index.php">
77          <table border="1">
78              <tr>
79                  <th>Product</th>
80                  <th>Amount</th>
81                  <th>Cost</th>
82              </tr>
83              <tr>
84                  <td>
85                      <input type="text" name="supplied_product" required />
86                  </td>
87                  <td>
88                      <input type="number" name="supplied_amount" min="0.01" step="0.01" value="0.01" required />
89                  </td>
90                  <td>
91                      <input type="number" name="supplied_cost" min="0.01" step="0.01" value="0.01" required />
92                  </td>
93              </tr>
94          </table>
95          <p>
96              <input type="submit" name="add_product" value="Add product">
97          </p>
98      </form>
99      <?php
100      } else {
101          echo "There are no awaiting deliveries";
102      }
103      ?>

```

### Supplied products

by contract 13 - Transservice LLC (2019-01-10 13:20:48) ▼

Product	Amount	Cost	Action
TV	15	900	<a href="#">Remove</a>
Camera	30	1200	<a href="#">Remove</a>
Watch	200	399.99	<a href="#">Remove</a>

Store products

Рисунок 9.9

### New product

Product	Amount	Cost
Bluetooth Speaker	99	120

Add product

Рисунок 9.10

## 5. Розробити функціональність для формування звіту у форматі Excel щодо обсягів поставленої продукції за певний період

Реалізація даної функціональної можливості буде знаходитися також у файлі `action.php`, який містить обробку користувацьких форм.

Рядки 1 – 34 даного файлу містять обробки форм, які призначені для створення записів про договори, а також оновлення та видалення існуючих записів. Слід звернути увагу, що для виконання операцій створення, оновлення та видалення записів з таблиці `contract` використовується створена раніше збережена процедура `sp_contract_ops`.

У рядках 36 – 60 обробляються форми, які призначені для створення запису про поставлений, але ще не збережений у базі даних товар, а також видалення таких записів з масиву, що зберігається в якості змінної сеансу користувача.

```

1  <?php
2  # process request to create contract
3  if (isset($_POST["create_contract"])) {
4      $supplier_id = $_POST["supplier_id"];
5      $contract_note = $_POST["contract_note"];
6
7      # use the stored procedure created earlier
8      $sql = "CALL sp_contract_ops('i', 0, '', {$supplier_id}, '{$contract_note}')";
9      mysqli_query($conn, $sql);
10
11     header("location: index.php");
12 }
13
14 # process request to delete contract
15 if (isset($_POST["delete_contract"])) {
16     $contract_number = $_POST["contract_number"];
17
18     $sql = "CALL sp_contract_ops('d', {$contract_number}, '', 0, '')";
19     mysqli_query($conn, $sql);
20
21     header("location: index.php");
22 }
23
24 # process request to update contract
25 if (isset($_POST["update_contract"])) {
26     $contract_number = $_POST["contract_number"];
27     $supplier_id = $_POST["supplier_id"];
28     $contract_note = $_POST["contract_note"];
29
30     $sql = "CALL sp_contract_ops('u', {$contract_number}, CURRENT_TIMESTAMP(), {$supplier_id}, '{$contract_note}')";
31     mysqli_query($conn, $sql);
32
33     header("location: index.php");
34 }
35
36 # process request to insert new record into session array of delivered products
37 if (isset($_POST["add_product"])) {
38     $supplied_product = $_POST["supplied_product"];
39     $supplied_amount = $_POST["supplied_amount"];
40     $supplied_cost = $_POST["supplied_cost"];
41
42     if (!empty($supplied_product) && !empty($supplied_amount) && !empty($supplied_cost)) {
43         if (is_numeric($supplied_amount) && is_numeric($supplied_cost)) {
44             if ($supplied_amount > 0 && $supplied_cost > 0) {
45                 $_SESSION["supplied_products"][$supplied_product] = array("amount" => $supplied_amount,
46                                     "cost" => $supplied_cost);
47             }
48         }
49     }
50
51     header("location: index.php");
52 }
53
54 # process request to remove a record from the session array
55 if (isset($_GET["supplied"]) && $_GET["supplied"] == "remove") {
56     $supplied_product = $_GET["product"];
57     unset($_SESSION["supplied_products"][$supplied_product]);
58
59     header("location: index.php");
60 }

```

Рядки 62 – 103 демонструють збереження поставлених товарів до бази даних. Слід звернути увагу на те, що створення записів про товари, поставлені за певним договором, у таблиці `supplied`, виконується всередині транзакції, оскільки часткове, в силу будь-яких обставин, перенесення даних про прийняті товари з сеансової змінної до оперативної бази даних є неприйнятним.

```

62 # process request to store delivered products into the database
63 if (isset($_POST["save_products"])) {
64     $contract_number = $_POST["contract_number"];
65
66     # begin transaction
67     mysqli_query($conn, "SET AUTOCOMMIT = 0");
68     mysqli_query($conn, "START TRANSACTION");
69
70     $failed = false;
71
72     foreach ($_SESSION["supplied_products"] as $key => $value) {
73         $amount = $value["amount"];
74         $cost = $value["cost"];
75
76         # keep result of each query inside the transaction
77         $result = mysqli_query($conn, "INSERT INTO supplied (contract_number,
78             supplied_product, supplied_amount, supplied_cost) VALUES (
79             {$contract_number}, '{$key}', {$amount}, {$cost})");
80
81         if (!$result) {
82             $failed = true;
83
84             # rollback the transaction if any query is failed
85             mysqli_query($conn, "ROLLBACK");
86             break;
87         }
88     }
89
90     if (!$failed) {
91         # commit the transaction if there are no failed queries
92         mysqli_query($conn, "COMMIT");
93     }
94
95     # restore autocommit property
96     mysqli_query($conn, "SET AUTOCOMMIT = 1");
97
98     # clear session array after products are stored into the database
99     $_SESSION["supplied_products"] = NULL;
100
101     header("location: index.php");
102 }
103 ?>

```

Розглянутий код файлу action.php необхідно доповнити наступним фрагментом, що призначений для формування та збереження документу Excel зі звітом про обсяги поставленої продукції за певний період. Для формування звіту буде використовуватися раніше створена збережена процедура `sp_contract_total`.

Вміст файлу manager.php необхідно доповнити посиланням (рисунок 9.11), яке дозволить сформувати та завантажити звіт (рядок 21).

```

8      <h3>Contracts</h3>
9      <p>
10     <?php
11     # if the page is in record's create/update or delete mode (action parameter is set) - show 'back' link
12     if (isset($_GET["action"]) && ($_GET["action"] == "create" || $_GET["action"] == "update"
13         || $_GET["action"] == "delete")) {
14         ?>
15         <a href="index.php">Back</a>
16     <?php
17     # otherwise - show 'new record' link
18     } else {
19         ?>
20         <a href="index.php?action=create">New contract</a>
21         <a href="index.php?action=export">Export data</a>
22     <?php
23     }
24     ?>
25 </p>

```

Крім того, файл action.php необхідно доповнити кодом (рядки 104 – 127), призначеним безпосередньо для формування та завантаження звіту (рисунок 9.12).

```

104 # process request to export report into the Excel document
105 if (isset($_GET["action"]) && $_GET["action"] == "export") {
106     $filename = "report_contracts_" . date('Ymd') . ".xls";
107
108     header("Content-Disposition: attachment; filename=\"$filename\"");
109     header("Content-Type: application/vnd.ms-excel");
110
111     $flag = false;
112     $result = mysqli_query($conn, "CALL sp_contract_total('2018-01-01', CURRENT_TIMESTAMP())");
113
114     while ($row = mysqli_fetch_assoc($result)) {
115         if (!$flag) {
116             echo implode("\t", array_keys($row)) . "\r\n";
117             $flag = true;
118         }
119
120         array_walk($row, __NAMESPACE__ . '\cleanData');
121         echo implode("\t", array_values($row)) . "\r\n";
122     }
123
124     exit;
125 }
126
127 function cleanData(&$str) {
128     $str = preg_replace("/\t/", "\\t", $str);
129     $str = preg_replace("/\r?\n/", "\\n", $str);
130
131     if (strpos($str, "'")) {
132         $str = "'" . str_replace("'", "'", $str) . "'";
133     }
134 }
135 ?>

```

## Contracts

[New contract](#) [Export data](#)

Рисунок 9.11

A1		contract_number		
	A	B	C	D
1	contract_number	contract_date	SUM(supplied.supplied_amount)	SUM(supplied.supplied_amount * supplied.supplied_cost)
2	1	9/1/2018 0:00	47	39500
3	2	9/10/2018 0:00	24	11350
4	3	9/23/2018 0:00	148	99600
5	4	9/24/2018 0:00	119	76112.5
6	5	10/2/2018 0:00	64	45630
7	7	12/27/2018 13:30	15	59985
8	13	1/10/2019 13:20		

Рисунок 9.12

## 6. Оформити звіт з лабораторної роботи

У звіт включити основні етапи виконання лабораторної роботи та знімки екрану, що їх демонструють.

## 7. Питання для самоконтролю

1. Розробити функціональність програмного забезпечення для роботи адміністратора бази даних supply. Адміністратор повинен мати можливість створювати, оновлювати та видаляти записи в усіх таблицях бази даних.

2. Доповнити розглянутий фрагмент програмного забезпечення для роботи з БД supply можливістю сортувати рядки у таблиці Contracts (файл manager.php) в прямому та зворотному напрямках:

- за номером договору;
- за датою укладення договору.

3. Доповнити розглянутий фрагмент програмного забезпечення для роботи з БД supply можливістю сортувати рядки у таблиці Supplied products by contract #X (файл manager.php) в прямому та зворотному напрямках:

- за назвою поставленого товару;
- за кількістю одиниць поставленого товару;
- за вартістю одиниці поставленого товару.

4. Під час спроби оновлення даних про певний договір відкривається відповідна форма, що містить combo box зі списком постачальників. Внести зміни до програмного забезпечення таким чином, щоб у цьому combo box одразу після завантаження форми був обраний постачальник, з яким на даний момент укладений договір, що редагується.

5. Зважаючи на механізм посилкової цілісності, що використовується у базі даних supply, наразі неможливим є видалення даних про договір, за



яким були поставлені товари. Внести зміни до програмного забезпечення (наприклад, до збереженої процедури `sp_contract_ops`) для того, щоб дані про договір можна було видалити незважаючи на наявність даних про поставлені за цим договором товари.

6. Зважаючи на механізм посилкової цілісності, що використовується у базі даних `supply`, наразі неможливим є видалення даних про договір, за яким були поставлені товари. Внести зміни до програмного забезпечення таким чином, що для не «порожніх» договорів не буде доступною спроба їх видалення.

7. Як видно з рисунку 9.12, назви стовпців у згенерованому звіті не є зрозумілими для користувача, особливо це стосується стовпців, що містять агреговані дані. Внести зміни до програмного забезпечення таким чином, щоб вказати назви `Contract`, `Date`, `Total amount` та `Total cost` для відповідних стовпців.

8. Поточна реалізація дозволяє сформувати звіт (рисунок 9.12) на основі фіксованого діапазону дат – починаючи з 1 січня 2018 року до часу генерації звіту. Внести зміни до програмного забезпечення таким чином, щоб користувач мав змогу самостійно вказувати необхідний період для генерації звіту про обсяги поставленої продукції.

9. Надати менеджеру з закупівель можливість працювати з даними про постачальників (додавати записи, оновлювати та видаляти існуючі записи). Забезпечити можливість перегляду переліку договорів, укладених з певним постачальником.

10. Доповнити програмне забезпечення автоматичною генерацією прибуткової накладної відразу після того, як дані про товари, що надійшли від постачальника в рамках певного договору на поставку, були збережені працівником складу до бази даних.