

## Лабораторная работа 1

### СОЗДАНИЕ БАЗЫ ДАННЫХ СРЕДСТВАМИ ЯЗЫКА SQL

**Цель работы:** научиться создавать и связывать таблицы базы данных с использованием СУБД MySQL.

### Ход работы

#### 1. Установить MySQL

Для упрощения установки и дальнейшего использования системы управления базами данных (СУБД) MySQL рекомендуется установить один из свободно распространяемых WAMP (Windows, Apache, MySQL, PHP) или LAMP (Linux, Apache, MySQL, PHP) серверов, например OpenServer или XAMPP. В дальнейших примерах выполнения лабораторных работ будет использоваться сервер XAMPP.

После того, как сервер XAMPP будет установлено, необходимо запустить панель управления компонентами сервера, запустить MySQL и открыть командную строку сервера (рисунок 1.1).

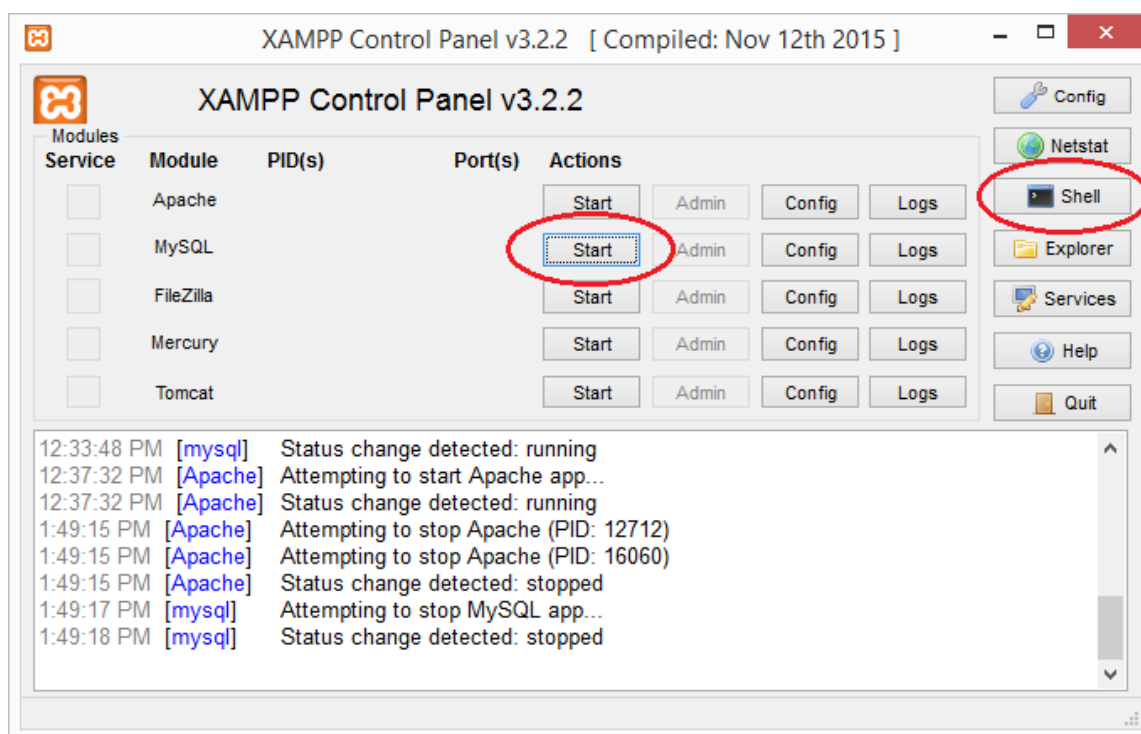


Рисунок 1.1

## 2. Подключиться к MySQL и создать базу данных

В командной строке сервера MySQL необходимо ввести команду:

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

После этого необходимо ввести пароль (рисунок 1.2). Обычно пароль пользователя root является пустым, поэтому можно просто нажать Enter.

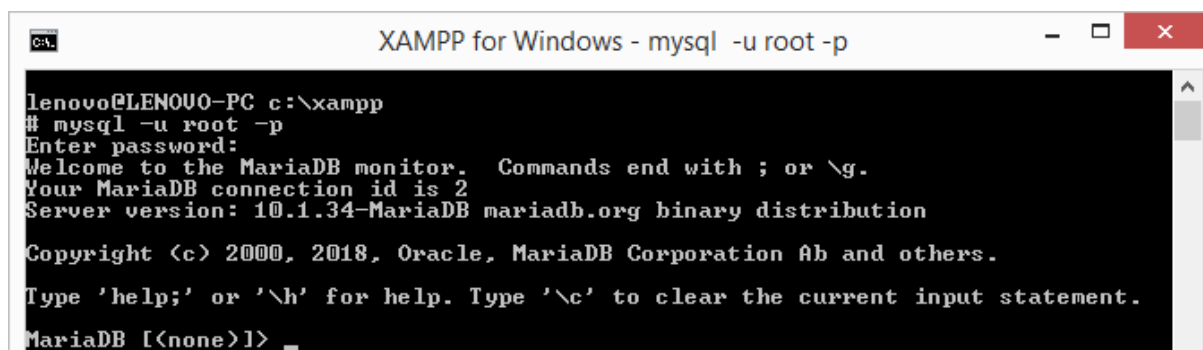


Рисунок 1.2

Для того, чтобы отключиться от MySQL необходимо использовать команду exit.

Основными командами, которые будут использоваться время от времени при работе с MySQL являются следующие:

- 1) USE database - выбрать базу данных (БД) для дальнейшей работы;
- 2) SHOW DATABASES - получить перечень баз данных;
- 3) SHOW TABLES - получить перечень таблиц для выбранной БД;
- 4) SHOW COLUMNS FROM table - получить информацию о таблице;
- 5) SHOW INDEX FROM table - получить информацию об индексах, определенных для таблицы.

Создание базы данных выполняется с помощью команды:

```
CREATE DATABASE supply;
```

Выполнение данной команды позволит создать базу данных, работа с которой будет рассматриваться в лабораторном практикуме. Проверить создания базы данных можно с помощью команды SHOW DATABASES.

### 3. Создать таблицы базы данных и связать их

Для изучения особенностей работы с СУБД MySQL будет рассматриваться база данных некоторого предприятия, приобретает товары у разных поставщиков. Приобретение товаров выполняется партиями и оформляется в виде договоров на поставку. Каждый договор имеет уникальный номер и заключается только с одним поставщиком. В документах по каждому договору указывается название, размер поставленной партии и цена (в грн.).

Создание таблиц выполняется с помощью оператора CREATE TABLE. Таким образом, для базы данных, рассматривается, необходимо создать следующие таблицы:

```
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;
```

Проверить созданы таблицы в базе данных supply (рисунок 1.3).

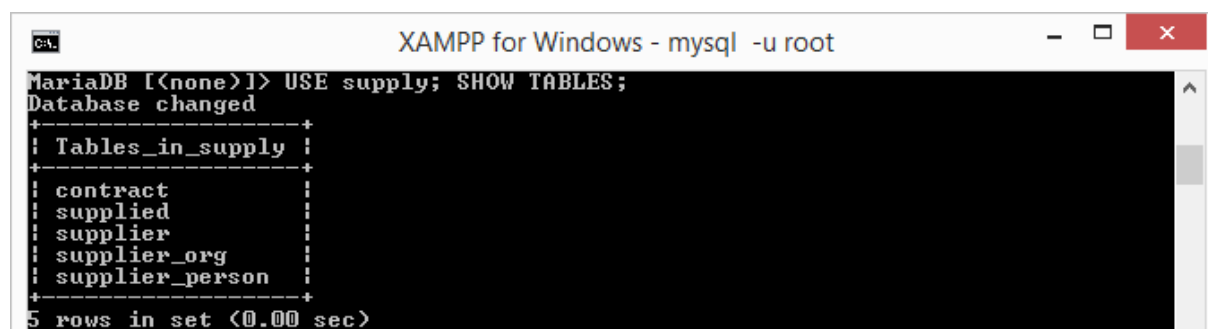
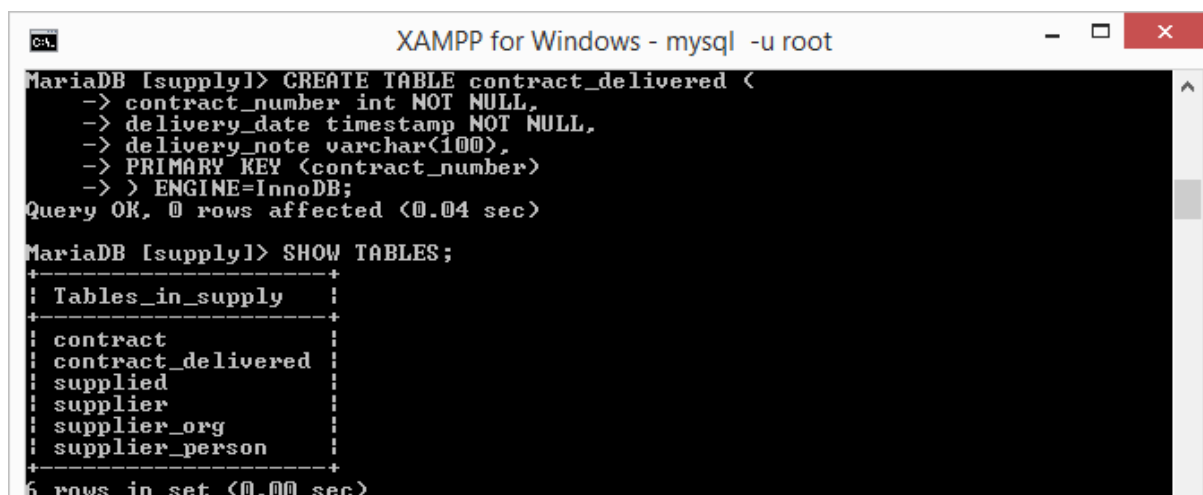


Рисунок 1.3

#### 4. Модификация структуры таблиц

Изменить структуру существующей таблицы можно с помощью оператора ALTER TABLE. Предположим, что необходимо создать еще одну таблицу в базе данных supply, которая будет предназначена для хранения данных о фактах выполнения поставок по договорам (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;
```



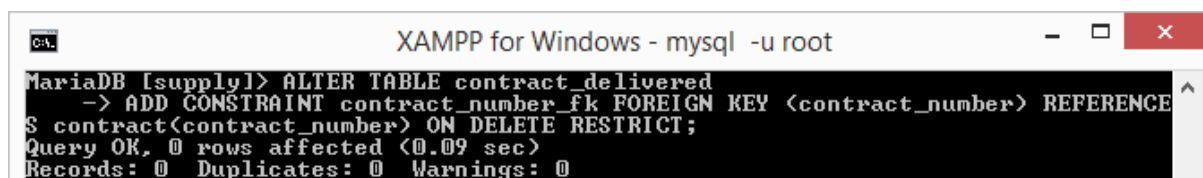
```
CA. XAMPP for Windows - mysql -u root
MariaDB [supply]> CREATE TABLE contract_delivered (
-> contract_number int NOT NULL,
-> delivery_date timestamp NOT NULL,
-> delivery_note varchar(100),
-> PRIMARY KEY (contract_number)
-> ) ENGINE=InnoDB;
Query OK, 0 rows affected (0.04 sec)

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, чтобы связать созданную таблицу `contract_delivered` таблице `contract` необходимо применить команду `ALTER TABLE` (рисунок 1.5).

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

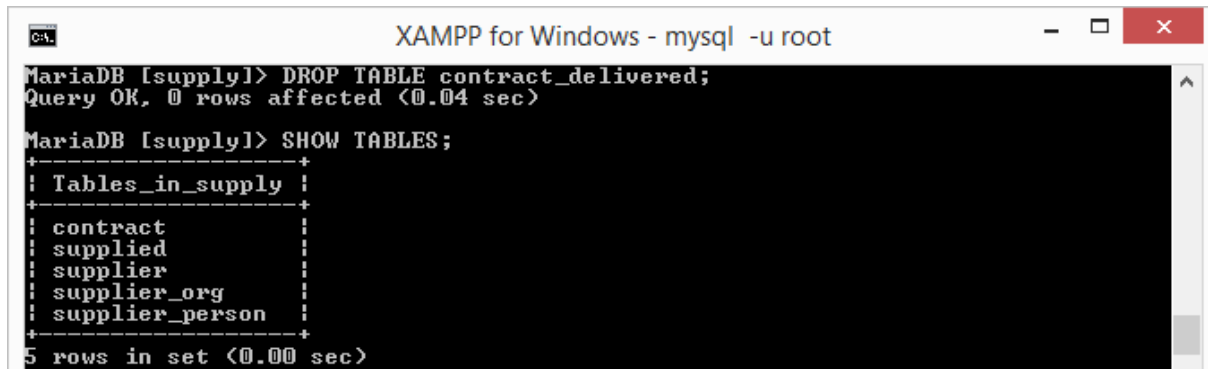


```
CA. XAMPP for Windows - mysql -u root
MariaDB [supply]> ALTER TABLE contract_delivered
-> ADD CONSTRAINT contract_number_fk FOREIGN KEY (contract_number) REFERENCE
S contract(contract_number) ON DELETE RESTRICT;
Query OK, 0 rows affected (0.09 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

Рисунок 1.5

## 5. Удаление таблиц

Удалить таблицу можно с помощью оператора `DROP TABLE`. Поскольку создана таблица `contract_delivered` не используется в дальнейшей работе, ее можно удалить с помощью данной команды (рисунок 1.6).



```

C:\XAMPP>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. Оформить отчет по лабораторной работе

В отчет включить основные этапы выполнения лабораторной работы и снимки экрана, которые демонстрируют.

## 7. Вопросы для самоконтроля

1. Как получить доступ к командной строке сервера MySQL?
2. Как установить соединение с сервером MySQL, используя имя и пароль определенного пользователя?
3. Назовите основные команды администрирования сервера БД MySQL и их назначения.
4. С помощью какой команды создается база данных? Как можно проверить создания БД?
5. С помощью каких операторов языка SQL выполняется создание и связывание таблиц?
6. С помощью какого оператора языка SQL выполняется модификация структуры таблицы?
7. С помощью какого оператора языка SQL выполняется удаление таблицы из базы данных?
8. Каким образом можно проверить наличие или отсутствие созданных или удаленных таблиц соответственно?
9. Каким образом можно задать имя внешнего ключа во время связывания таблиц?
10. Какие недостатки содержит рассмотренная структура БД? Как их устранить?

## Лабораторная работа 2

### МАНИПУЛИРОВАНИЕ ДАННЫМИ СРЕДСТВАМИ ЯЗЫКА SQL:

#### СЛОЖЕНИЕ, ОБНОВЛЕНИЯ И УДАЛЕНИЯ ДАННЫХ

**Цель работы:** научиться использовать операторы языка SQL, предназначенные для добавления, обновления и удаления данных на примере СУБД MySQL.

### Ход работы

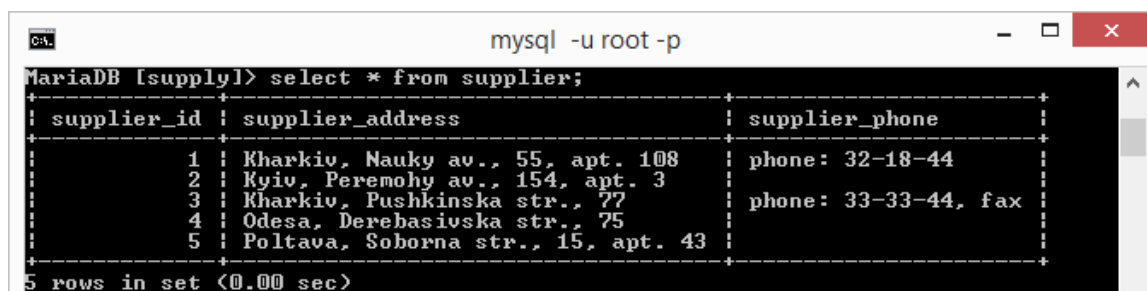
#### 1. Добавление данных в созданной базе данных

Для добавления данных используется оператор INSERT.

Следующие команды позволяют заполнить данные о поставщиках в созданной базе данных:

```
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', '');
```

Проверить записи, созданные в таблице 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

Следующие команды позволяют заполнить данные о поставщиках-физических лицах в созданной базе данных:

```
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');
```

Проверить записи, созданные в таблице `supplier_person` (рисунок 2.2).

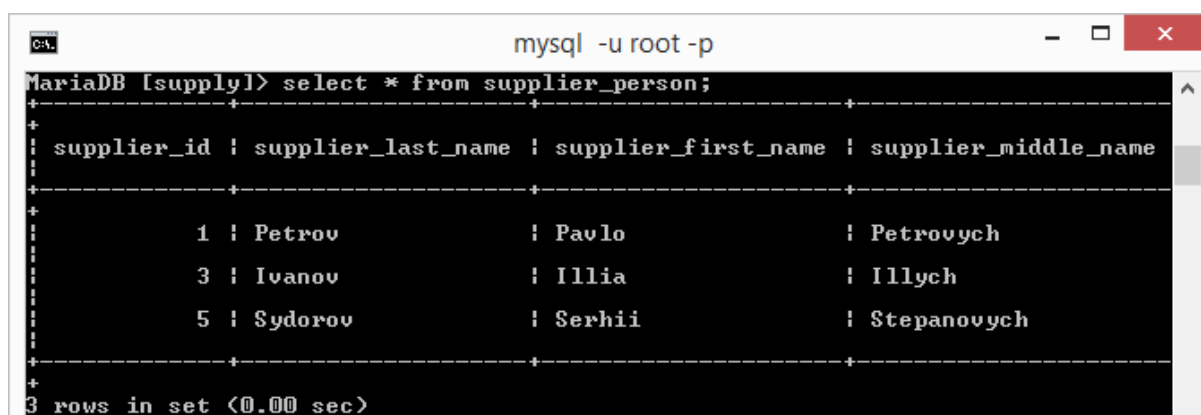


Рисунок 2.2

Следующие команды позволяют заполнить данные о поставщиках-юридических лицах в созданной базе данных:

```
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');
```

Проверить записи, созданные в таблице `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

Следующие команды позволяют заполнить данные о заключенных договорах в созданной базе данных:

```
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');
```

Проверить записи, созданные в таблице 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

Следующие команды позволяют заполнить данные о поставленных товарах в созданной базе данных:

```
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);
```

Проверить записи, созданные в таблице 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. Обновление данных в базе данных

Обновление данных (изменение значений полей в существующих записях) в БД выполняется с помощью оператора UPDATE.

Например, если необходимо уменьшить стоимость принтера который был поставлен по договору с номером 4 на 5%, команда будет следующей (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)

```

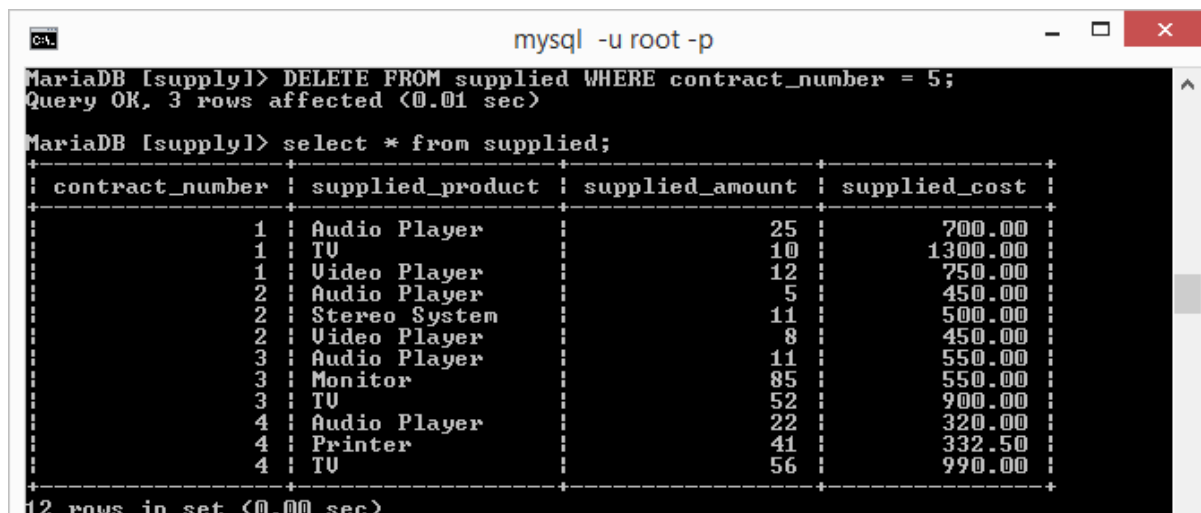
Рисунок 2.6

## 3. Удаление данных из базы данных

Для удаления данных из таблиц базы данных применяется оператор DELETE.

Например, для удаления поставленных товаров, соответствующих договора с номером 5, необходимо выполнить следующую 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

Восстановите удаленные записи с помощью команд INSERT.

#### 4. Оформить отчет по лабораторной работе

В отчет включить основные этапы выполнения лабораторной работы и снимки экрана, которые демонстрируют.

#### 5. Вопросы для самоконтроля

1. Приведите структуру и примеры использования команды INSERT.
2. Приведите структуру и примеры использования команды UPDATE.
3. Приведите структуру и примеры использования команды DELETE.
4. Каким образом можно обновить все записи в таблице БД?
5. Каким образом можно удалить все записи из таблицы БД?
6. Каким образом можно удалить последние 20 заключенных договоров?

7. Каким образом можно увеличить цену на 15% для 5 самых дешевых товаров, которые были поставлены по определенному договору?

8. Какой должна быть структура команды INSERT для того, чтобы новые записи с дублирующим ключом отвергались без генерации ошибки?

## Лабораторная работа 3

### МАНИПУЛИРОВАНИЕ ДАННЫМИ СРЕДСТВАМИ ЯЗЫКА SQL: ПРЕДЛОЖЕНИЯ SELECT И ИХ ОСНОВНЫЕ ОСОБЕННОСТИ

**Цель работы:** научиться использовать оператор SELECT языка SQL, предназначенный для выборки данных, на примере СУБД MySQL.

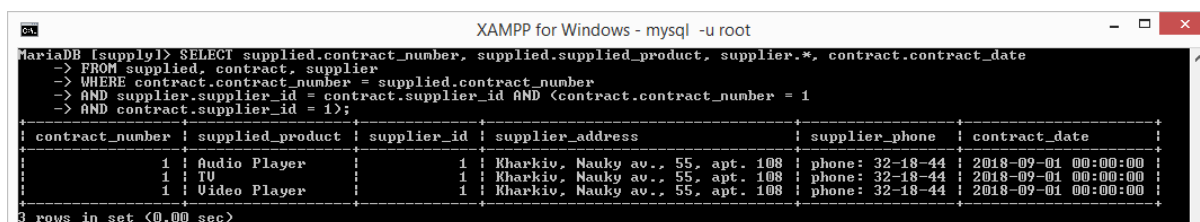
#### Ход работы

##### 1. Создать и выполнить запросы SQL SELECT

###### Запрос 1

Сформировать список товаров, поставленных поставщиком 1 (Петров П.П.) по договору 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);
```



XAMPP for Windows - mysql -u root

```
MariaDB [supply]> 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	TV	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

###### Запрос 2

Сформировать список товаров, которые были поставлены поставщиком 1 (Петров П.П.) в период с 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

### Запрос 3

Сформировать список товаров, которые были поставлены в 9 месяцев 2018 году с выводом названия поставщика и даты 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

### Запрос 4

Сформировать список договоров (номер, дата, название) и общую сумму по каждому договору (размер партии умножить на цену за единицу и просуммировать по договору). Список должен быть отсортирован по возрастанию номеров договоров (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

### Запрос 5

Сформировать список договоров (номер, дата, название) и общую сумму по каждому договору (размер партии умножить на цену за единицу и просуммировать по договору). Список должен быть отсортирован по возрастанию общей суммы по каждому договору. После этого на список должна быть наложена условие фильтрации, которая заключается в исключении из результата запроса записей, для которых номер договора меньше 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



### Запрос 6

Вывести информацию о самой по размеру партии товара во всех договорах с указанием поставщика, а также номера и даты договора (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);
```

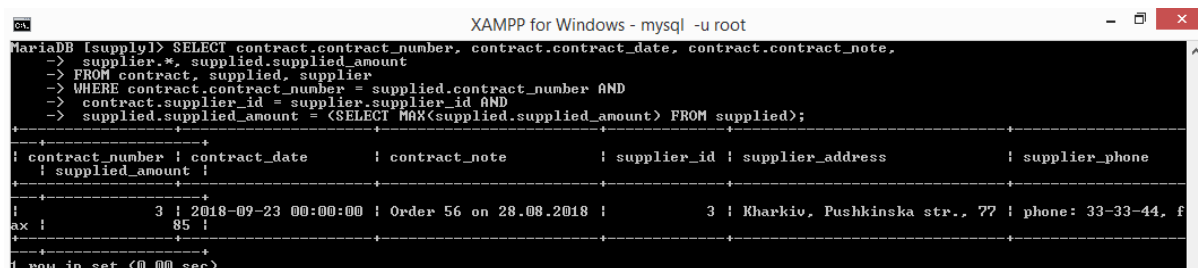


Рисунок 3.6

### Запрос 7

Сформировать список поставщиков (название и код), с которыми не было заключено ни одного договора (рисунок 3.7).

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



Рисунок 3.7

### Запрос 8

Сформировать список названий поставленных товаров с указанием средней цены поставки за единицу (независимо от поставщика) (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
TU	1012.500000
Video Player	683.333333

6 rows in set (0.00 sec)

Рисунок 3.8

### Запрос 9

Сформировать список товаров (название, количество и цена, поставщик), для которых цена за единицу больше средней (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
TU	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
TU	56	290.00	2	Kyiv, Peremohy av., 154, apt. 3	
TU	14	860.00	2	Kyiv, Peremohy av., 154, apt. 3	
Video Player	17	850.00	2	Kyiv, Peremohy av., 154, apt. 3	
TU	52	900.00	3	Kharkiv, Pushkinska str., 77	phone: 33-33-44, fax

7 rows in set (0.00 sec)

Рисунок 3.9

### Запрос 10

Вывести информацию о пяти самых дорогих товаров (название, цена за единицу, поставщик) (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 1;
```

```

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

### Запрос 11

Сформировать список поставщиков с указанием кода, адреса и данных поставщика. При формировании данных поставщика для физических лиц вывести фамилию и инициалы, а для юридических лиц - название(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

### Запрос 12

Сформировать список договоров (с указанием номера, даты поставки и данных о поставщике), общее количество поставленных товаров и общую сумму по каждому договору. При формировании данных поставщика для физических лиц вывести фамилию и инициалы, а для юридических лиц - название. В результате должны быть включены только те договоры, на основании которых товары действительно поставлялись (то есть в результат запроса не должны попасть так называемые «пустые» договоры) (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(supplier.supplied_amount) AS `Size`,
       SUM(supplier.supplied_cost * supplier.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;

```

The screenshot shows a terminal window titled "XAMPP for Windows - mysql -u root". The user has entered a SQL query into the MariaDB prompt. The query is identical to the one shown in the previous block. The output of the query is displayed as a table with 5 columns: contract\_number, contract\_date, Supplier, Size, and Total. There are 3 rows of data.

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

### Запрос 13

Сформировать список товаров (с указанием номера договора и даты поставки), поставленных поставщиками 1 (Петров П.П.) и 2 («Интерфрут») (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;

```

```

C:\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 | TV | 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 | TV | 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 | TV | 2 |
| 5 | 2018-10-02 00:00:00 | Video Player | 2 |
+-----+-----+-----+-----+
12 rows in set (0.00 sec)

```

Рисунок 3.13

#### Запрос 14

Сформировать номенклатуру товаров (то есть список названий товаров), которые поставлялись только поставщиком 1 (Петров П.П.), или только поставщиком 2 («Интерфрут»), или и поставщиком 1, и поставщиком 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;

```

```

C:\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    |
| TV               |
| Video Player     |
+-----+
5 rows in set (0.00 sec)

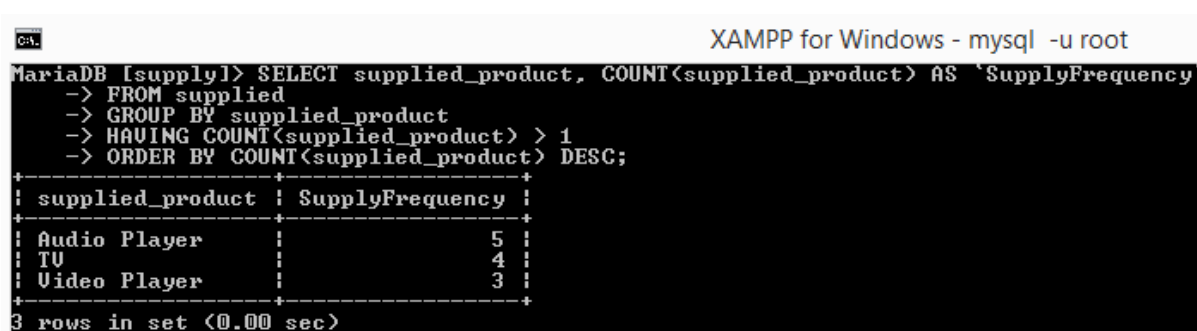
```

Рисунок 3.14

### Запрос 15

Сформировать список товаров, который должен отражать частоту поставок. В список включить только товары, которые поставлялись более одного раза. Список должен быть отсортирован по убыванию частоты 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 user is in the MariaDB [supply] database. The query entered is: `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 shows a table with two columns: `supplied_product` and `SupplyFrequency`. The results are: `Audio Player` with frequency 5, `TU` with frequency 4, and `Video Player` with frequency 3. The terminal also shows "3 rows in set (0.00 sec)".

supplied_product	SupplyFrequency
Audio Player	5
TU	4
Video Player	3

Рисунок 3.15

### Запрос 16

Сформировать данные о количественной динамике поставок товаров в течение 2018 года. Данные должны быть агрегированные по месяцам и представлены в виде таблицы, строками которой являются названия товаров, а столбцами - номера месяцев 2018 года. На пересечении строки и столбца должна отображаться количество данного товара, поставленного в данном месяце (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

### Запрос 17

Сформировать список поставленных товаров. Для каждого товара в этом списке должны быть указаны следующие данные: номер договора, наименование товара, количество единиц, цена за единицу, дата поставки, название месяца и номер года (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
2	Audio Player	5	450.00	2018-09-10 00:00:00	September	2018
2	Stereo System	11	500.00	2018-09-10 00:00:00	September	2018
2	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
5	Audio Player	33	580.00	2018-10-02 00:00:00	October	2018
5	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. Оформить отчет по лабораторной работе**

В отчет включить основные этапы выполнения лабораторной работы и снимки экрана, которые демонстрируют.

## **3. Вопросы для самоконтроля**

1. Какой оператор языка SQL используется для выбора данных из одной или нескольких таблиц?
2. Привести общую структуру оператора SELECT.
3. Как можно записать SQL SELECT запрос, если необходимо вывести все столбцы таблицы?
4. Какая конструкция используется для выбора записей, удовлетворяющих критериям поиска?
5. Какое ключевое слово используется для исключения повторений?
6. Какая конструкция используется для сортировки значений по одному или нескольким столбцам?
7. Каким образом реализуется обратный порядок сортировки?
8. С помощью какого ключевого слова выполняется ограничения выборки?
9. С помощью какой конструкции выполняется группировка строк, изымаются?
10. Назвать функции агрегации, их назначение и основные особенности.
11. Каким образом столбцу можно назначить новое имя?
12. Назовите назначение и отличие ключевого слова HAVING от WHERE?
13. Назвать основные арифметические, логические и операторы сравнения, их назначение и примеры использования.
14. Назначение функции MONTH и примеры ее использования.
15. Назначение функции YEAR и примеры ее использования.
16. Назначение функции IFNULL и примеры ее использования.
17. Назначение функции CONCAT и примеры ее использования.
18. Назначение функции RTRIM и примеры ее использования.
19. Назначение функции SUBSTRING и примеры ее использования.
20. Назначение функции IF и примеры ее использования.



21. Какой оператор используется для объединения результатов двух запросов.

## Лабораторная работа 4

### СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ ПРЕДСТАВЛЕНИЙ

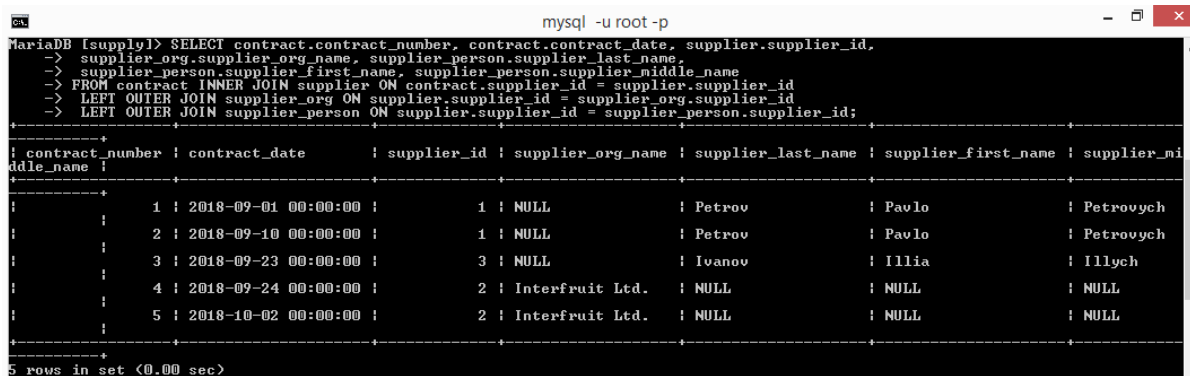
**Цель работы:** научиться создавать и применять представления (view) в базе данных на примере СУБД MySQL.

#### Ход работы

##### 1. Создать представление, позволяющее при просмотре списка договоров видеть название поставщика

Создание представлений осуществляется с помощью оператора CREATE VIEW. Таким образом, создать представление, которое позволит просматривать список договоров с указанием названия поставщика, можно на основе следующего запроса (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 terminal window with the MySQL command prompt. The query is executed, and the results are displayed in a table format. The table has 7 columns: contract\_number, contract\_date, supplier\_id, supplier\_org\_name, supplier\_last\_name, supplier\_first\_name, and supplier\_middle\_name. There are 5 rows of data, each representing a contract with its date, supplier ID, and the supplier's name (last, first, and middle names).

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 запроса имеет определенный недостаток - данные поставщиков - юридических и физических лиц находятся в разных столбцах, а также присутствуют значения NULL. Этот недостаток можно исправить с помощью следующего запроса (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

Теперь можно создать данное представление под названием `contract_supplier` с помощью соответствующей команды языка 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)

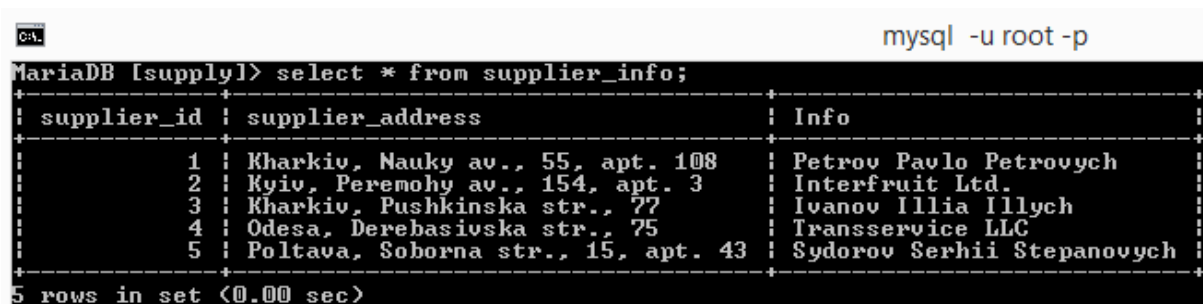
Рисунок 4.3

## 2. Создать представление, позволяет пользователю работать с ограниченными данными о поставщиках

Предположим, что для определенных пользователей должна быть недоступна вся общая информация о поставщиках (сохранена в таблице

supplier), а только информация о коде и адрес поставщика. При этом пользователь должен иметь возможность видеть данные поставщика как субъекта предпринимательской деятельности (для юридических лиц - название, для физических - фамилия, имя, отчество) (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 случае возникновения необходимости, удалить представления можно с помощью оператора DROP VIEW.

### 3. Оформить отчет по лабораторной работе

В отчет включить основные этапы выполнения лабораторной работы и снимки экрана, которые демонстрируют.

### 4. Вопросы для самоконтроля

1. Что такое представление?
2. Назвать преимущества и недостатки представлений.
3. Какой оператор языка SQL используется для создания представлений?
4. Какой оператор языка SQL используется для удаления представлений?
5. Каким образом можно проверить наличие представления в базе данных?
6. Как указать список столбцов при создании представления?
7. Что такое вертикальное представление?

8. Что такое горизонтальное представление?

## Лабораторная работа 5

### СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ ХРАНИМЫХ ПРОЦЕДУР И ТРИГГЕРА

**Цель работы:** научиться создавать и применять программные объекты базы данных - хранимые процедуры и триггеры, на примере СУБД MySQL.

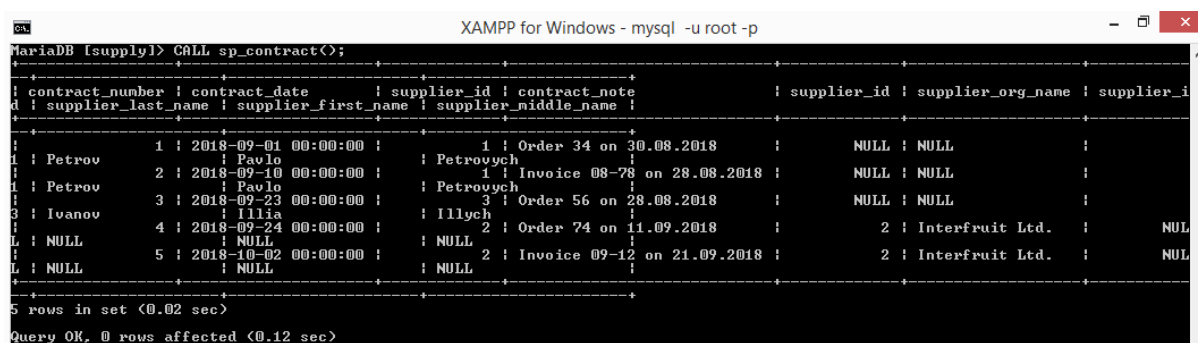
#### Ход работы

##### 1. Создание и использование хранимых процедур

Создание хранимых процедур реализуется оператором CREATE PROCEDURE. Таким образом, создать хранимую процедуру, которая реализует выборку данных из таблиц contract, supplier\_org, supplier\_person, можно с помощью следующей команды (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 //
```

Вызов процедуры осуществляется с помощью 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)

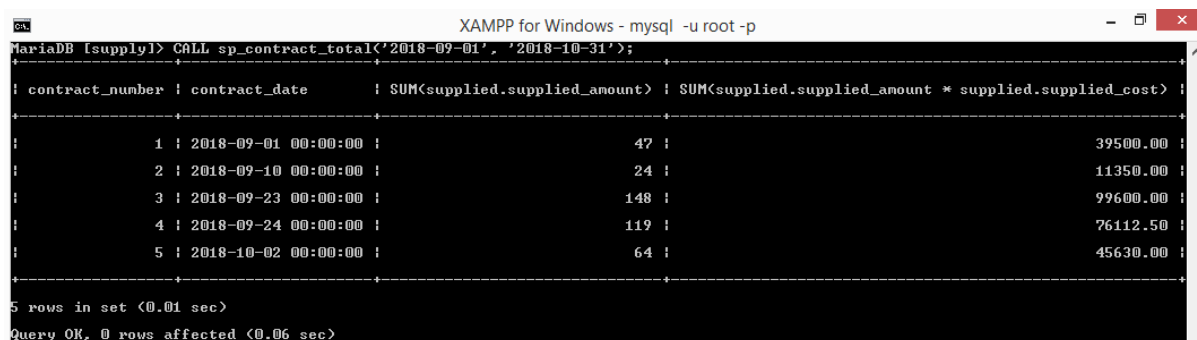
Рисунок 5.1

Для знакомства с особенностями создания и использования процедур с параметрами, необходимо создать хранимую процедуру, которая обеспечивает формирование агрегатных данных по поставкам для указанного интервала календарных 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 //
```

Позвонить созданной процедуры можно с помощью следующего запроса.

```
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

Рисунок 5.2

Следующая хранимая процедура предназначена для выполнения различных операций модификации данных для таблицы contract. Данная процедура использует оператор условия IF, предназначенный для управления потоком данных.

```

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 //

```

Следующий запрос позволяет создавать договор (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

Следующий запрос позволяет модифицировать договор (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

Следующий запрос позволяет удалять договор (рисунок 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 [supply1] > CALL sp_contract_ops('d', 6, '2018-12-31', 0, '');
Query OK, 1 row affected (0.01 sec)

MariaDB [supply1] > 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. Создание и использование триггеров

Предположим, что при вводе данных в таблицу `contract`, в которой хранится информация о договорах на поставку продукции, поле `contract_date`, в котором хранится дата заключения договора, должно быть обязательно заполнено. Причем в случае, если при вводе нового договора данное поле остается незаполненным, у него должна быть автоматически записана дата. Данную задачу можно решить с помощью создания определенного триггера, используя соответствующую команду `CREATE TRIGGER` (рисунок 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 проверки работы триггера необходимо добавить новый договор с помощью следующего запроса.

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

```

C:\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 базе данных хранится как общая информация о поставщиках, так и информация, которая относится только к физическим или юридическим лицам. Одновременное наличие данных о поставщике в таблицах `supplier_org` и `supplier_person` не допускается с точки зрения логики управления бизнесом. Таким образом, возникает необходимость сложного контроля отношений посылочной целостности. Для решения данной задачи создадим триггер, который при вводе информации в таблицу `supplier_person` будет контролировать наличие кода соответствующего поставщика в таблице `supplier_org` и блокировать ввод данных о поставщике как о физическом лице в том случае, если уже имеющиеся данные о данном поставщике как о юридическом лице (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 проверки работы триггера необходимо попытаться добавить данные о поставщике 2 (который уже хранится в БД в качестве юридического лица) как о физическом лице.

```

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 удаления хранимых процедур и триггеров необходимо воспользоваться операторами DROP PROCEDURE и DROP TRIGGER соответственно.

### 3. Оформить отчет по лабораторной работе

В отчет включить основные этапы выполнения лабораторной работы и снимки экрана, которые демонстрируют.

### 4. Вопросы для самоконтроля

1. Что такое хранимая процедура?
2. Назвать преимущества использования хранимых процедур.
3. Какой оператор используется для создания хранимой процедуры?
4. Каким образом можно определить входные или выходные параметры хранимой процедуры?
5. Для чего используется оператор IF?
6. Каково назначение операторов BEGIN и END?
7. Что такое триггер?
8. Назвать преимущества использования триггеров.
9. С помощью какого оператора триггер связывается с таблицей?
10. К каким событиям, связанных с изменением содержания таблицы, можно привязать триггер?
11. Каким образом можно определить до или после операции изменения содержимого таблицы должен срабатывать триггер?
12. Для чего используются префиксы NEW и OLD?
13. Каково назначение оператора SET?
14. С помощью каких операторов выполняется удаление процедур и триггеров?

## Лабораторная работа 6

### ОСНОВЫ ИСПОЛЬЗОВАНИЯ СРЕДСТВ КОНТРОЛЯ ЦЕЛОСТНОСТИ ДАННЫХ

**Цель работы:** изучить основы работы со средствами контроля посылочной целостности данных на примере СУБД MySQL.

#### Ход работы

**Внимание!** Прежде чем перейти к выполнению лабораторной работы, необходимо создать временную базу данных, используя запросы, использованные в лабораторной работе 2. Во всех последующих пунктах данной лабораторной работы предполагается использование временной базы данных.

#### 1. Изучить особенности работы механизма посылочной целостности NO ACTION

Особенности работы механизма посылочной целостности NO ACTION рассмотрим на примере отношений между таблицами supplier и contract, supplier и supplier\_person, supplier и supplier\_org. Данные таблицы связаны между собой по полю supplier\_id. В этой связи таблица supplier является родительской, а таблицы contract, supplier\_org, supplier\_person - дочерними. Для изучения особенностей работы механизма посылочной целостности необходимо выполнить следующую последовательность действий.

Установить параметры ON DELETE и ON UPDATE, определяющие поведение при удалении и обновления записей из таблицы-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;
```

Предположим, что в силу определенных причин необходимо удалили поставщика с кодом 4 (рисунок 6.1).

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

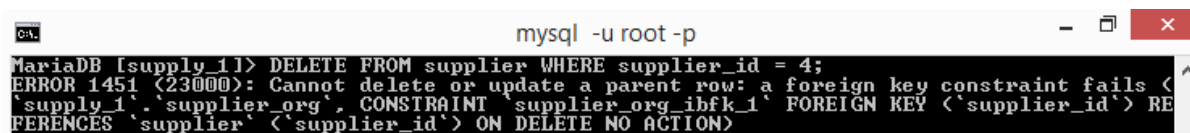


Рисунок 6.1

Таким образом, для того, чтобы удалить данного поставщика, необходимо предварительно удалить все связанные с ним данные. Для этого нужно удалить соответствующую запись из таблицы `supplier_org` и проверить наличие договоров с данным поставщиком в таблице `contract`. Если такие договоры есть, их также нужно удалить (при этом необходимо иметь в виду, что может потребоваться удаление и содержания данных договоров). После этого необходимо попытаться удалить поставщика с кодом 4 снова. Если связанных с ним данных нет, поставщик будет удален.

Предположим, что в силу определенных причин возникла необходимость для поставщика с кодом 5 изменить код на 7 (рисунок 6.2).

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

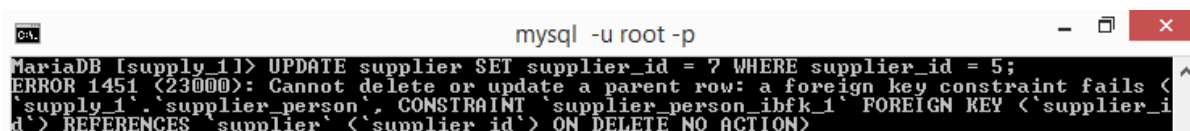


Рисунок 6.2

Поскольку договоры с данным поставщиком отсутствуют, ссылки на него есть только в таблице `supplier_person`. Удалив эту запись, необходимо повторить смену кода поставщика с 5 на 7. Теперь эта операция должна пройти успешно. После этого необходимо проверить содержимое таблиц.

## 2. Изучить особенности работы механизма каскадной целостности CASCADE

Изменим механизмы посылочной целостности для связей между всеми рассмотренными выше таблицами 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;

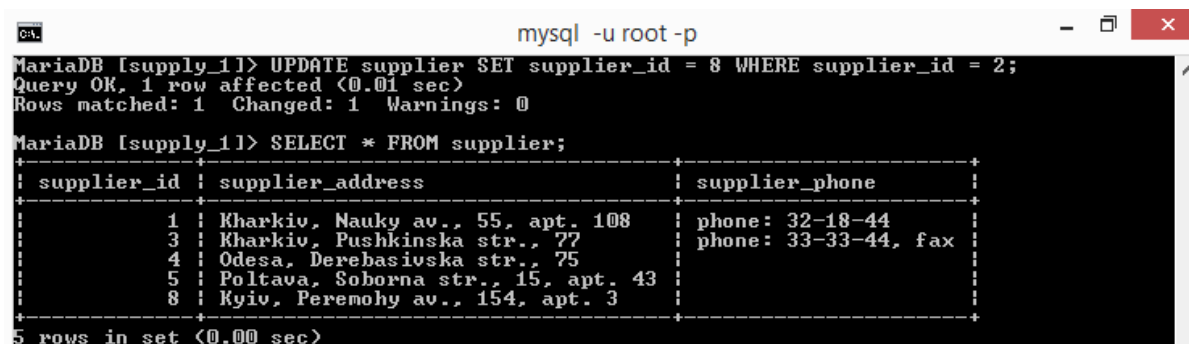
ALTER TABLE supplier_org
ADD CONSTRAINT supplier_org_ibfk_1 FOREIGN KEY (supplier_id) REFERENCES supplier(supplier_id) ON DELETE CASCADE ON UPDATE CASCADE;

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;
```

Предположим, что в силу определенных причин возникла необходимость для поставщика с кодом 2 изменить 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	

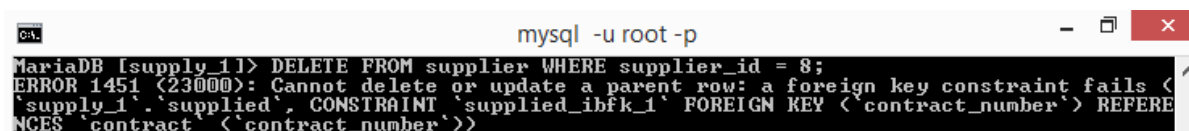
```
5 rows in set (0.00 sec)
```

Рисунок 6.3

Проверить наличие соответствующих изменений в таблице supplier\_org.

Теперь предположим, что данного поставщика (который сейчас имеет код 8) надо удалить (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'> REFERE
NCES 'contract' (<'contract_number'>))
```

Рисунок 6.4

Определить причину, по которой записи не были удалены. Внести необходимые изменения в механизмы посылочной целостности необходимых таблиц для того, чтобы необходимые данные все же были удалены.

### 3. Изучить особенности работы механизма посылочной целостности SET NULL

Особенности механизма посылочной целостности SET NULL рассмотрим на примере таблиц supplier и contract.

Изменим механизмы посылочной целостности для связей между всеми рассмотренными выше таблицами на SET NULL.

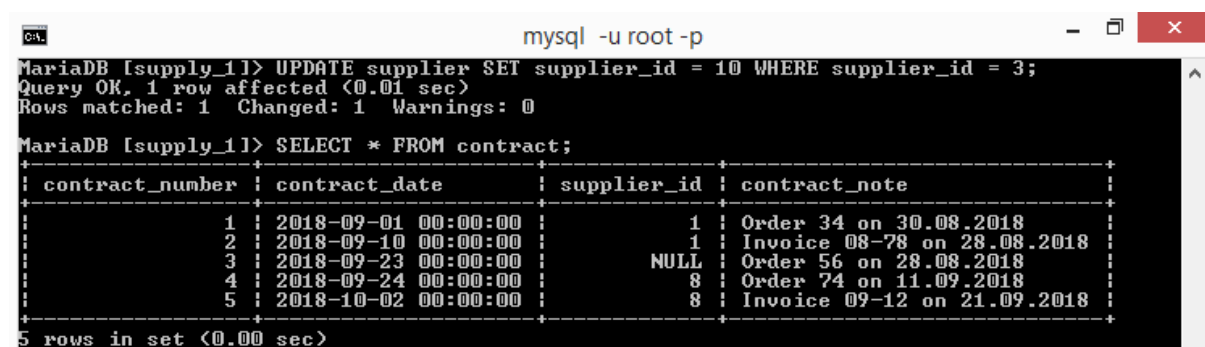
```
ALTER TABLE contract
DROP FOREIGN KEY contract_ibfk_1;

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 таблице supplier изменить код поставщика 3 на 10. Проверить данные в таблице contract (рисунок 6.5).

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



The screenshot shows a MySQL terminal window with the title 'mysql -u root -p'. The user is in the 'supply\_11' database. The first command executed is 'UPDATE supplier SET supplier\_id = 10 WHERE supplier\_id = 3;', which returns 'Query OK, 1 row affected (0.01 sec)' and 'Rows matched: 1 Changed: 1 Warnings: 0'. The second command is 'SELECT \* FROM contract;', which returns a table with 5 rows. The table has columns: contract\_number, contract\_date, supplier\_id, and contract\_note. The data is as follows:

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

The terminal also shows '5 rows in set (0.00 sec)' at the bottom.

Рисунок 6.5

Вместо NULL установить значение кода поставщика 10 для договора с номером 3.

#### **4. Оформить отчет по лабораторной работе**

В отчет включить основные этапы выполнения лабораторной работы и снимки экрана, которые демонстрируют.

#### **5. Вопросы для самоконтроля**

1. Есть конструкции ON DELETE и ON UPDATE обязательными при формировании команды CREATE TABLE или ALTER TABLE?

2. Какое поведение СУБД задает конструкция ON DELETE?

3. Какое поведение СУБД задает конструкция ON UPDATE?

4. Какие параметры можно установить после конструкций ON DELETE и ON UPDATE?

5. Назвать особенности механизма послочной целостности CASCADE.

6. Назвать особенности механизма послочной целостности SET NULL.

7. Назвать особенности механизма послочной целостности NO ACTION.

8. Назвать особенности механизма послочной целостности SET DEFAULT.

9. Назвать особенности механизма послочной целостности RESTRICT.

10. Почему в данной лабораторной работе не была рассмотрена работа с механизмом послочной целостности SET DEFAULT?

11. Каким образом можно установить тот или иной механизм послочной целостности для внешнего ключа таблицы?

12. Зачем перед тем, как установить механизм SET NULL, была выполнена модификация поля supplier\_id таблицы contract?

13. В каких случаях не рекомендуется использование механизма послочной целостности CASCADE?

14. Какой механизм послочной целостности всегда используется по умолчанию в СУБД MySQL в случае, если конструкции ON DELETE и ON UPDATE не определены?



## Лабораторная работа 7

### РАБОТА С ТРАНЗАКЦИЯМИ

**Цель работы:** изучить основы работы с механизмом транзакций в примере СУБД MySQL.

#### Ход работы

**Внимание!** Прежде чем перейти к выполнению лабораторной работы, необходимо создать временную базу данных, используя запросы, использованные в лабораторной работе 2. Во всех последующих пунктах данной лабораторной работы предполагается использование временной базы данных.

#### 1. Создать запрос, иллюстрирующий работу механизма транзакций при добавлении данных в одну таблицу

Рассмотрим последовательность действий при создании и использовании запроса, с помощью которого запускается транзакция, в таблицу `supplied` добавляется новая запись, а затем имитируется ситуация некорректного или корректного завершения транзакции. Состояние таблицы контролируется до начала транзакции, во время выполнения транзакции и после ее завершения. Для этого необходимо выполнить следующую последовательность действий.

```
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;
```

Запросы SELECT позволяют вывести данные, иллюстрирующие состояние таблицы до начала транзакции (рисунок 7.1), в процессе выполнения транзакции и после завершения транзакции.

```

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

Как видно из приведенных данных, новая запись в таблице появляется (рисунок 7.2), а затем исчезает (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

Теперь необходимо рассмотреть ситуацию корректного завершения транзакции. Для этого в приведенном тексте запроса необходимо изменить оператор ROLLBACK на COMMIT. Выполнить запрос и проанализировать полученные результаты.

## 2. Создать запрос, иллюстрирующий работу механизма транзакций при добавлении данных в несколько таблиц

Рассмотрим последовательность действий при создании и использовании запроса, с помощью которого запускается транзакция, а затем создается новый поставщик, с этим поставщиком заключается договор на поставку, по этому договору поставляется продукция. Имитируется ситуация некорректного или корректного завершения транзакции. Состояние таблиц контролируется до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Для этого необходимо выполнить следующую последовательность действий.

```
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;
```

Запросы SELECT позволяют вывести данные, иллюстрирующие состояние таблиц до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Как видно из приведенных данных, новые записи в таблицах появляется, а потом исчезают.

Теперь необходимо рассмотреть ситуацию корректного завершения транзакции. Для этого в приведенном тексте запроса необходимо изменить оператор ROLLBACK на COMMIT. Выполнить запрос и проанализировать полученные результаты.

### 3. Создать запрос, иллюстрирующий работу механизма транзакций при изменении данных в нескольких таблицах

Рассмотрим последовательность действий при создании и использовании запроса, с помощью которого запускается транзакция, потом меняются данные, введенные в таблицы при выполнении предыдущего запроса. Имитируется ситуация некорректного или корректного завершения транзакции. Состояние таблиц контролируется до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Для этого необходимо выполнить следующую последовательность действий.

```
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;
```

Запросы SELECT позволяют вывести данные, иллюстрирующие состояние таблиц до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Как видно из приведенных данных, новые записи в таблицах появляется, а потом исчезают.

Теперь необходимо рассмотреть ситуацию корректного завершения транзакции. Для этого в приведенном тексте запроса необходимо изменить оператор ROLLBACK на COMMIT. Выполнить запрос и проанализировать полученные результаты.

### 4. Создать запрос, иллюстрирующий работу механизма транзакций при удалении данных из нескольких таблиц

Рассмотрим последовательность действий при создании и использовании запроса, с помощью которого запускается транзакция, в рамках которой удаляется поставщик, который был создан при

выполнении запроса 2 и данные которого были изменены при выполнении запроса 3. С учетом механизма контроля посылочной целостности, используется (CASCADE), данные будут удалены в нескольких таблицах. Имитируется ситуация некорректного или корректного завершения транзакции. Состояние таблиц контролируется до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Для этого необходимо выполнить следующую последовательность действий.

```
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;
```

Запросы SELECT позволяют вывести данные, иллюстрирующие состояние таблиц до начала транзакции, в процессе выполнения транзакции и после завершения транзакции. Как видно из приведенных данных, новые записи в таблицах появляются, а потом исчезают.

Теперь необходимо рассмотреть ситуацию корректного завершения транзакции. Для этого в приведенном тексте запроса необходимо изменить оператор ROLLBACK на COMMIT. Выполнить запрос и проанализировать полученные результаты.

## **5. Оформить отчет по лабораторной работе**

В отчет включить основные этапы выполнения лабораторной работы и снимки экрана, которые демонстрируют.

## **6. Вопросы для самоконтроля**

1. Что такое транзакция?
2. Таблицы всех типов в СУБД MySQL поддерживают транзакции?
3. Таблицы всех типов в СУБД MySQL не поддерживают транзакции?

4. Каким образом в СУБД MySQL можно отключить режим автоматического завершения транзакций?
5. Какой оператор используется для завершения транзакции?
6. Какой оператор используется для отката изменений, выполненных транзакцией?
7. С помощью какой команды в СУБД MySQL можно включить режим автоматического завершения транзакций для отдельной последовательности операторов?
8. С таблицами какого типа могут быть использованы операторы SAVEPOINT и ROLLBACK TO SAVEPOINT?
9. Каково назначение операторов SAVEPOINT и ROLLBACK TO SAVEPOINT?
10. С какими проблемами связано параллельное выполнение транзакций?
11. Какие существуют уровни изоляции транзакций и проблемы каждый из этих уровней позволяет решить?
12. Какой тип таблиц используется в MySQL по умолчанию (начиная с версии 5.5)?
13. Какие уровни изоляции транзакций поддерживает InnoDB?
14. Какой уровень изоляции транзакций по умолчанию используется в InnoDB?

## Лабораторная работа 8

### УПРАВЛЕНИЕ ПРАВАМИ ПОЛЬЗОВАТЕЛЕЙ

**Цель работы:** изучить основы работы с учетными записями и привилегиями пользователей на примере СУБД MySQL.

#### ход работы

##### 1. Создать новые учетные записи пользователей

Система управления базами данных MySQL является многопользовательской средой, поэтому для доступа к таблицам базы данных `supply` могут быть созданы различные учетные записи с разным уровнем привилегий.

Учетной записи менеджера по закупкам можно предоставить привилегии на просмотр таблиц `supplier`, `supplier_org`, `supplier_person` и `contract`, добавление новых записей, удаление и обновление уже существующих записей в данных таблицах.

Администратору базы данных `supply` можно предоставить более широкие полномочия (возможность создания таблиц, редактирование и удаление уже существующих, создание и редактирование учетных записей пользователей и т.п.).

Для работника склада достаточно лишь просмотра таблиц `contract` и `supplied`, а также добавление новых записей, удаление и обновление уже существующих записей в таблице `supplied`.

Рассмотрим создание учетных записей для разных пользователей базы данных.

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

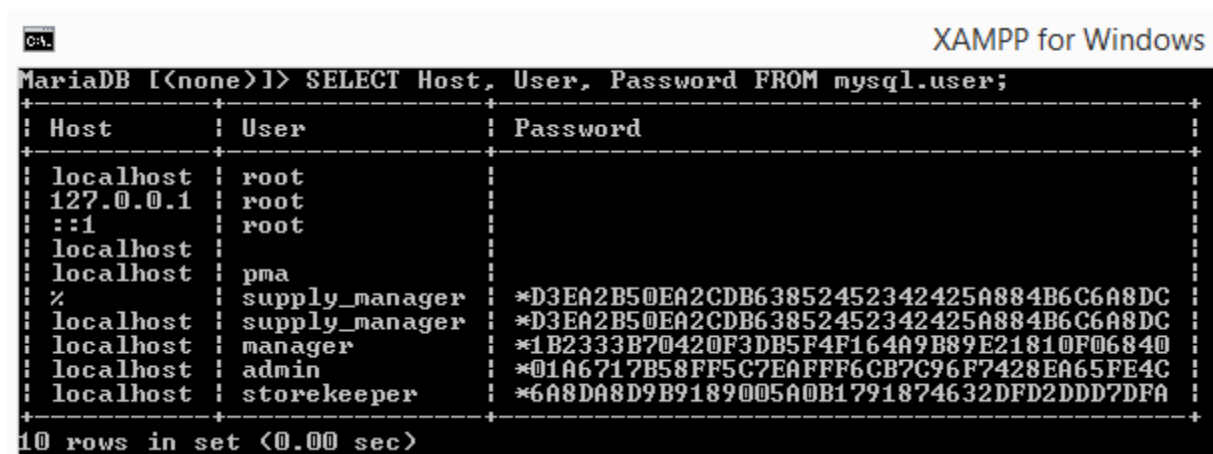
Данный запрос позволяет создать учетные записи для следующих пользователей:

- 1) администратора с паролем «admin123»;
- 2) менеджера по закупкам с паролем «manager123»;
- 3) работника склада с паролем «storekeeper123».

Для удаления учетной записи используется оператор DROP USER. Изменение имени пользователя в аккаунте выполняется с помощью оператора RENAME USER% old\_name% TO% new\_name%.

Поскольку все учетные записи пользователей хранятся в таблице user системной базы данных mysql, проверить создания рассмотренных учетных записей можно с помощью следующего запроса (рисунок 8.1):

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



Host	User	Password
localhost	root	
127.0.0.1	root	
:::1	root	
localhost	pma	
localhost	supply_manager	*D3EA2B50EA2CDB63852452342425A884B6C6A8DC
%	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. Назначить привилегии для созданных учетных записей

Рассмотренные выше операторы позволяют создавать, удалять и редактировать учетные записи, однако они не позволяют изменять привилегии пользователя - сообщать MySQL, пользователь имеет право только на чтение информации, который на чтение и редактирование, а кому предоставлены права изменять структуру БД и создавать учетные записи.

Необходимо назначить привилегии для созданных учетных записей.



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

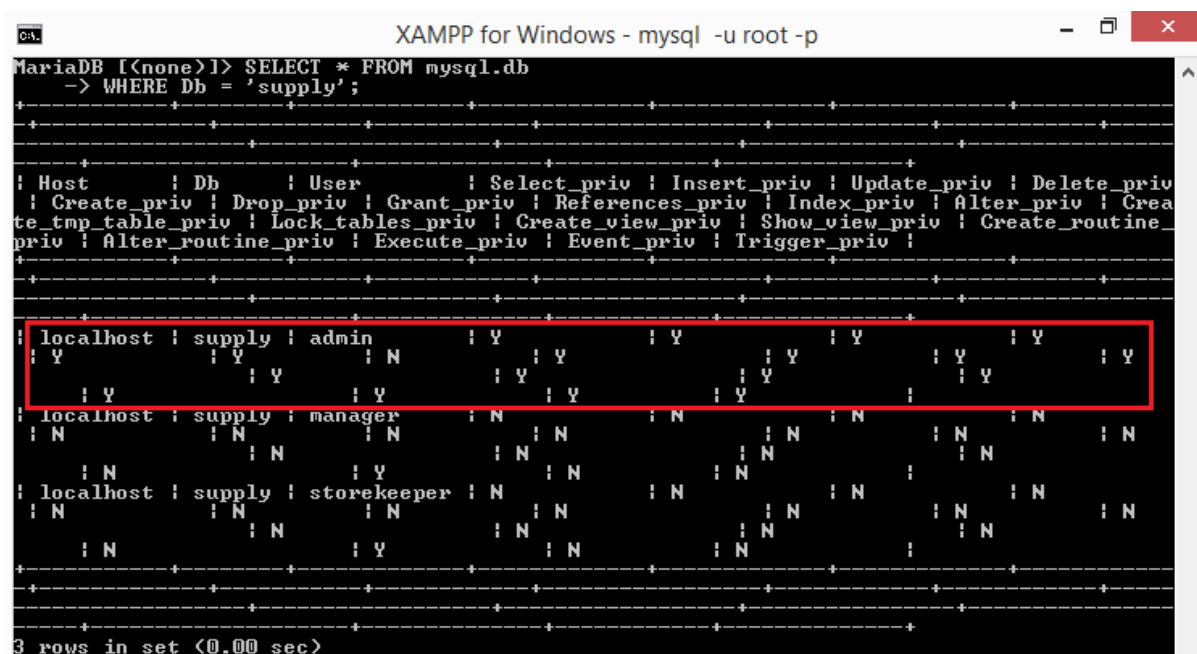
GRANT SELECT, INSERT, UPDATE, DELETE ON supply.supplier TO 'manager'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON supply.supplier_org TO 'manager'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON supply.supplier_person TO 'manager'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON supply.contract 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 избавления учетной записи пользователя определенных привилегий используется оператор REVOKE. Данный оператор не удаляет учетные записи, а только отменяет предоставленные ранее привилегии. Поэтому для окончательного удаления аккаунта необходимо воспользоваться оператором DROP USER.

Проверить привилегии учетной записи admin, которому были предоставлены все права на уровне базы данных supply, можно с помощью следующего запроса (рисунок 8.2).

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



XAMPP for Windows - mysql -u root -p

MariaDB [(none)]> SELECT \* FROM mysql.db  
-> WHERE Db = 'supply';

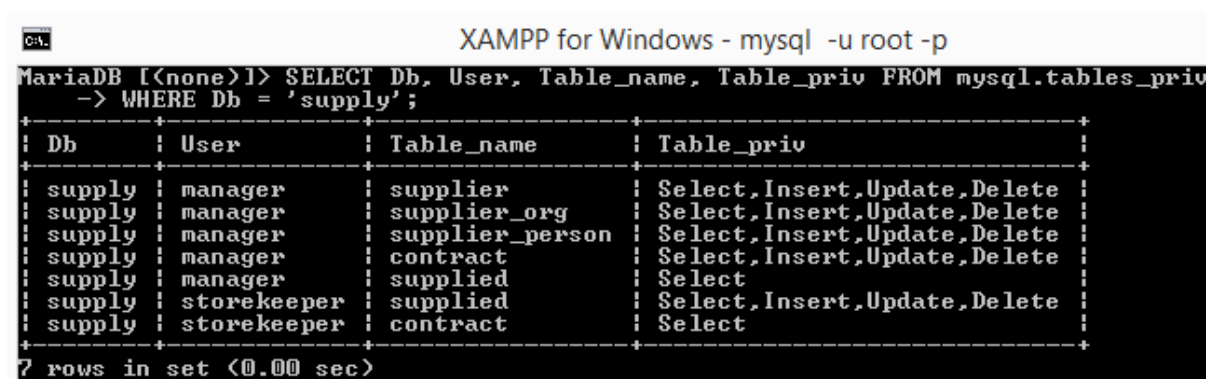
Host	Db	User	Select_priv	Insert_priv	Update_priv	Delete_priv	Create_priv	Drop_priv	Grant_priv	References_priv	Index_priv	Alter_priv	Create_tmp_table_priv	Lock_tables_priv	Create_view_priv	Show_view_priv	Create_routine_priv	Alter_routine_priv	Execute_priv	Event_priv	Trigger_priv
localhost	supply	admin	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
localhost	supply	manager	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
localhost	supply	storekeeper	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

3 rows in set (0.00 sec)

Рисунок 8.2

Аналогично можно проверить привилегии учетных записей manager и storekeeper, для которых были определены определенные ограничения в работе с таблицами базы данных supply (рисунок 8.3).

```
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

Кроме того, определенным пользователям необходимо предоставить также привилегии, которые позволят им использовать представления, содержащиеся в базе данных supply. Например, пользователю manager должны быть предоставлены права для просмотра представлений contract\_supplier и supplier\_info, тогда как для пользователя storekeeper должно быть доступным только представления contract\_supplier.

### 3. Оформить отчет по лабораторной работе

В отчет включить основные этапы выполнения лабораторной работы и снимки экрана, которые демонстрируют.

### 4. Вопросы для самоконтроля

1. Как выглядит аккаунт пользователя в СУБД MySQL?
2. Из каких составляющих формируется аккаунт?
3. Каково назначение в составляющих учетной записи?
4. Каким образом можно проверить все учетные записи?
5. Какая команда используется для создания учетной записи?
6. Какая команда используется для удаления учетной записи?
7. Каким образом можно изменить имя пользователя в аккаунте?

8. С помощью какого оператора можно определить определенные привилегии для необходимого аккаунта?
9. Какой оператор может быть использован для отмены привилегий?
10. Какие привилегии могут быть определены для учетной записи?
11. Какие уровни назначения привилегий?
12. Каким образом можно проверить глобальные привилегии, привилегии базы данных и привилегии таблиц?

## Лабораторная работа 9

### РАЗРАБОТКА ПРИКЛАДНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РАБОТЫ С БАЗОЙ ДАННЫХ

**Цель работы:** изучить основы разработки прикладного программного обеспечения, предназначенного для работы с СУБД MySQL, с использованием языка PHP.

#### Ход работы

**Внимание!** В лабораторной работе будет продемонстрировано создание только упрощенного фрагмента приложения для работы с базой данных.

#### 1. Определить основные функциональные возможности программного обеспечения

Основные функциональные возможности фрагмента web-приложения, предназначенного для работы с базой данных supply, приведены в виде UML диаграммы прецедентов (рисунок 9.1).

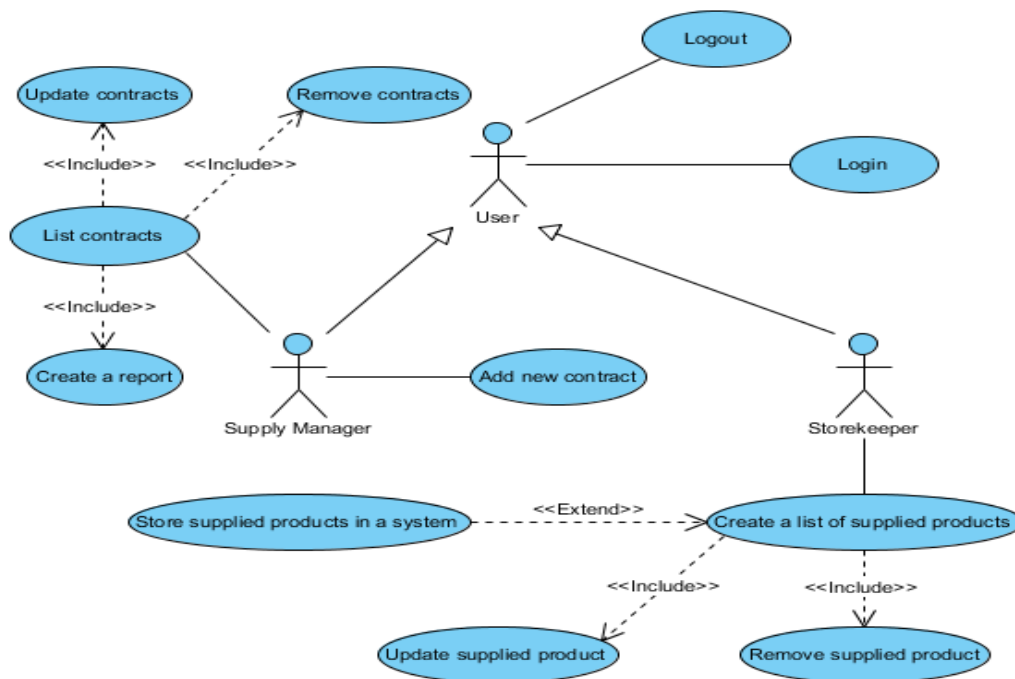


Рисунок 9.1

## 2. Разработать страницу для авторизации пользователей программного обеспечения

Все страницы web-приложения необходимо разместить в каталоге `xampp / htdocs / supply`.

Прежде чем приступить к созданию страницы авторизации, необходимо разработать функциональность программного обеспечения по установлению соединения с базой данных. Для этого создать файл `connect.php` с последующим содержанием.

```
<?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;
}
?>
```

Кроме того, необходимо разработать основную страницу web-приложения, для чего создать файл `index.php` со следующим содержанием.

```
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 ?>
```

Строки 2 - 4 содержат начало пользовательского сеанса и подключения файла, содержащего функцию `db_conn ()` для установления

соединения с базой данных. Строки 9 - 15 содержат проверку наличия пользовательского сеанса и подключения к базе данных. В случае если пользователь не был авторизован, он будет перенаправлен на страницу авторизации (строка 14). Строка 11 определяет подключение файла, который содержит обработку форм добавления, обновления и удаления данных, он будет создан позже.

```

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

```

Следующие строки (17 - 39) определяют вид главной страницы: информацию о текущем пользователе (рисунок 9.2), содержимое страницы в соответствии с типом пользователя, отключение соединения с базой данных (строка 39). В строке 24 определена ссылка, позволяющая удалить все переменные сеанса и завершить сеанс использования приложения пользователем. Для этого используется 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");

```

Страница авторизации пользователя сохраняется в файле 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 ?>
```

Строки 9 и 10 определяют запись переменных сеанса, содержащих данные об учетной записи пользователя. Именно эти переменные используются в файле connect.php для установления соединения с базой данных с помощью `mysqli_connect()`. Если сеанс пользователя уже был установлен, он будет перенаправлен на главную страницу `index.php` (строки 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>
```

В строках 20 - 45 определена статическая структура страницы авторизации пользователя, которая содержит соответствующую форму с необходимыми элементами пользовательского интерфейса (рисунок 9.3).

### Supply Application Login

User name

Password

Login

Рисунок 9.3

### 3. Разработать функциональность программного обеспечения для работы менеджера по закупкам

Страница, содержащая функциональность программного обеспечения для работы менеджера по закупкам, содержится в файле 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>
```

Строки 1 - 24 содержат проверку наличия пользовательского сеанса, а также режима работы с данными о договорах (создание, обновление или удаление), от чего зависит элемент интерфейса - ссылка New contract,



предназначенный для создания нового договора (рисунок 9.4), или Back - для возврата к просмотру данных обо всех договорах (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

Строки 26 - 99 содержат проверку режимов создания новой записи (рисунок 9.5), обновления (рисунок 9.6) или удаления существующей записи (рисунок 9.7) и отображения соответствующих форм с определенными элементами интерфейса пользователя.

```

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 <p>
56     <b>Note</b>
57 </p>
58 <p>
59     <?php
60     # retrieve and display contract note of the updated contract
61     if (isset($_GET["action"]) && $_GET["action"] == "update") {
62         $contract_number = $_GET["id"];
63
64         $sql = "SELECT contract_note FROM contract WHERE contract_number = {$contract_number}";
65         $result = mysqli_query($conn, $sql);
66         $row = mysqli_fetch_assoc($result);
67     }
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

Строки 100 - 133, в свою очередь, определяют таблицу с данными о договорах и соответствующими ссылками (столбик Action), предназначенными для манипулирования этими данными (рисунок 9.4).

Строки 135 - 179 содержат определения дополнительной таблицы, предназначенной для отображения перечня поставленных товаров по определенному договору (рисунок 9.8). Для демонстрации данной таблицы выполняется необходимая проверка режима просмотра данных о договорах (строки 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. Разработать функциональность программного обеспечения для работы работника склада

Страница, содержащая функциональность программного обеспечения для работы работника склада, содержится в файле storekeeper.php.

Строки 1 - 14 содержат проверку наличия пользовательского сеанса, а также наличия переменной сеанса - массива, в который будут записаны товары, поставленные на склад, но еще не сохраненные в базе данных.

```

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>

```

В строках 16 - 72 определяется таблица поставленных на склад товаров.

```

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 этом, выполняется проверка наличия товаров в массиве (переменная сеанса) и вывод формы (рисунок 9.9), что позволяет записать принятые товары в базу данных (строки 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      ?>

```

Также проверяется наличие ожидаемых поставок (если есть так называемые «пустые» договоры, которые были заключены, но по которым еще не были поставлены товары) в строках 17 - 24. В случае наличия таких договоров, демонстрируется форма (рисунок 9.10) для добавления принятого товара (строки 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. Разработать функциональность для формирования отчета в формате Excel по объемам поставленной продукции за определенный период

Реализация данной функциональной возможности будет находиться также в файле action.php, который содержит обработку пользовательских форм.

Строки 1 - 34 данного файла содержат обработки форм, которые предназначены для создания записей о договорах, а также обновления и удаления существующих записей. Следует обратить внимание, что для выполнения операций создания, обновления и удаления записей из таблицы contract используется созданная ранее хранимая процедура sp\_contract\_ops.

В строках 36 - 60 обрабатываются формы, которые предназначены для создания записи о поставленном, но еще не сохраненном в базе данных товаре, а также для удаления таких записей из массива, хранящегося в качестве переменной сеанса пользователя.



```

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 }

```

Строки 62 - 103 демонстрируют сохранение поставленных товаров в базу данных. Следует обратить внимание на то, что создание записей о товарах, поставленные по определенному договору, в таблице `supplied`, выполняется внутри транзакции, поскольку частичный, в силу каких-либо обстоятельств, перенос данных о принятых товары из сеансовой переменной в оперативную базу данных является неприемлемым.

```

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 ?>

```

Рассмотренный код файла action.php необходимо дополнить следующим фрагментом, который предназначен для формирования и сохранения документа Excel отчету об объемах поставленной продукции за определенный период. Для формирования отчета будет использоваться ранее созданная хранимая процедура sp\_contract\_total.

Содержимое файла manager.php необходимо дополнить ссылкой (рисунок 9.11), которое позволит сформировать и загрузить отчет (строка 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>

```

Кроме того, файл action.php необходимо дополнить кодом (строки 104 - 127), предназначенным непосредственно для формирования и загрузки отчета (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. Оформить отчет по лабораторной работе

В отчет включить основные этапы выполнения лабораторной работы и снимки экрана, которые их демонстрируют.

## 7. Вопросы для самоконтроля

1. Разработать функциональность программного обеспечения для работы администратора базы данных supply. Администратор должен иметь возможность создавать, обновлять и удалять записи во всех таблицах базы данных.

2. Дополнить рассмотренный фрагмент программного обеспечения для работы с БД supply возможностью сортировать строки в таблице Contracts (файл manager.php) в прямом и обратном направлениях:

- по номеру договора;
- по дате заключения договора.

3. Дополнить рассмотренный фрагмент программного обеспечения для работы с БД supply возможностью сортировать строки в таблице Supplied products by contract #X (файл manager.php) в прямом и обратном направлениях:

- по названию поставленного товара;
- по количеству единиц поставленного товара;
- по стоимости единицы поставляемого товара.

4. При попытке обновления данных об определенном договоре открывается соответствующая форма, содержащая combo box со списком поставщиков. Внести изменения в программное обеспечение таким образом, чтобы в этом combo box сразу после загрузки формы был выбран

поставщик, с которым на данный момент заключен договор, который редактируется.

5. Учитывая механизм ссылочной целостности, используемый в базе данных supply, пока невозможно удаление данных о договоре, по которому были поставлены товары. Внести изменения в программное обеспечение (например, в хранимую процедуру sp\_contract\_ops) для того, чтобы данные о договоре можно было удалить несмотря на наличие данных о поставленных по этому договору товарах.

6. Учитывая механизм ссылочной целостности, используемый в базе данных supply, пока невозможно удаление данных о договоре, по которому были поставлены товары. Внести изменения в программное обеспечение таким образом, что для не «пустых» договоров не будет доступна попытка их удаления.

7. Как видно из рисунка 9.12, названия столбцов в сгенерированном отчете не понятны для пользователя, особенно это касается столбцов, содержащих агрегированные данные. Внести изменения в программное обеспечение таким образом, чтобы указать названия Contract, Date, Total amount и Total cost для соответствующих столбцов.

8. Текущая реализация позволяет сформировать отчет (рисунок 9.12) на основе фиксированного диапазона дат - начиная с 1 января 2018 года до времени генерации отчета. Внести изменения в программное обеспечение таким образом, чтобы пользователь мог самостоятельно указывать необходимый период для генерации отчета об объемах поставленной продукции.

9. Предоставить менеджеру по закупкам возможность работать с данными о поставщиках (добавлять записи, обновлять и удалять существующие записи). Обеспечить возможность просмотра перечня договоров, заключенных с определенным поставщиком.

10. Дополнить программное обеспечение автоматической генерацией приходной накладной сразу после того, как данные поступившие от поставщика в рамках определенного договора на поставку, были сохранены работником склада в базу данных.