Laboratory work 1

DATABSE CREATION USING SQL LANGUAGE

Goal: learn how to create and link database tables using the MySQL DBMS.

Progress

1. Install MySQL

It is recommended to install MySQL using one of the freely distributed WAMP (Windows, Apache, MySQL, and PHP) or LAMP (Linux, Apache, MySQL, and PHP) servers, such as OpenServer or XAMPP, to simplify the installation and subsequent use of the database management system (DBMS). In subsequent examples of laboratory work, the XAMPP server will be used.

Once the XAMPP server is installed, run the server control panel, run MySQL and open the server command line (figure 1.1).

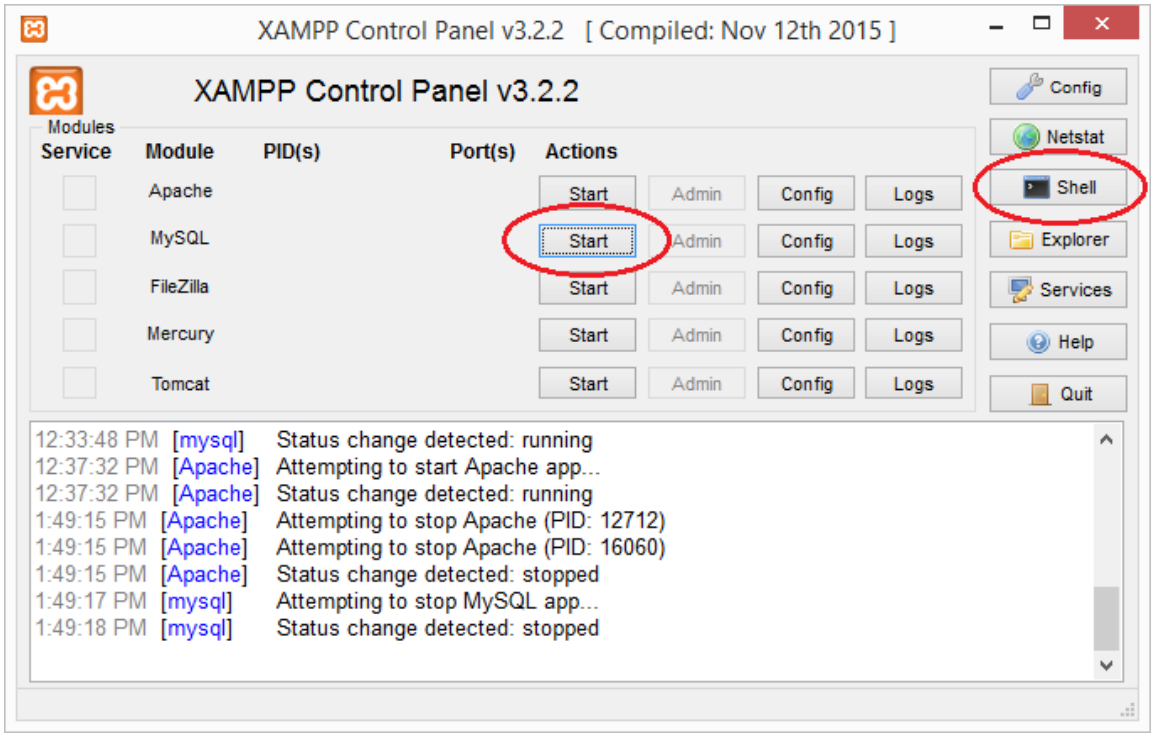

Figure 1.1

2. Connect to MySQL and create a database

Use the following command in the MySQL server command line:

mysql -u root -p

Then enter the password (figure 1.2). Usually the root user password is empty, so just press Enter.

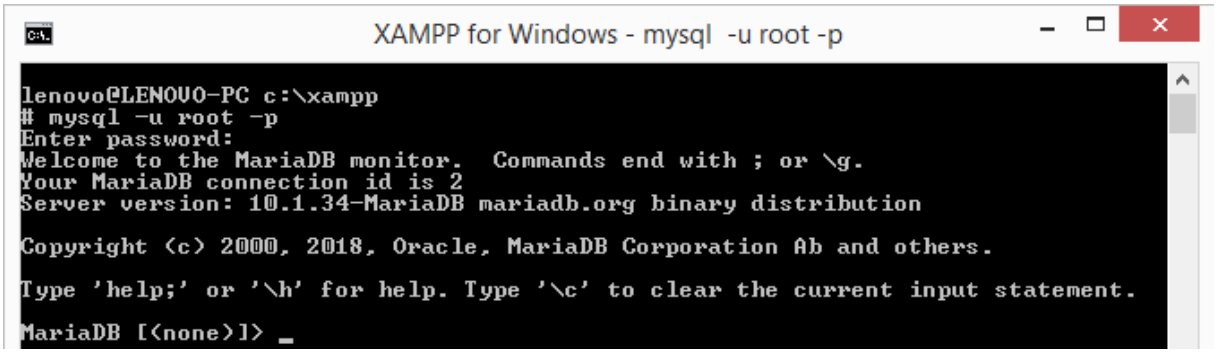

Figure 1.2

In order to disconnect from MySQL, use the exit command.

The main commands that will be used from time to time when working with MySQL are the following:

1) USE database – select a database (DB) for further work;

2) SHOW DATABASES – get a list of databases;

3) SHOW TABLES – get a list of tables for the selected database;

4) SHOW COLUMNS FROM table – get information about the table;

5) SHOW INDEX FROM table – get information about the indexes defined for the table.

A database should be created using the command:

CREATE DATABASE supply;

Execution of this command will allow to create a database, the work to be considered in the laboratory practice. You can check the database creation using the SHOW DATABASES command.

3. Create database tables and link them

To study the peculiarities of working with a MySQL database, a database of a company that purchases goods from different suppliers will be considered. Purchase of goods is carried out in batches and is executed in the form of supply contracts. Each contract has a unique number and is concluded with only one supplier. The documents for each contract specify the name, the size of the delivered batch and the price (in UAH).

Creating tables is performed using the CREATE TABLE statement. Thus, for the current database, it is necessary to create the following tables:

```
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;
```

Check the generated tables in the supply database (figure 1.3).

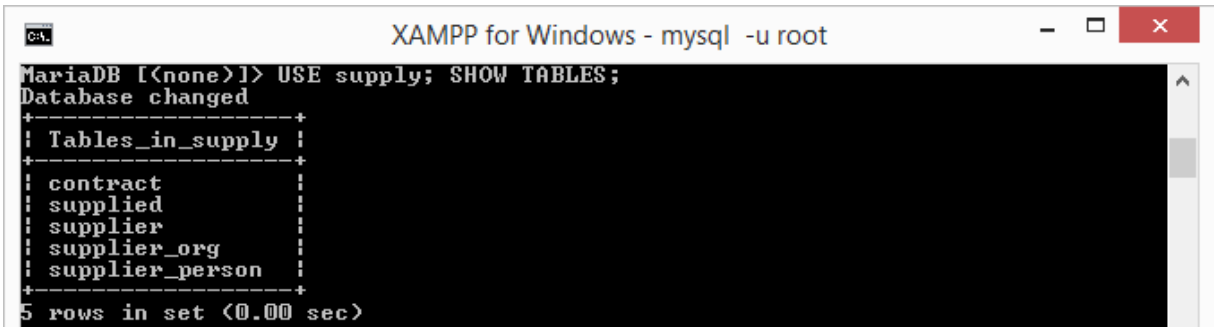

Figure 1.3

4. Modification of table structure

Change the structure of an existing table using the ALTER TABLE statement. Assume that you need to create another table in the supply database, which will be used to store data on the facts of implementation of supply contracts (figure 1.4).

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

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

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

Figure 1.4

In order to link the created contract_delivered table with the contract table, apply the ALTER TABLE command (figure 1.5).

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

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

Figure 1.5

5. Delete tables

Delete a table using the DROP TABLE statement. Since the created contract_delivered table will not be used in future work, it can be deleted using this command (figure 1.6).

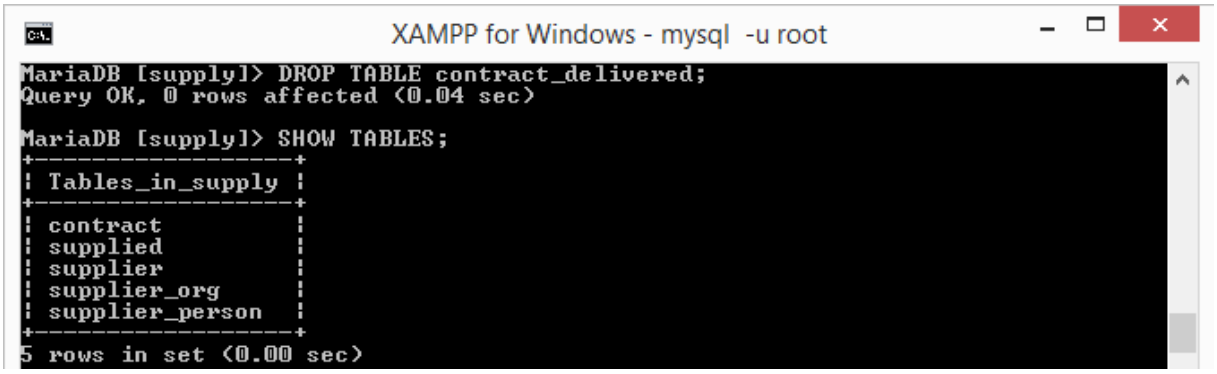

Figure 1.6

6. Make a report on laboratory work

The report should include the main stages of laboratory work and screenshots that demonstrate them.

7. Questions

1. How to access the command line of the MySQL server?
2. How to make a connection to the MySQL server using the name and password of the specific user?
3. List the basic commands used for MySQL server administration and their purposes.
4. Which command is used to create the new database? How to check that database is created?
5. Which SQL statements are used to create and link tables?
6. Which SQL statement is used to modify table's structure?
7. Which SQL statement is used to delete tables from a database?
8. How to check presence or absence of the created or removed tabled respectively?
9. How to set the name of a foreign key while linking tables?
10. Which shortcomings are present in the current database structure? How to resolve these issues?

Laboratory work 2

DATA MANIPULATION USING SQL LANGUAGE TOOLS: INSERT, UPDATE, AND DELETE

Goal: learn how to use SQL language operators to add, update, and delete data in MySQL DBMS.

Progress

1. Adding data to a created database

The INSERT statement is used to add data.

The following commands allow to insert the supplier data in the created database:

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

Check entries created in the supplier table (figure 2.1).

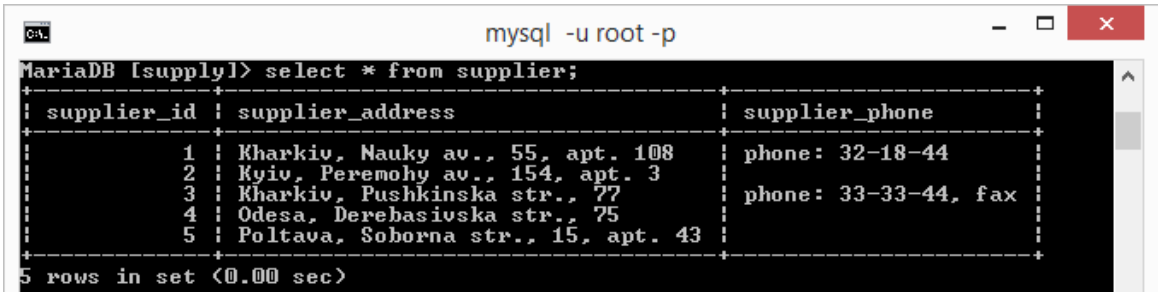

Figure 2.1

The following commands allow to insert the data about the individual entrepreneurs in the database created:

```
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');
```

Check entries created in the supplier_person table (figure 2.2).

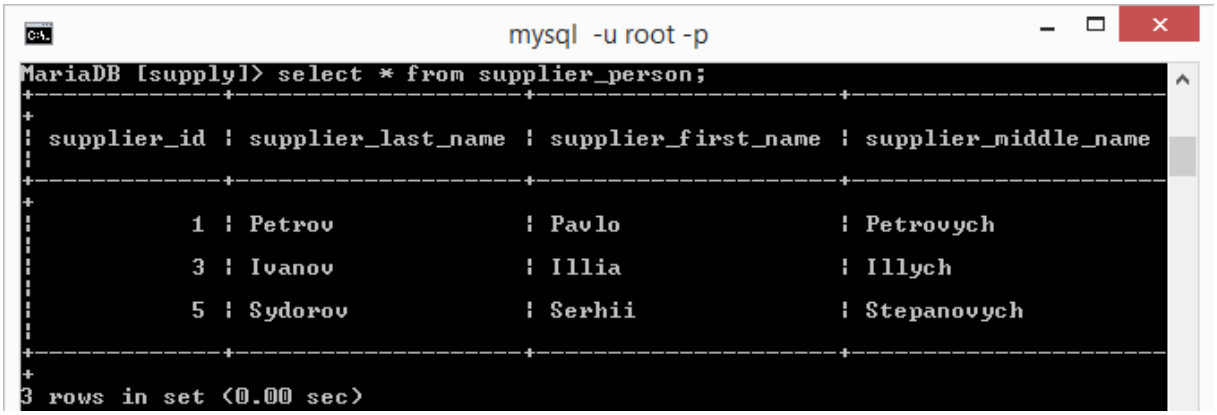

Figure 2.2

The following commands allow you to insert the data about the legal entities in the created database:

```
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');
```

Check entries created in the supplier_org table (figure 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)
```

Figure 2.3

The following commands allow to insert the details of the concluded contracts in the created database:

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');

Check entries created in the contract table (figure 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 28.08.2
018 |
|               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 21.09.2
018 |
+-----------------+---------------------+-------------+-------------------------
----+
5 rows in set (0.00 sec)
```

Figure 2.4

The following commands allow to insert the data about the delivered goods in the created database:

```
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);
```

Check the entries created in the supplied table (figure 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)
```

Figure 2.5

2. Database update

Updating data (changing the value of fields in existing records) in the database is performed using the operator UPDATE.

For example, if you want to reduce the value of the printer that was delivered under contract number 4, by 5%, the command will be the following (figure 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)
```

Figure 2.6

3. Deleting data from a database

To delete data from database tables, the DELETE statement is used.

For example, to remove the delivered goods that were supplied according to the contract with the number 5, it is required to execute the following command (figure 2.7):

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

```
mysql -u root -p
MariaDB [supply]> DELETE FROM supplied WHERE contract_number = 5;
Query OK, 3 rows affected (0.01 sec)

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 |        332.50 |
|               4 | TV                |              56 |        990.00 |
+-----------------+-------------------+-----------------+---------------+
12 rows in set (0.00 sec)
```

Figure 2.7

Restore deleted entries using INSERT commands.

4. Make a report on laboratory work

The report should include the main stages of laboratory work and screenshots that demonstrate them.

5. Questions

1. Show the structure and examples of the INSERT statement.
2. Show the structure and examples of the UPDATE statement.
3. Show the structure and examples of the DELETE statement.
4. How to update all records in the database table?
5. How to remove all records from the database table?
6. How to remove the 20 latest concluded contracts?
7. How to increase the price of the 5 cheapest product supplied by the specific contract for 15%?
8. Which structure of the INSERT command should be provided in order to skip duplicate keys without error occurrence?

Laboratory work 3

DATA MANIPULATION USING SQL LANGUAGE: SELECT QUERIES AND THEIR BASIC FEATURES

Goal: learn how to use the SQL SELECT statement for data querying, using the MySQL database.

Progress

1. Create and execute SQL SELECT queries

Query 1

Form a list of goods delivered by supplier 1 (Petrov P. P.) under the contract 1 (figure 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);
```

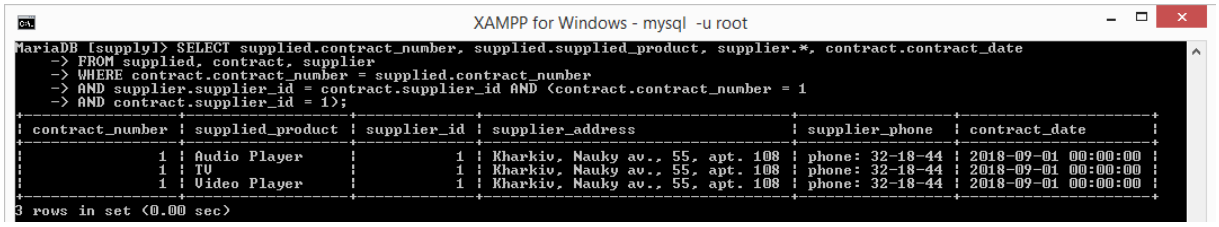

Figure 3.1

Query 2

Form a list of the goods delivered by the supplier 1 (Petrov P. P.) in the period from 2018-09-05 to 09/08/2012 (figure 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   |
+----------------+---------------------+------------------+---------------+-------------+--------------------------------+-----
-------------+
|              2 | 2018-09-10 00:00:00 | Audio Player     |        450.00 |           1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
|              2 | 2018-09-10 00:00:00 | Stereo System    |        500.00 |           1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
|              2 | 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)
```

Figure 3.2

Query 3

Form a list of goods that were delivered in the 9 month of 2018 including the name of the supplier and delivery date (figure 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   |
+----------------+---------------------+------------------+---------------+-------------+--------------------------------+-----
-------------+
|              1 | 2018-09-01 00:00:00 | Audio Player     |        700.00 |           1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
|              1 | 2018-09-01 00:00:00 | TV               |       1300.00 |           1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
|              1 | 2018-09-01 00:00:00 | Video Player     |        750.00 |           1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
|              2 | 2018-09-10 00:00:00 | Audio Player     |        450.00 |           1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
|              2 | 2018-09-10 00:00:00 | Stereo System    |        500.00 |           1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
|              2 | 2018-09-10 00:00:00 | Video Player     |        450.00 |           1 | Kharkiv, Nauky av., 55, apt. 108 | phon
e: 32-18-44 |
|              4 | 2018-09-24 00:00:00 | Audio Player     |        320.00 |           2 | Kyiv, Peremohy av., 154, apt. 3  |
|              4 | 2018-09-24 00:00:00 | Printer          |        332.50 |           2 | Kyiv, Peremohy av., 154, apt. 3  |
|              4 | 2018-09-24 00:00:00 | TV               |        990.00 |           2 | Kyiv, Peremohy av., 154, apt. 3  |
|              3 | 2018-09-23 00:00:00 | Audio Player     |        550.00 |           3 | Kharkiv, Pushkinska str., 77     | phon
e: 33-33-44, fax |
|              3 | 2018-09-23 00:00:00 | Monitor          |        550.00 |           3 | Kharkiv, Pushkinska str., 77     | phon
e: 33-33-44, fax |
|              3 | 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)
```

Figure 3.3

Query 4

Form a list of contracts (number, date, title) and the total amount for each contract (batch size multiplied by the price per unit and summed up by the contract). The list should be sorted by contract numbers (figure 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)
```

Figure 3.4

Query 5

Form a list of contracts (number, date, title) and the total amount for each contract (batch size multiplied by the price per unit and summed up by the contract). The list should be sorted by increasing total amounts for each contract. After that, the filter must be applied to the list, in order to exclude from the result of the query those records for which the contract number is less than 4 (figure 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)
```

Figure 3.5

Query 6

Display the information on the largest batch of goods in all contracts with the supplier, as well as the number and the date of the contract (figure 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);
```

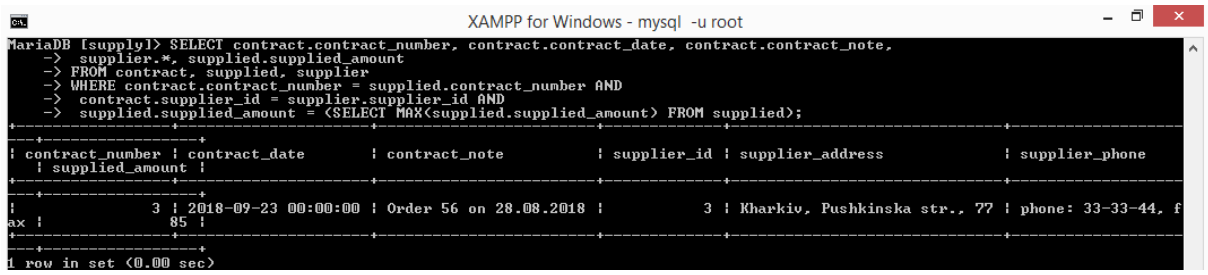

Figure 3.6

Query 7

Form a list of suppliers (name and code) with which no contract has been concluded (figure 3.7).

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

Figure 3.7

Query 8

Form a list of the names of supplied goods with an indication of the average delivery price per unit (regardless of the supplier) (figure 3.8).

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

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

Figure 3.8

Query 9

Form a list of goods (name, quantity and price, supplier), for which the price per unit is more than average (figure 3.9).

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

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

Figure 3.9

Query 10

Display information about the five most expensive products (name, price per unit, supplier) (figure 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)
```

Figure 3.10

Query 11

Form a supplier list with code, address and supplier information. When forming supplier data for individuals, display the surname and initials, and for legal entities – the name (figure 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)
```

Figure 3.11

Query 12

Form a list of contracts (include the number, delivery date and supplier information), the total number of goods delivered and the total amount for each contract. When forming the supplier data for individuals, display the last name and initials, and for legal entities – the name. The result should contain only those contracts on the basis of which the goods were actually delivered (e.g., the result of the query should not contain so-called “empty” contracts) (figure 3.12).

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

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

Figure 3.12

Query 13

Form a list of goods (with the number of the contract and delivery date) delivered by suppliers 1 (Petrov P. P.) and 2 (Interfruit) (figure 3.13).

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

```
XAMPP for Windows - mysql  -u root
MariaDB [supply]> SELECT supplied.contract_number, contract.contract_date,
    -> supplied.supplied_product, supplier.supplier_id
    -> FROM supplied, contract, supplier
    -> WHERE contract.contract_number = supplied.contract_number
    -> AND supplier.supplier_id = contract.supplier_id AND contract.supplier_id = 1
    -> UNION
    -> SELECT supplied.contract_number, contract.contract_date,
    -> supplied.supplied_product, supplier.supplier_id
    -> FROM supplied, contract, supplier
    -> WHERE contract.contract_number = supplied.contract_number
    -> AND supplier.supplier_id = contract.supplier_id AND contract.supplier_id = 2
    -> ORDER BY supplier_id, contract_number;
+-----------------+---------------------+------------------+-------------+
| contract_number | contract_date       | supplied_product | supplier_id |
+-----------------+---------------------+------------------+-------------+
|               1 | 2018-09-01 00:00:00 | Audio Player     |           1 |
|               1 | 2018-09-01 00:00:00 | 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)
```

Figure 3.13

Query 14

Form a nomenclature of goods (a list of product names) that were supplied only by supplier 1 (Petrov P. P.), or only supplier 2 (Interfruit), or both supplier 1 and supplier 2 (figure 3.14).

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

```
XAMPP for Windows - mysql  -u root
MariaDB [supply]> SELECT DISTINCT supplied.supplied_product
    -> FROM supplied, contract
    -> WHERE contract.contract_number = supplied.contract_number AND contract.supplier_id = 1
    -> UNION
    -> SELECT DISTINCT supplied.supplied_product
    -> FROM supplied, contract
    -> WHERE contract.contract_number = supplied.contract_number AND contract.supplier_id = 2
    -> ORDER BY supplied_product;
+------------------+
| supplied_product |
+------------------+
| Audio Player     |
| Printer          |
| Stereo System    |
| TV               |
| Video Player     |
+------------------+
5 rows in set (0.00 sec)
```

Figure 3.14

Query 15

Generate a list of items that should demonstrate the frequency of deliveries. Include only items that shipped more than once to the list. The list should be sorted by decreasing the supply frequency (figure 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;
```

```
XAMPP for Windows - mysql  -u root
MariaDB [supply]> 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;
+------------------+------------------+
| supplied_product | SupplyFrequency  |
+------------------+------------------+
| Audio Player     |                5 |
| TV               |                4 |
| Video Player     |                3 |
+------------------+------------------+
3 rows in set (0.00 sec)
```

Figure 3.15

Query 16

Retrieve data on quantitative dynamics of goods deliveries during 2018. The data should be aggregated in months and presented as a table with lines of product names, and columns are the numbers of months in 2018. At the intersection of a row and a column, the quantity of this product delivered in this month should be displayed (figure 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 |
| TV               |    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)
```

Figure 3.16

Query 17

Form a list of supplied goods. For each item in this list, the following information must be shown: contract number, product name, unit number, unit price, delivery date, month name and year number (figure 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 | TV              |              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 | TV              |              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 | TV              |              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 | TV              |              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)
```

Figure 3.17

2. Make a report on laboratory work

The report should include the main stages of laboratory work and screenshots that demonstrate them.

3. Questions

1. What SQL statement is used to retrieve data from one or several tables?
2. Show the common structure of the SELECT statement.
3. Which form of the SQL SELECT statement might be used if it is required to display all columns of a certain table?
4. Which construction is used to select records that satisfy search criterions?
5. What keyword is used to exclude duplicate rows?
6. Which construction is used to sort values by single or multiple columns?
7. How the reverse sorting might be implemented?
8. Which keyword is used to limit the range of retrieved records?
9. Which construction is used to group retrieved records?
10. Name the aggregation functions, their purpose and basic features.
11. How to give the new name to a specific column?
12. What is the purpose of the HAVING keyword? What is the difference of this keyword from WHERE?
13. Name basic arithmetic, logic, and comparison operators, their purpose and examples of usage.
14. The purpose of the MONTH function and examples of its usage.
15. The purpose of the YEAR function and examples of its usage.
16. The purpose of the IFNULL function and examples of its usage.
17. The purpose of the CONCAT function and examples of its usage.
18. The purpose of the RTRIM function and examples of its usage.
19. The purpose of the SUBSTRING function and examples of its usage.
20. The purpose of the IF function and examples of its usage.
21. Which operator is used to combine results of two queries?

Laboratory work 4

CREATION AND USING VIEWS

Goal: learn how to create and apply views using the MySQL database.

Progress

1. Create a view that allows to see the name of the supplier when viewing the list of contracts

Creating views is done with the CREATE VIEW operator. Thus, you can create a view that allows you to view the list of contracts with the name of the supplier, based on the next query (figure 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;
```

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

Figure 4.1

The result of this query has a certain disadvantage – the data of suppliers - legal and individual suppliers are shown in different columns, and also there are NULL values present. This problem can be fixed by applying the following query (figure 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)
```

Figure 4.2

Now you can create this view with the name contract_supplier using the appropriate SQL statement (figure 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)
```

Figure 4.3

2. Create a view that allows the user to work with limited supplier data

Suppose that for some users, not all general supplier information (stored in the supplier's table) should be available, but only information about the code and supplier address. In this case, the user should be able to see the data of the

supplier as a business entity (for legal entities – the name, for physical persons – surname, name, and patronymic) (figure 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;
```

```
mysql  -u root -p
MariaDB [supply]> select * from supplier_info;
+-------------+-----------------------------------+---------------------------+
| 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)
```

Figure 4.4

If necessary, you can delete the view using the DROP VIEW operator.

3. Make a report on laboratory work

The report should include the main stages of laboratory work and screenshots that demonstrate them.

4. Questions

1. What is the view?
2. Name views advantages and shortcomings.
3. Which SQL language operator is used to build views?
4. Which SQL language operator is used to remove views?
5. How you can check existence of a view in a database?
6. How to specify the list of columns in order to create a view?
7. What is a vertical view?
8. What is a horizontal view?

Laboratory work 5

CREATION AND USING STORED PROCEDURES AND TRIGGERS

Goal: learn how to use and apply the program objects of a database – stored procedures and triggers, using the MySQL database.

Progress

1. Create and use stored procedures

Create stored procedures by using the CREATE PROCEDURE operator. Therefore, you can create a stored procedure that implements a selection of data from the contract, supplier_org, and supplier_person tables using the following statement (figure 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 //
```

Use the CALL operator to execute a certain procedure.

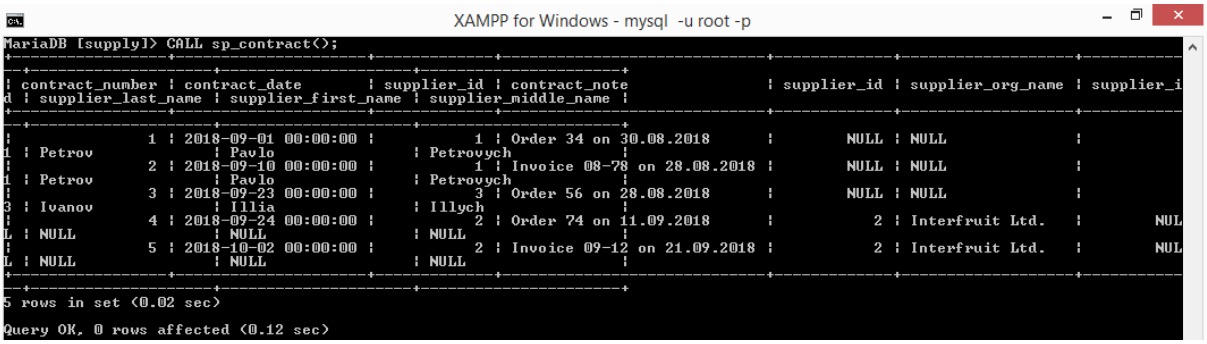

Figure 5.1

To learn about the peculiarities of creating and using procedures with parameters, it is required to create a stored procedure that generates aggregate supply data for a specified interval of dates (figure 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 //
```

You can call the created procedure using the following statement.

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

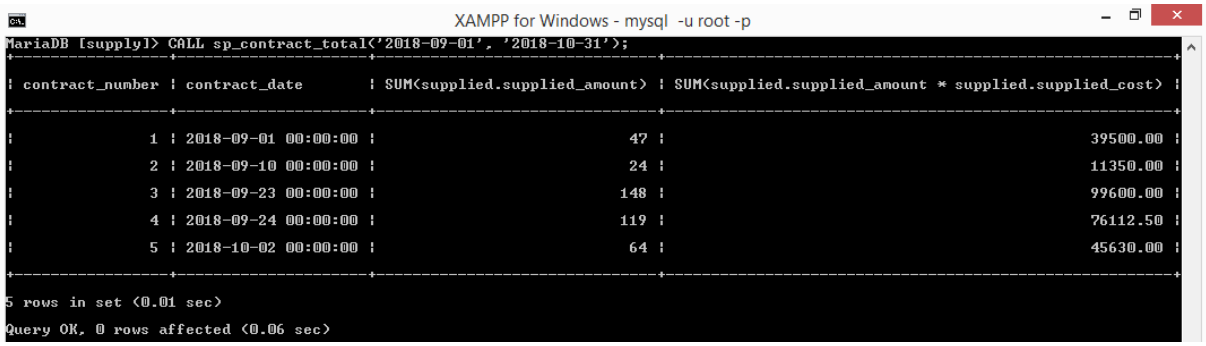

Figure 5.2

The next stored procedure is intended to perform various data modification operations for the contract table. This procedure uses the IF operator to control the data flow.

```
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 //
```

The following query allows to create a contract (Figure 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)
```

Figure 5.3

The following query allows to modify the contract (figure 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)
```

Figure 5.4

The following query allows to delete the contract (figure 5.5).

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

```
XAMPP for Windows - mysql  -u root -p
MariaDB [supply]> CALL sp_contract_ops('d', 6, '2018-12-31', 0, '');
Query OK, 1 row affected (0.01 sec)

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

Figure 5.5

2. Create and use triggers

Assume that when entering data into the contract table, which stores information on supply contracts, the field contract_date, in which the date of the contract is kept, must be completed. Moreover, if this field is left blank when entering a new contract, the current date must be automatically recorded. This task can be solved by creating a specific trigger using the appropriate command CREATE TRIGGER (figure 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 //
```

To check the trigger, it is required to add a new contract with the next statement.

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

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

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

Figure 5.6

The database stores both general supplier information and information that only applies to individuals or legal entities. The simultaneous availability of supplier data in the supplier_org and supplier_person tables is not allowed in terms of business logic. Thus, there is a need for complex control of the relations of referential integrity. To solve this problem we will create a trigger which, when entering the information in the supplier_person table, will control the availability of the code of the respective supplier in the supplier_org table and

block the input of the supplier's data as an individual in case if there is already available data on the given supplier as a legal entity (figure 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 //
```

To check the trigger, you must try to add data about supplier 2 (which is already stored in the database as a legal entity) as an individual.

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

```
XAMPP for Windows - mysql  -u root -p
MariaDB [supply]> INSERT INTO supplier_person VALUES (2, 'Makarov', 'Oleg', 'Petrovych');
ERROR 1644 (45001): The person with id 2 is already stored as the organization!
MariaDB [supply]> 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)
```

Figure 5.7

To delete stored procedures and triggers, it is required to use the DROP PROCEDURE and DROP TRIGGER operators respectively.

3. Make a report on laboratory work

The report should include the main stages of laboratory work and screenshots that demonstrate them.

4. Questions

1. What is a stored procedure?
2. Name the advantages of stored procedures.
3. What operator is used to create a stored procedure?
4. How to define input or output parameters of a stored procedure?

5. What is the purpose of the IF operator?

6. What is the purpose of BEGIN and END operators?

7. What is a trigger?

8. Name the advantages of triggers.

9. Which operator is used to bind a trigger to a table?

10. Which events related to the table modification operations might be processed with triggers?

11. How to define before or after the table modification operation a trigger should be executed?

12. What are the prefixes NEW and OLD used for?

13. What is the operator SET used for?

14. Which operators are used to remove stored procedures and triggers?

Laboratory work 6

BASICS OF DATA INTEGRITY CONTROL MECHANISMS

Goal: learn how to use the referential integrity control mechanisms using the MySQL database.

Progress

Warning! It is recommended to create the temporary database using the queries shown in the laboratory work 2. Use this temporary database in this laboratory work.

1. Learn the features of the referential integrity control mechanism NO ACTION

Let's consider the features of the referential integrity mechanism NO ACTION on the example of the relationship between supplier and contract tables, supplier and supplier_person, supplier and supplier_org. These tables are linked by the supplier_id field. In this regard, the supplier table is parent, and the tables contract, supplier_org, supplier_person are child tables. In order to learn the features of the mechanism of referential integrity, the following sequence of statements must be executed.

Set the ON DELETE and ON UPDATE parameters that determine the behavior when deleting and updating entries from the parent table.

```
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;
```

Assume that due to certain reasons, it is required to remove the vendor with code 4 (figure 6.1).

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

```
mysql -u root -p
MariaDB [supply_1]> DELETE FROM supplier WHERE supplier_id = 4;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (
`supply_1`.`supplier_org`, CONSTRAINT `supplier_org_ibfk_1` FOREIGN KEY (`supplier_id`) RE
FERENCES `supplier` (`supplier_id`) ON DELETE NO ACTION)
```

Figure 6.1

Therefore, in order to remove this vendor, you must first delete all the data associated with it. To do this, delete the corresponding entry from the supplier_org table and check the availability of contracts with this supplier in the contract table. If there are such contracts, they should also be deleted (it should be kept in mind that there may be a need to remove and a content of these contracts). After that, you need to try to remove the vendor with code 4 again. If there is no data associated with it, the vendor will be deleted.

Suppose that for some reason there was a need for a vendor with code 5 to change the code to 7 (figure 6.2).

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

```
mysql -u root -p
MariaDB [supply_1]> UPDATE supplier SET supplier_id = 7 WHERE supplier_id = 5;
ERROR 1451 (23000): Cannot delete or update a parent row: a foreign key constraint fails (
`supply_1`.`supplier_person`, CONSTRAINT `supplier_person_ibfk_1` FOREIGN KEY (`supplier_i
d`) REFERENCES `supplier` (`supplier_id`) ON DELETE NO ACTION)
```

Figure 6.2

Since the contracts with this supplier are not available, the link to it is only in the supplier_person table. After deleting this entry, you must repeat the vendor code change from 5 to 7. Now, this operation must be successful. After that, you need to check the contents of the tables.

2. Learn the features of the referential integrity control mechanism CASCADE

Change the referential integrity mechanisms for links between all the above tables to the CASCADE.

```sql
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;
```

Suppose that for some reason there was a need for a supplier with code 2 to change the code to 8 (figure 6.3).

```sql
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)
```

Figure 6.3

Check for the appropriate changes in the supplier_org table.

Now assume that this supplier (which now has code 8) must be removed (figure 6.4).

```sql
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`.`supplied`, CONSTRAINT `supplied_ibfk_1` FOREIGN KEY (`contract_number`) REFERE
NCES `contract` (`contract_number`))
```

Figure 6.4

Determine the reason why entries were not deleted. Make the necessary changes in the referential integrity mechanisms of the required tables in order to ensure that the necessary data has still been deleted.

3. Learn the features of the referential integrity control mechanism SET NULL

Consider the features of the SET NULL referential integrity mechanism, e.g., for the supplier and contract tables.

Change the referential integrity mechanisms for links between all the above tables to the 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;
```

In the supplier table, change supplier code 3 to 10. Check the data in the contract table (figure 6.5).

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

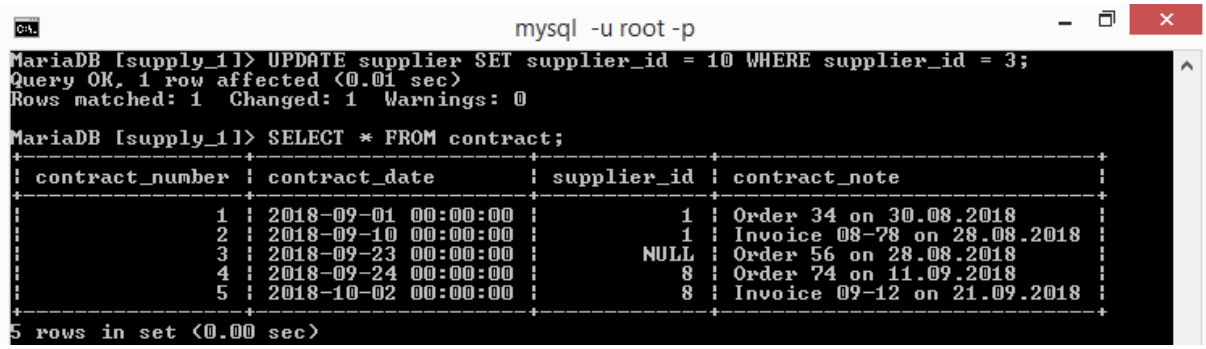

Figure 6.5

Instead of NULL set the value of the supplier code 10 for the contract with number 3.

4. Make a report on laboratory work

The report should include the main stages of laboratory work and screenshots that demonstrate them.

5. Questions

1. Are the ON DELETE and ON UPDATE commands are necessary for the CRATE TABLE or ALTER TABLE commands?

2. What behavior of a database the ON DELETE command defines?

3. What behavior of a database the ON UPDATE command defines?

4. Which parameters might be defined after the ON DELETE and ON UPDATE statements?

5. Name features of the referential integrity mode CASCADE.

6. Name features of the referential integrity mode SET NULL.

7. Name features of the referential integrity mode NO ACTION.

8. Name features of the referential integrity mode SET DEFAULT.

9. Name features of the referential integrity mode RESTRICT.

10. Why the referential integrity mechanism SET DEFAULT has not been considered in this laboratory work?

11. How to set a certain referential integrity mechanism for a foreign key of a table?

12. Why the supplier_id field of the contract table was modified before the SET NULL mode is set?

13. In which cases it is not recommended to used the referential integrity mechanism CASCADE?

14. Which referential integrity mode is always used by default in the MySQL database in case if it was not defined using the ON DELETE and ON UPDATE statements?

Laboratory work 7

WORK WITH TRANSACTIONS

Goal: learn the basics of the transactional mechanism using the MySQL database.

Progress

Warning! It is recommended to create the temporary database using the queries shown in the laboratory work 2. Use this temporary database in this laboratory work.

1. Create a query that demonstrates usage of transactions to add data into a single table

Consider the sequence of actions when creating and using a query that triggers a transaction, a new entry is added to the table, and then the situation of the incorrect or correct completion of the transaction is simulated. The table status is controlled before the transaction begins, during the execution of the transaction and after it is completed. To do this you need to do the following sequence of actions.

```
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;
```

The SELECT queries can output data that illustrates the state of the table before the transaction begins (figure 7.1), during the execution of the transaction, and after the transaction is completed.

```
mysql  -u root -p
MariaDB [supply_1]> 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 | TV                |       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)
```

Figure 7.1

As can be seen from the data shown, a new entry in the table appears (figure 7.2), and then disappears (figure 7.3).

```
mysql  -u root -p
MariaDB [supply_1]> 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 | TV                |       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)
```

Figure 7.2

```
mysql  -u root -p
MariaDB [supply_1]> ROLLBACK;
Query OK, 0 rows affected (0.00 sec)

MariaDB [supply_1]>
MariaDB [supply_1]> 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 | TV                |       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)
```

Figure 7.3

Now it is necessary to consider the situation of the correct completion of the transaction. To do this, in the text of the query, you need to change the ROLLBACK statement to COMMIT. Perform the statement and analyze the results.

2. Create a query that demonstrates usage of transactions to add data into multiple tables

Consider the sequence of actions when creating and using a query that triggers a transaction, and then creates a new supplier, a contract for the supply is concluded with that supplier, the products delivered under this contract. The situation of the incorrect or correct completion of the transaction is simulated. The status of the tables is controlled before the transaction begins, in the process of executing the transaction and after the transaction is completed. To do this you need to do the following sequence of actions.

```
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;
```

The SELECT queries allow you to output data that illustrates the status of the tables before the transaction begins, in the process of executing the transaction and after the transaction is completed. As can be seen from the data shown, new entries in the tables appear and then disappear.

Now it is necessary to consider the situation of the correct completion of the transaction. To do this, change the ROLLBACK statement to COMMIT. Perform the query and analyze the results.

3. Create a query that demonstrates usage of transactions to update data in multiple tables

Consider the sequence of actions when creating and using the query that triggers the transaction, then the data entered in the table are changed when the previous request is executed. The situation of the incorrect or correct completion of the transaction is simulated. The status of the tables is controlled before the transaction begins, in the process of executing the transaction and after the transaction is completed. To do this you need to do the following sequence of actions.

```
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;
```

The SELECT queries allow you to output data that illustrates the status of the tables before the transaction begins, in the process of executing the transaction and after the transaction is completed. As can be seen from the data shown, new entries in the tables appear and then disappear.

Now it is necessary to consider the situation of the correct completion of the transaction. To do this, change the ROLLBACK statement to COMMIT. Perform the query and analyze the results.

4. Create a query that demonstrates usage of transactions to delete data from multiple tables

Consider the sequence of actions when creating and using a query that triggers a transaction that removes the supplier that was created when query 2 was executed and whose data was modified by query 3. Considering the CASCADE referential integrity control mechanism the data will be deleted in several tables. The situation of the incorrect or correct completion of the

transaction is simulated. The status of the tables is controlled before the transaction begins, in the process of executing the transaction and after the transaction is completed. To do this you need to do the following sequence of actions.

```
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;
```

The SELECT queries allow you to output data that illustrates the status of the tables before the transaction begins, in the process of executing the transaction and after the transaction is completed. As can be seen from the data shown, new entries in the tables appear and then disappear.

Now it is necessary to consider the situation of the correct completion of the transaction. To do this, change the ROLLBACK statement to COMMIT. Perform the query and analyze the results.

5. Make a report on laboratory work

The report should include the main stages of laboratory work and screenshots that demonstrate them.

6. Questions

1. What is a transaction?
2. Which table types support transactions in the MySQL DBMS?
3. Which table types do not support transactions in the MySQL DBMS?
4. How to disable transaction auto commit in the MySQL DBMS?
5. What operator is used to complete a transaction?
6. What operator is used to cancel changes performed by a transaction?
7. Which command of the MySQL DBMS should be used to enable the transaction auto commit mode for a certain sequence of statements?

8. With which type of tables the SAVEPOINT and ROLLBACK TO SAVEPOINT operators might be used?

9. What is the purpose of the SAVEPOINT and ROLLBACK TO SAVEPOINT operators?

10. Which issues might be caused by parallel execution of transactions?

11. What are the levels of transaction isolation and what problems can each of these levels solve?

12. What table type is used in the MySQL database by default (starting from the version 5.5)?

13. Which transaction isolation levels are supported by InnoDB?

14. Which transaction isolation level is set by default in the InnoDB engine?

Laboratory work 8

USER RIGHTS MANAGEMENT

Goal: learn basics of user accounts and privileges using the MySQL database.

Progress

1. Create new user accounts

The MySQL database management system is a multi-user environment, so different accounts with different privileges can be created to access the supply database tables.

The supply manager's account can be provided with privileges to view the supplier, supplier_org, supplier_person and contract tables, add new records, delete and update existing records in the data tables.

Database administrator can be given wider rights (privileges to create tables, editing and deleting existing tables, creating and editing user accounts, etc.).

For a warehouse employee it is enough just to view the contract and supplied tables, as well as add new records, delete and update already existing records in the supplied table.

Consider creating accounts for different database users.

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

These statements allow to create accounts for the following users:

1) administrator with the password «admin123»;

2) supply manager with the password «manager123»;

3) warehouse employee with the password «storekeeper123».

The DROP USER statement is used to delete an account. The change of user name in the account is performed with the operator RENAME USER %old_name% TO %new_name%.

Since all user accounts are stored in the mysql system user's table, you can check the creation of the accounts by using the following query (figure 8.1):

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

```
XAMPP for Windows
MariaDB [(none)]> SELECT Host, User, Password FROM mysql.user;
+-----------+----------------+-------------------------------------------+
| Host      | User           | Password                                  |
+-----------+----------------+-------------------------------------------+
| localhost | root           |                                           |
| 127.0.0.1 | root           |                                           |
| ::1       | root           |                                           |
| localhost |                |                                           |
| localhost | pma            |                                           |
| %         | supply_manager | *D3EA2B50EA2CDB63852452342425A884B6C6A8DC |
| localhost | supply_manager | *D3EA2B50EA2CDB63852452342425A884B6C6A8DC |
| localhost | manager        | *1B2333B70420F3DB5F4F164A9B89E21810F06840 |
| localhost | admin          | *01A6717B58FF5C7EAFFF6CB7C96F7428EA65FE4C |
| localhost | storekeeper    | *6A8DA8D9B9189005A0B1791874632DFD2DDD7DFA |
+-----------+----------------+-------------------------------------------+
10 rows in set (0.00 sec)
```

Figure 8.1

2. Assign privileges for created accounts

The above operators allow you to create, delete, and edit accounts, but they do not allow you to change user privileges – to tell the MySQL DBMS, which user is only allowed to read information or to read and edit, and who are given the rights to change the structure of the database and create accounts.

It is required to assign privileges for the created accounts.

```
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';
```

The REVOKE operator is used to deprive the user of certain privileges. This operator does not delete accounts, but only cancels the previously granted privileges. Therefore, for the final removal of the account, you must use the operator DROP USER.

Check the privileges of the admin account granted with all rights at the supply database level using the following query (figure 8.2).

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

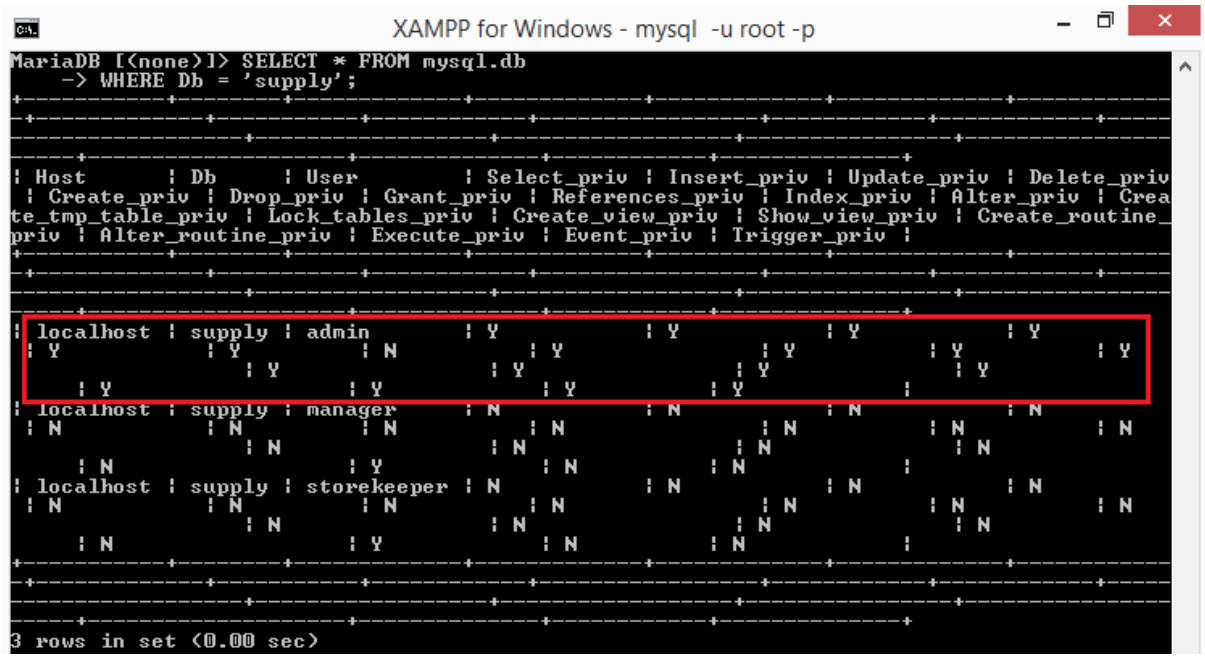

Figure 8.2

Similarly, you can check the privileges of the manager and storekeeper accounts, for which certain restrictions were encountered with the supply database tables (figure 8.3).

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

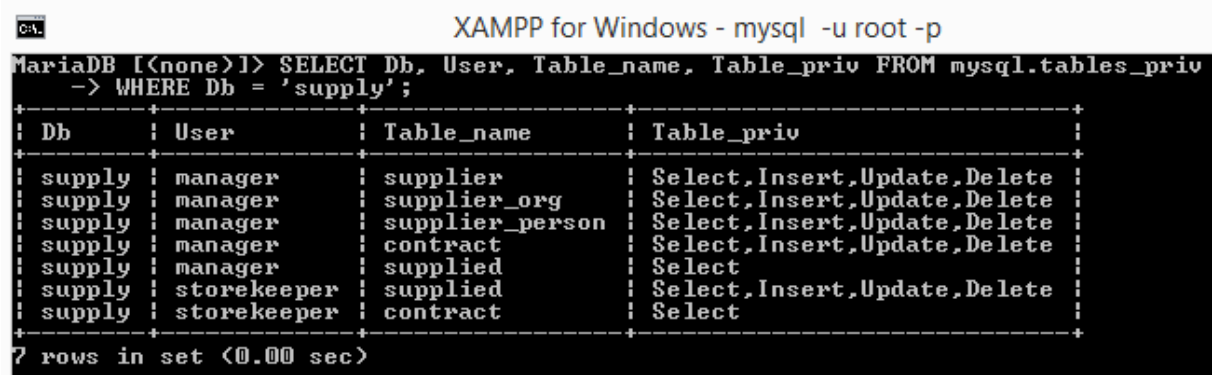

Figure 8.3

In addition, certain users must also have privileges that allow them to use the views contained in the supply database. For example, the manager user should be given permissions to view the contract_supplier and supplier_info views, whereas only the contract_supplier view for the user storekeeper should be available.

3. Make a report on laboratory work

The report should include the main stages of laboratory work and screenshots that demonstrate them.

4. Questions

1. What is the structure of a user account in the MySQL DBMS?
2. Which components a user account is contains of?
3. What is the purpose of each component of a user account?
4. How to view all user accounts?
5. What command is used to create a user account?
6. What command is used to delete a user account?
7. How to change a name of a user?
8. What statement is used to define certain privileges for a certain user account?
9. What operator is used to cancel given privileges?
10. Which privileges might be defined for a user account?
11. What levels of privileges do you know?
12. How to check the global privileges, database privileges, and table privileges?

Laboratory work 9

DATABASE APPLICATION DEVELOPMENT

Goal: learn the basics of the database application development using the MySQL DBMS and PHP programming language.

Progress

Warning! This laboratory work demonstrates development of just a simple part of the whole database application.

1. Define the basic functionality of an application

The basic functionality of a web application fragment that is designed to work with the supply database is presented in the form of a UML use-case diagram (figure 9.1).

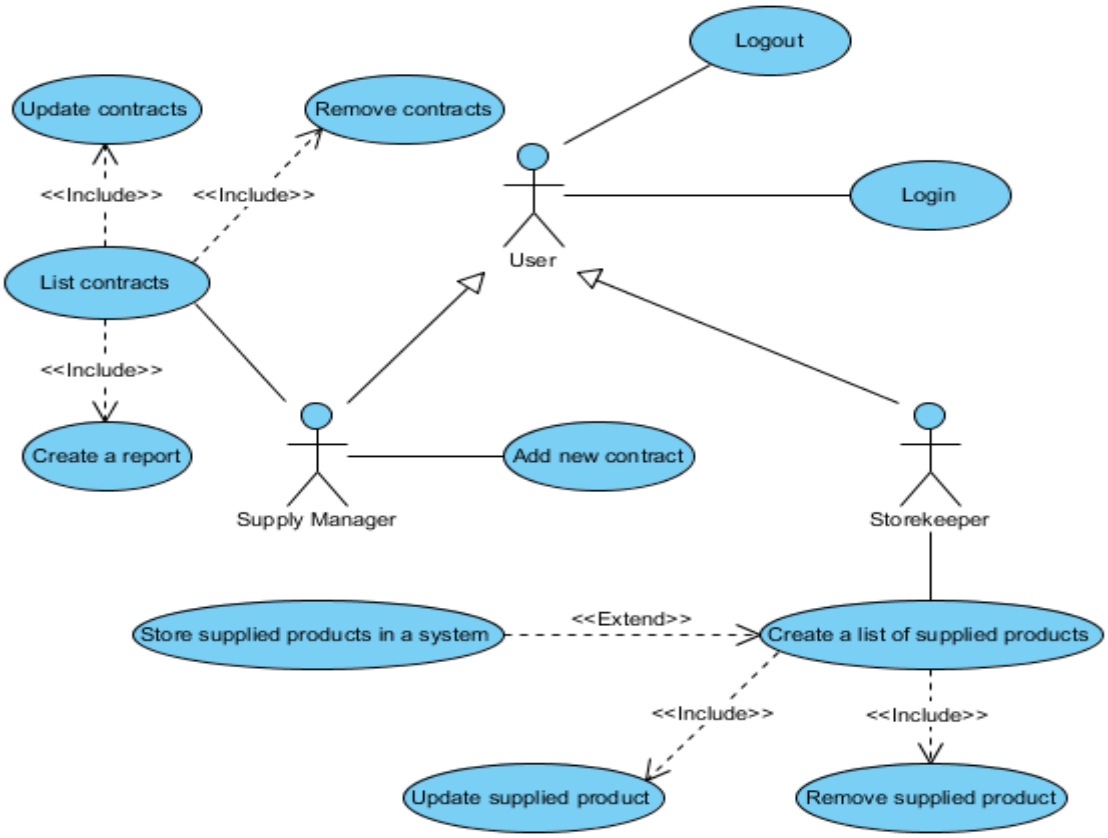

Figure 9.1

2. Develop a page for application users' authorization

All pages of the web application must be placed in the directory xampp/htdocs/supply.

Before you begin creating an authorization page, you need to develop the functionality of the software to establish a connection to the database. To do this, create a connect.php file with the following content.

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

In addition, you need to develop a main page of the web application, which is to create an index.php file with the following content.

```php
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      # redired to login page if the user is not set
14      header("location: login.php");
15  }
16  ?>
```

Lines 2-4 contain the start of a user session and connect a file that contains the function db_conn() to establish a connection to the database. Lines 9 through 15 include checking for a user session and connecting to the database. If the user was not authorized, it will be redirected to the authorization page

(line 14). Line 11 defines a file connection that includes the processing of forms for adding, updating and deleting data; it will be created later.

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

The following lines (17 – 39) determine the appearance of the main page: information about the current user (figure 9.2), the content of the page according to the type of user, disconnecting the database connection (line 39). Line 24 specifies a link that allows you to delete all session variables and finish the session. To do this, use the logout.php file.

User: *manager* | Logout

Figure 9.2

```
<?php
session_start();

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

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

The user authorization page is stored in the login.php file.

```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 ?>
```

Lines 9 and 10 define session record entries that contain user account information. These variables are used in the connect.php file to connect to the database using the mysqli_connect () function. If the user session has already been set, it will be redirected to the index.php homepage (lines 15 – 17).

```html
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>
```

The lines 20 - 45 define the static structure of the user authorization page, which contains the corresponding form with the necessary elements of the user interface (figure 9.3).

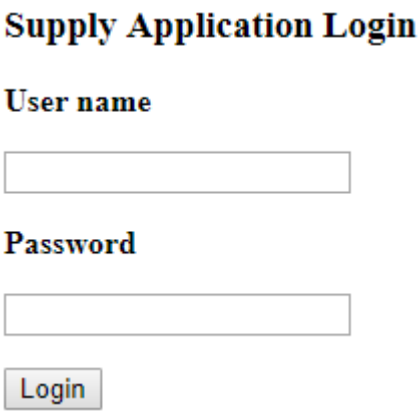

Figure 9.3

3. Develop software functionality for the supply manager

The page containing the software functionality for the supply manager work is contained in the manager.php file.

```
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>
```

Lines 1 through 24 contain a check on the availability of a custom session, as well as the mode of working with data on contracts (creation, update or deletion), which depends on the interface element – the New contract link, designed to create a new contract (figure 9.4), or Back – for return to viewing data on all contracts (figure 9.5).

Contracts

New contract

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

Figure 9.4

Back

Supplier

Petrov Pavlo Petrovych

Note

Save

Figure 9.5

Lines 26 to 99 include checking the modes of creating a new record (figure 9.5), updating (figure 9.6), or deleting an existing record (figure 9.7) and displaying the corresponding forms with certain elements of the user interface.

```
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

Figure 9.6

Back

Delete the contract #13?

Continue

Figure 9.7

Lines 100 – 133, in turn, define a table with data about contracts and corresponding links (Action column), intended for manipulation of these data (figure 9.4).

Lines 135 – 179 contain the definition of an additional table designed to display the list of delivered goods under a specific contract (figure 9.8). To demonstrate this table, the necessary check of the data view of contracts is performed (lines 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
1	2018-09-01 00:00:00	Petrov Pavlo Petrovych	Order 34 on 30.08.2018	Update Delete
2	2018-09-10 00:00:00	Petrov Pavlo Petrovych	Invoice 08-78 on 28.08.2018	Update Delete
3	2018-09-23 00:00:00	Ivanov Illia Illych	Order 56 on 28.08.2018	Update Delete
4	2018-09-24 00:00:00	Interfruit Ltd.	Order 74 on 11.09.2018	Update Delete
5	2018-10-02 00:00:00	Interfruit Ltd.	Invoice 09-12 on 21.09.2018	Update Delete
7	2018-12-27 13:30:04	Petrov Pavlo Petrovych		Update Delete
13	2019-01-10 13:20:48	Transservice LLC	Order #9876	Update Delete

Supplied products by contract #4

Hide

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

Figure 9.8

4. Develop software functionality for the warehouse employee

The page containing the software functionality for the storekeeper's work is contained in the storekeeper.php file.

Lines 1 through 14 contain a check for the presence of a custom session, as well as the presence of a session variable, an array to which goods that are put into the warehouse but not yet stored in a database are recorded.

```
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>
```

In rows 16 – 72 the table of products supplied to the warehouse is determined.

```
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      # chech 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>
```

In this case, checking the presence of products in the array (session variable) and the output of the form (figure 9.9), which allows you to record the received goods in the database (lines 27 – 67) is performed.

```
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        ?>
```

Also, the presence of expected deliveries is checked (if there are so-called “empty” contracts that have been concluded, but for which no goods have been delivered yet) in lines 17 – 24. In the case of such contracts, a form (figure 9.10) is displayed for adding the supplied product (lines 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	Remove
Camera	30	1200	Remove
Watch	200	399.99	Remove

Store products

Figure 9.9

New product

Product	Amount	Cost
Bluetooth Speaker	99	120

Add product

Figure 9.10

5. Develop a functionality to generate an Excel report that will display supplies over a given period

The implementation of this functionality will also be located in the file action.php, which contains the processing of user forms.

Lines 1 through 34 of this file contain forms processing, which are intended to create contract records, as well as update and delete existing records. It should be noted that in order to perform operations for creating, updating and deleting entries from the table contract, the created previously stored procedure sp_contract_ops is used.

Lines 36 – 60 process forms that are designed to create a record of the delivered, but not yet stored in the database of the product, as well as the removal of such entries from an array stored as a session user variable.

```php
<?php
# process request to create contract
if (isset($_POST["create_contract"])) {
    $supplier_id = $_POST["supplier_id"];
    $contract_note = $_POST["contract_note"];

    # use the stored procedure created earlier
    $sql = "CALL sp_contract_ops('i', 0, '', {$supplier_id}, '{$contract_note}')";
    mysqli_query($conn, $sql);

    header("location: index.php");
}

# process request to delete contract
if (isset($_POST["delete_contract"])) {
    $contract_number = $_POST["contract_number"];

    $sql = "CALL sp_contract_ops('d', {$contract_number}, '', 0, '')";
    mysqli_query($conn, $sql);

    header("location: index.php");
}

# process request to update contract
if (isset($_POST["update_contract"])) {
    $contract_number = $_POST["contract_number"];
    $supplier_id = $_POST["supplier_id"];
    $contract_note = $_POST["contract_note"];

    $sql = "CALL sp_contract_ops('u', {$contract_number}, CURRENT_TIMESTAMP(), {$supplier_id}, '{$contract_note}')";
    mysqli_query($conn, $sql);

    header("location: index.php");
}
```

```php
# process request to insert new record into session array of delivered products
if (isset($_POST["add_product"])) {
    $supplied_product = $_POST["supplied_product"];
    $supplied_amount = $_POST["supplied_amount"];
    $supplied_cost = $_POST["supplied_cost"];

    if (!empty($supplied_product) && !empty($supplied_amount) && !empty($supplied_cost)) {
        if (is_numeric($supplied_amount) && is_numeric($supplied_cost)) {
            if ($supplied_amount > 0 && $supplied_cost > 0) {
                $_SESSION["supplied_products"][$supplied_product] = array("amount" => $supplied_amount,
                    "cost" => $supplied_cost);
            }
        }
    }

    header("location: index.php");
}

# process request to remove a record from the session array
if (isset($_GET["supplied"]) && $_GET["supplied"] == "remove") {
    $supplied_product = $_GET["product"];
    unset($_SESSION["supplied_products"][$supplied_product]);

    header("location: index.php");
}
```

Lines 62 – 103 demonstrate the preservation of delivered goods to the database. It should be noted that the creation of records about goods delivered under a specific contract in the table supplied is carried out inside the transaction, because the partial (due to any circumstances) transfer of data received from the session variable to the operational database is not acceptable.

```
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      }
```

```
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 ?>
```

The code of the file action.php should be supplemented with the following fragment, which is intended to create and save an Excel document with a report on volumes of supplied products for a certain period. To create a report, the previously saved stored procedure sp_contract_total will be used.

The contents of the manager.php file must be supplemented with a link (figure 9.11), which will allow to generate and download the report (line 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>
```

In addition, the action.php file must be supplemented by a code (lines 104 – 127), designed directly to generate and download the report (figure 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", $str);
130
131     if (strstr($str, '"')) {
132         $str = '"' . str_replace('"', '""', $str) . '"';
133     }
134 }
135 ?>
```

Contracts

New contract Export data

Figure 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		

Figure 9.12

6. Make a report on laboratory work

The report should include the main stages of laboratory work and screenshots that demonstrate them.

7. Questions

1. Develop software functionality for the supply database administrator. The administrator should be able to create, modify, and remove records in all database tables.

2. Add functionality used to sort rows in the Contracts table (manager.php file) in both ascending and descending order:

– by the contract number;

– by the contract date.

3. Add functionality used to sort rows in the Supplied products by contract #X table (manager.php file) in both ascending and descending order:

– by supplied product name;

– by supplied product amount;

– by supplied product cost.

4. The form used to update data about a certain contract includes the combo box with the list of suppliers. Modify the application in order to after the form is loaded, the supplied assigned to a current contract will be selected in this combo box.

5. It is impossible to remove the contract with the assigned supplied products due to the used referential integrity mode. Modify the software (e.g., by modifying the stored procedure sp_contract_ops) in order to allow deleting data about contracts even if there are products supplied by a contract you are trying to remove.

6. It is impossible to remove the contract with the assigned supplied products due to the used referential integrity mode. Modify the software (e.g., by modifying the stored procedure sp_contract_ops) in order to deny deletion of data about “not empty” contracts.

7. As it is shown in figure 9.12, the column titles in the generated report are not user-friendly; especially the columns that contain aggregated data. Modify the application in order to assign the Contract, Date, Total amount, and Total cost titles for corresponding columns.

8. Current implementation allows to generate report (figure 9.12) based on the fixed range of dates – starting from the 01/01/2018 to the time of report generation. Modify the application in order to user would be able to set the required range of dates.

9. Provide the supply manager with the ability to work with data about suppliers (add records, update and delete existing records). Ensure that it is possible to check the list of contracts concluded with a certain supplier.

10. Add functionality of automatic generation of the invoice document just after the list of products supplied according to a certain contract is saved into the operational database by the storekeeper.