МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

GUIDELINES
for laboratory works
on the topic «Learning the basics of working with DBMS MySQL: Fundamental DDL and DML tools of SQL language»

for students of specialties:
121 «Software engineering»
122 «Computer science»
126 «Information systems and technologies»

Харків 2021

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ХАРКІВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»

GUIDELINES
for laboratory works
on the topic «Learning the basics of working with DBMS MySQL: Fundamental DDL and DML tools of SQL language»

for students of specialties:
121 «Software engineering»
122 «Computer science»
126 «Information systems and technologies»

Затверджено
Редакційно-видавничою
радою університету,
протокол № 2 від 29.06.2021

Харків
НТУ «ХПІ»
2021

Методичні вказівки до виконання лабораторних робіт за темою «Вивчення основ роботи з СУБД MySQL: Основні засоби DDL та DML мови SQL» для студентів спеціальностей 121 «Інженерія програмного забезпечення», 122 «Комп'ютерні науки» та 126 «Інформаційні системи та технології» / уклад. Д.Л. Орловський, А.М. Копп. – Харків: НТУ «ХПІ», 2021. – 29 с. – Англ. мовою.

Укладачі: Д.Л. Орловський

А.М. Копп

Рецензент Малько М.М.

Кафедра програмної інженерії та інформаційних технологій управління

CONTENTS

INTRODUCTION

MySQL Database Management System (DBMS) is a freely distributed relational database developed and maintained by Oracle. From the very beginning, MySQL was developed by the Swedish company MySQL AB, which was later acquired by Sun Microsystems, which, in turn, was later acquired by Oracle.

MySQL is distributed under both the GNU General Public License (GPL) and its commercial license. Under the terms of the GPL, software that uses MySQL libraries must also be distributed under the GPL license. For cases where developers do not want to open the source code of their software, a commercial license is provided. The advantage of a commercial license is quality service support. Contrary to Oracle's MySQL licensing policy and to ensure free DBMS status, a fork of MySQL was created and called MariaDB. This database supports high compatibility with MySQL, ensuring the exact correspondence of the programming interface, the so-called API (Application Programming Interface), and MySQL commands.

MySQL is a great solution for small, medium, and sometimes even large software systems. MySQL is also part of the WAMP (Windows, Apache, MySQL, PHP/Perl/Python) and LAMP (Linux, Apache, MySQL, PHP/Perl/Python) web application development stacks. This database is included in many ready-made assemblies of servers designed for web applications, such as XAMPP (which is proposed for use in this laboratory workshop), OpenServer, Denwer, and more. Recently, however, it is because of openness support, server builders and hosting providers are increasingly incorporating MariaDB into WAMP and LAMP stacks.

Typically, MySQL is used as a server accessed by local or remote clients. However, the distribution also contains a library that provides the deployment of an internal server for standalone applications. In this laboratory workshop, we get acquainted with the main features of MySQL in the role of the client-server database.

LABORATORY WORK 1. DATABASE 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).

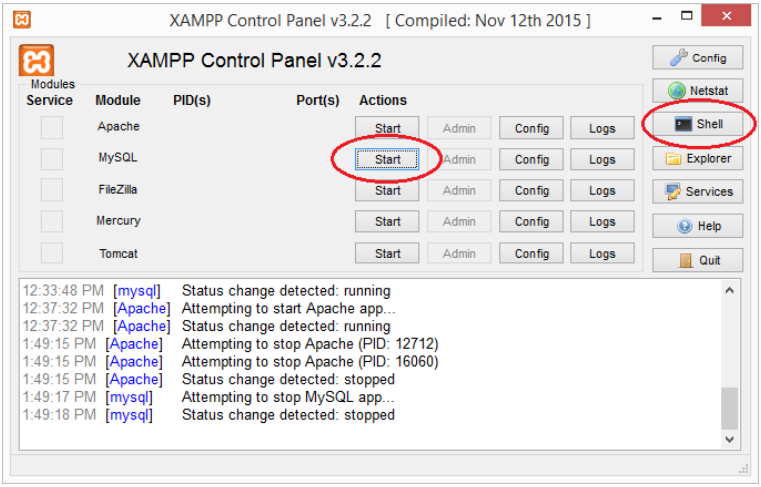


Figure 1.1 – XAMPP control panel

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.

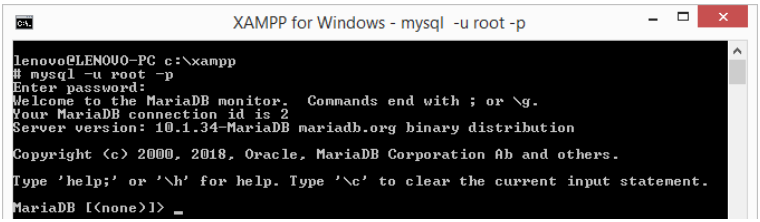


Figure 1.2 – MySQL connection

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 creating a database, the work to be considered in the laboratory practice. You can check the database created 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. The 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).

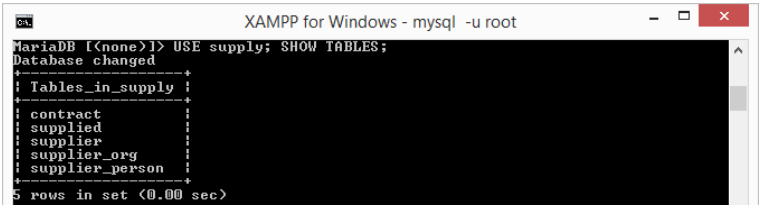


Figure 1.3 – Created database tables

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;
```

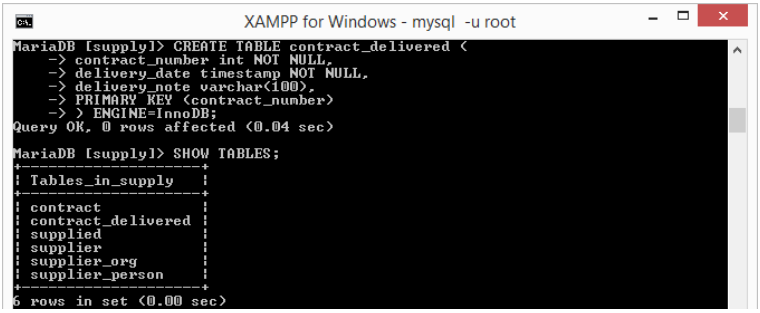


Figure 1.4 – Check existence of the created table

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);
```

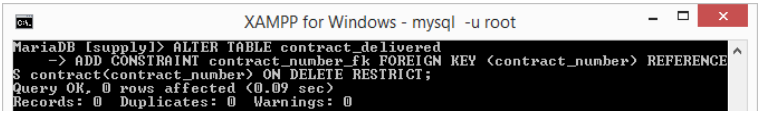


Figure 1.5 – ALTER TABLE command execution

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).

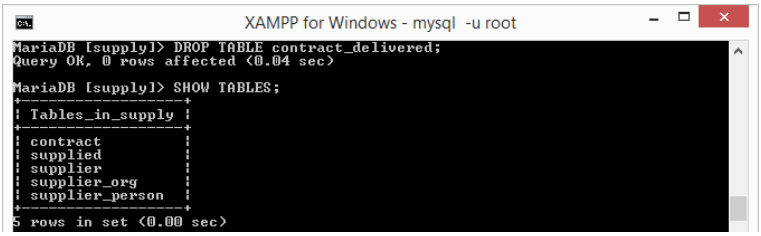


Figure 1.6 – Check that table was deleted

6. Make a report for the 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 the table's structure?
7. Which SQL statement is used to delete tables from a database?
8. How to check the 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 inserting 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).

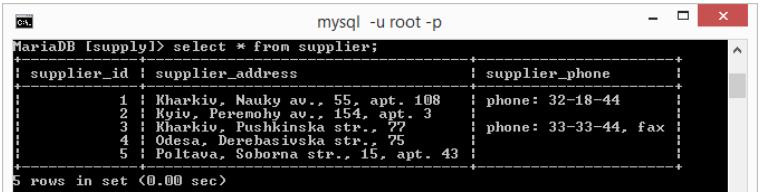


Figure 2.1 – Records created in the supplied table

The following commands allow inserting 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).

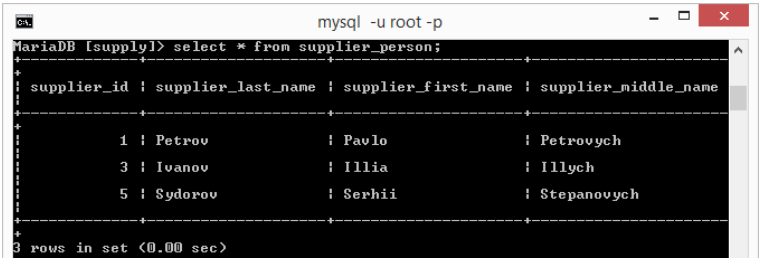


Figure 2.2 – Records created in the supplied_person table

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).

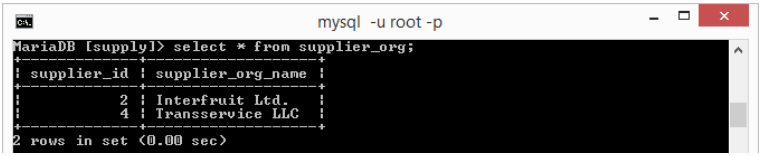


Figure 2.3 – Records created in the supplied_org table

The following commands allow inserting 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).

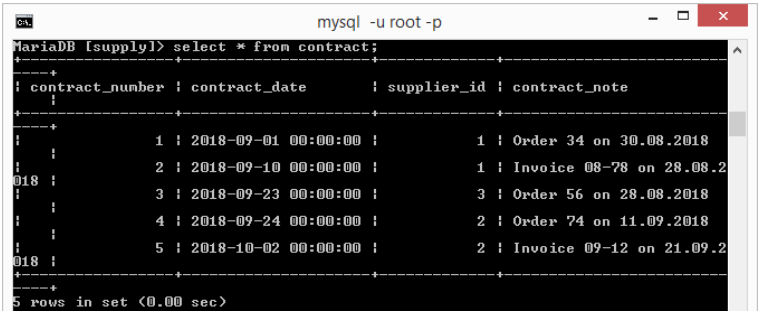


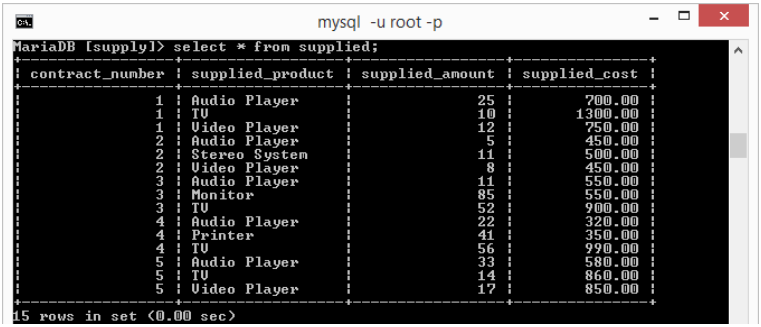
Figure 2.4 – Records created in the contract table

The following commands allow inserting 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).



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

Figure 2.5 – Records created in the supplied table

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';
```

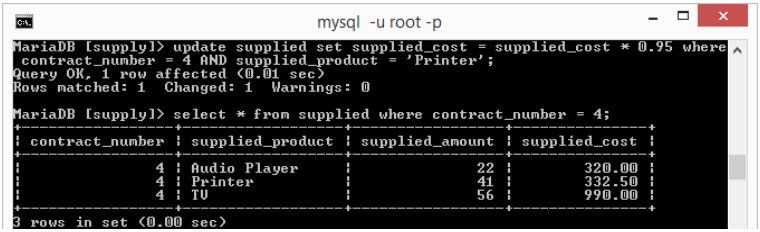


Figure 2.6 – UPDATE query results

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;
```

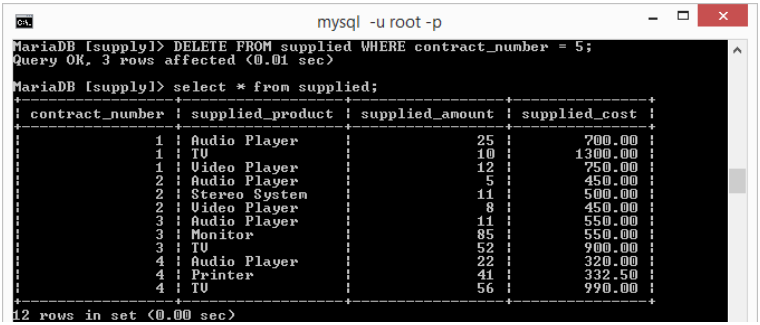


Figure 2.7 – DELETE query results

Restore deleted entries using INSERT commands.

4. Make a report for the 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 products supplied by the specific contract by 15%?

8. Which structure of the INSERT command should be provided 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 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);
```

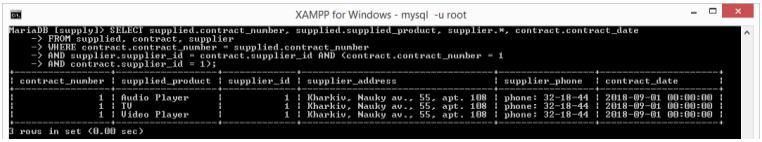


Figure 3.1 – Result of query 1

Query 2

Form a list of the goods delivered by 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;
```

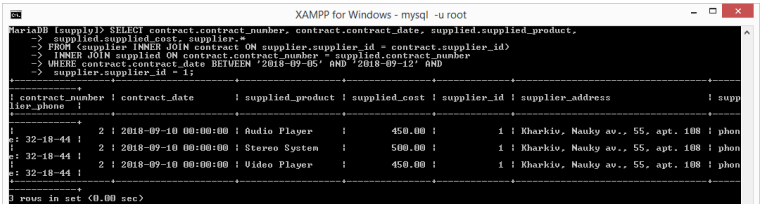


Figure 3.2 – Result of query 2

Query 3

Form a list of goods that were delivered in month 9 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;
```

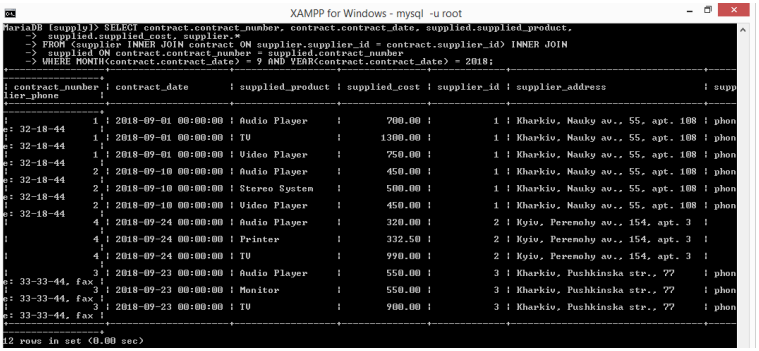


Figure 3.3 – Result of query 3

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;
```

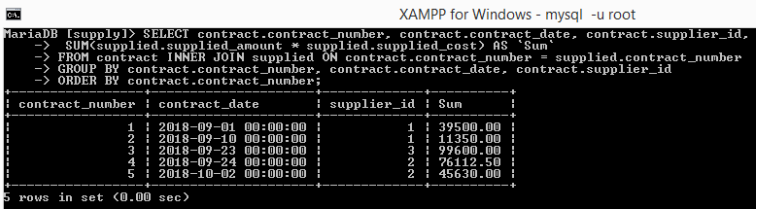


Figure 3.4 – Result of query 4

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 the 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;
```

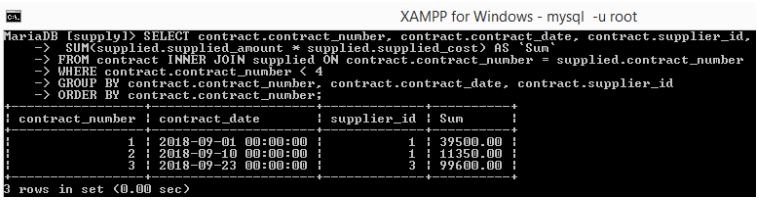


Figure 3.5 – Result of query 5

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);
```

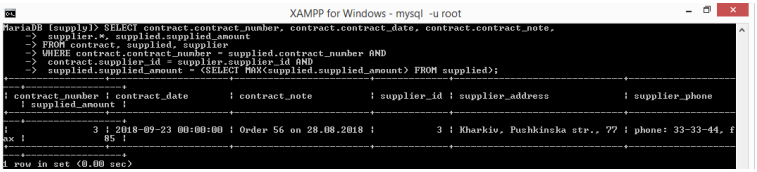


Figure 3.6 – Result of query 6

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);
```

```
XAMPP for Window
MariaDB [supply]> SELECT * FROM supplier
    -> WHERE supplier_id NOT IN (SELECT supplier_id FROM supplier);
Empty set (0.00 sec)
```

Figure 3.7 – Result of query 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 – Result of query 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);
```

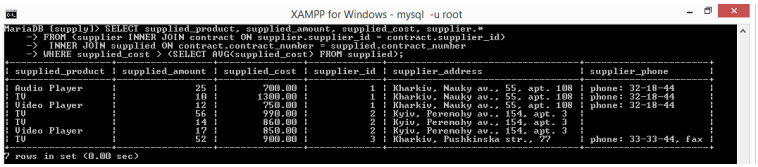


Figure 3.9 – Result of query 9

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;
```

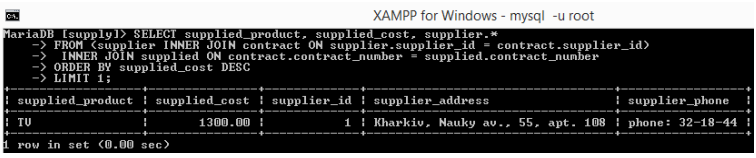


Figure 3.10 – Result of query 10

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;
```

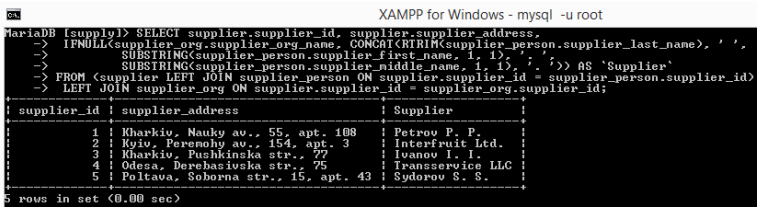


Figure 3.11 – Result of query 11

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 based on which the goods were 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 – Result of query 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;
```

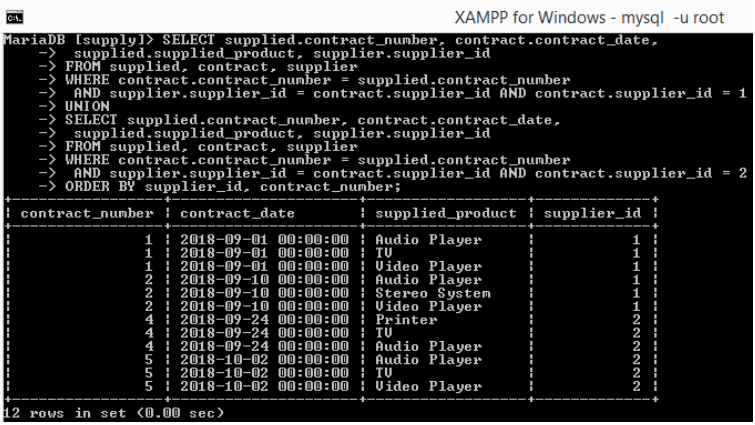


Figure 3.13 – Result of query 13

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;
```

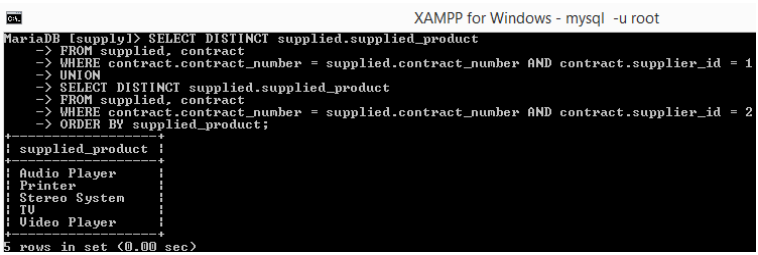


Figure 3.14 – Result of query 14

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;
```

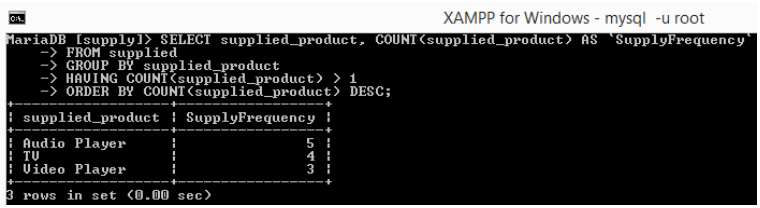


Figure 3.15 – Result of query 15

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;
```

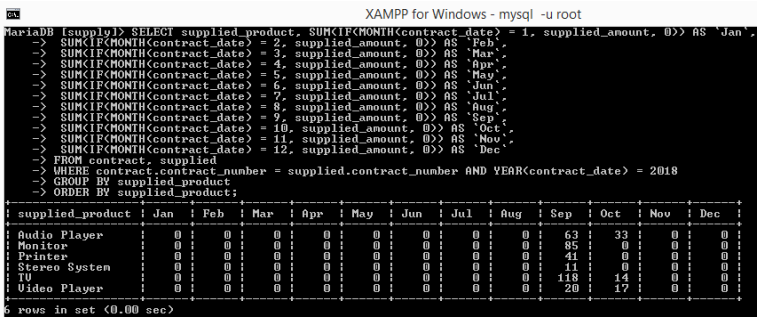
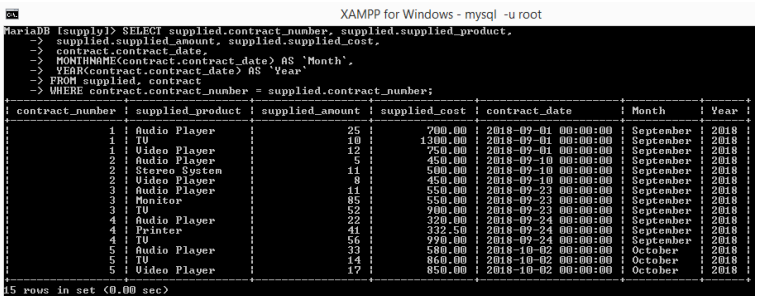


Figure 3.16 – Result of query 16

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 [supply1]> 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 – Result of query 17

2. Make a report for the 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 criteria?
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 between 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 the results of two queries?

LABORATORY WORK 4. CREATION AND USAGE OF VIEWS

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

Progress

1. Create a view that allows seeing 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;
```

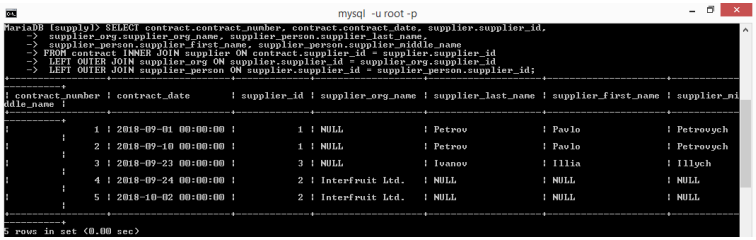


Figure 4.1 – Result of view that allows seeing the name of the supplier when viewing the list of contracts

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;
```

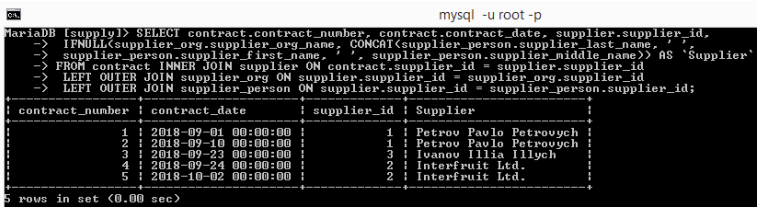


Figure 4.2 – Result of modified query

Now you can create this view with the name contract_supplier using the appropriate SQL statement (figure 4.3).

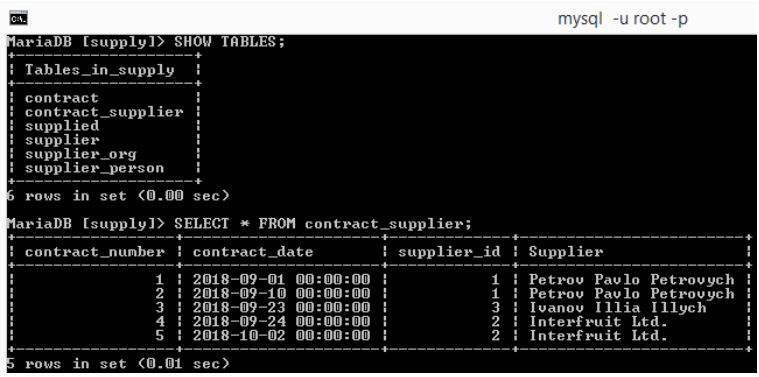


Figure 4.3 – Result of modified view

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;
```

Figure 4.4 – Result of view that allows the user to work with limited supplier data

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

3. Make a report for the 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 the existence of a view in a database?
6. How to specify the list of columns to create a view?
7. What is a vertical view?
8. What is a horizontal view?

REFERENCES

1. SQL Tutorial. URL: https://www.w3schools.com/sql/default.asp (дата звернення: 30.09.2020).

2. SQL. URL: https://www.tutorialspoint.com/sql/index.htm (дата звернення: 30.09.2020).

3. MySQL. URL: https://www.tutorialspoint.com/mysql/index.htm (дата звернення: 30.09.2020).

4. MySQL Documentation. URL: https://dev.mysql.com/doc/ (дата звернення: 30.09.2020).

5. MySQL Tutorial – Learn MySQL Fast, Easy, and Fun. URL: https://www.mysqltutorial.org/ (дата звернення: 30.09.2020).

6. MySQL Tutorial for Beginners Learn in 7 Days. URL: https://www.guru99.com/mysql-tutorial.html (дата звернення: 30.09.2020).

7. MySQL by Examples for Beginners. URL: https://www3.ntu.edu.sg/home/ehchua/programming/sql/MySQL_Beginner.html (дата звернення: 30.09.2020).

8. Drake M. An Introduction to Queries in MySQL. URL: https://www.digitalocean.com/community/tutorials/introduction-to-queries-mysql (дата звернення: 30.09.2020).

9. Мулеса О.Ю. Основи мови запитів SQL. Ужгород, 2015. 48 с.

10. Балик Н.Р., Мандзюк В.І. MySQL: лабораторний практикум. Тернопіль: Навчальна книга–Богдан, 2008. 88 с.

11. Балик Н.Р., Мандзюк В.І. Бази даних MySQL: Навчальний посібник. Тернопіль: Навчальна книга–Богдан, 2010. 160 с.

12. Берко А.Ю., Верес О.М., Пасічник В.В. Системи баз даних та знань. Книга 2. Системи управління базами даних та знань: навч. посібник. Львів : «Магнолія-2006», 2013. 584 с.

13. Ульман Л. MySQL: Руководство по изучению языка. Litres, 2019. 352 с.

14. Фиайли К. SQL: Руководство по изучению языка. Litres, 2019. 456 с.

15. Шварц Б., Зайцев П., Ткаченко В. MySQL по максимуму. 3-е издание. Издательский дом «Питер», 2018. 864 с.

16. Гольцман В.И. MySQL 5.0. Библиотека программиста. Издательский дом «Питер», 2009. 258 с.

17. Кузнецов М.В. MySQL 5. БХВ-Петербург, 2010. 1024 с.

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до виконання лабораторних робіт
за темою «Вивчення основ роботи з СУБД MySQL:
Основні засоби DDL та DML мови SQL»
для студентів спеціальностей
121 «Інженерія програмного забезпечення»
122 «Комп'ютерні науки»
126 «Інформаційні системи та технології»

Англійською мовою

Укладачі:
ОРЛОВСЬКИЙ Дмитро Леонідович
КОПП Андрій Михайлович

Відповідальний за випуск проф. Годлевський М.Д.
Роботу до видання рекомендував проф. Гамаюн І.П.

План 2021 р., поз. 210

Підп. до друку 28.12.2021. Гарнітура Times New Roman.
Ум. друк. арк. 0,4.

Видавничий центр НТУ «ХПІ»,
вул. Кирпичова, 2, м. Харків, 61002
Свідоцтво про державну реєстрацію ДК № 3478 від 21.08.2017 р.

Самостійне електронне видання