9. DATABASE APPLICATION DEVELOPMENT

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

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

9.1. 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).

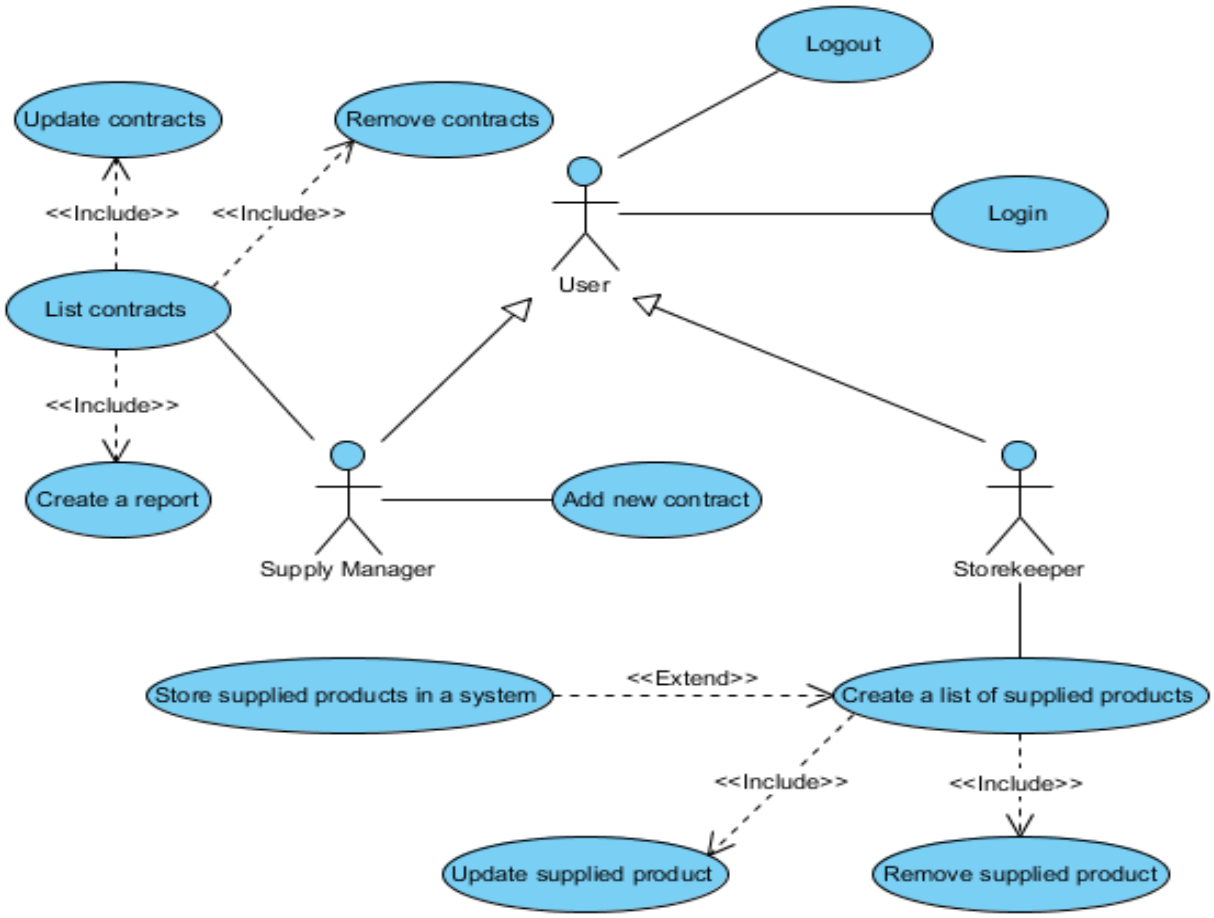

Figure 9.1

9.2. 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
<?php
session_start();

require_once("connect.php");

$conn = NULL;

# check for a user session
if (isset($_SESSION["user"])) {
    $conn = db_conn();
    include("action.php");
} else {
    # redired to login page if the user is not set
    header("location: login.php");
}
?>
```

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
<?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
<?php
session_start();

# process login form
if (isset($_POST["login"])) {
    session_unset();

    # set user session variables
    $_SESSION["user"] = $_POST["user"];
    $_SESSION["pass"] = $_POST["pass"];

    header("location: index.php");
} else {
    # redirect to a home page if user is already signed in
    if (isset($_SESSION["user"])) {
        header("location: index.php");
    }
}
?>
```

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).

```
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).

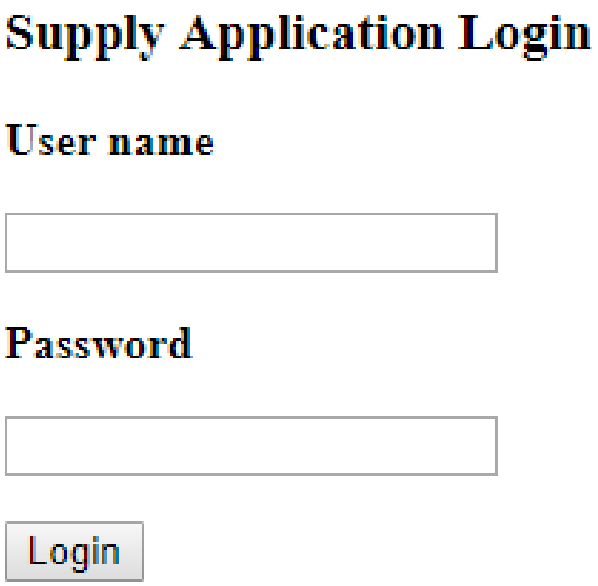

Figure 9.3

9.3. 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

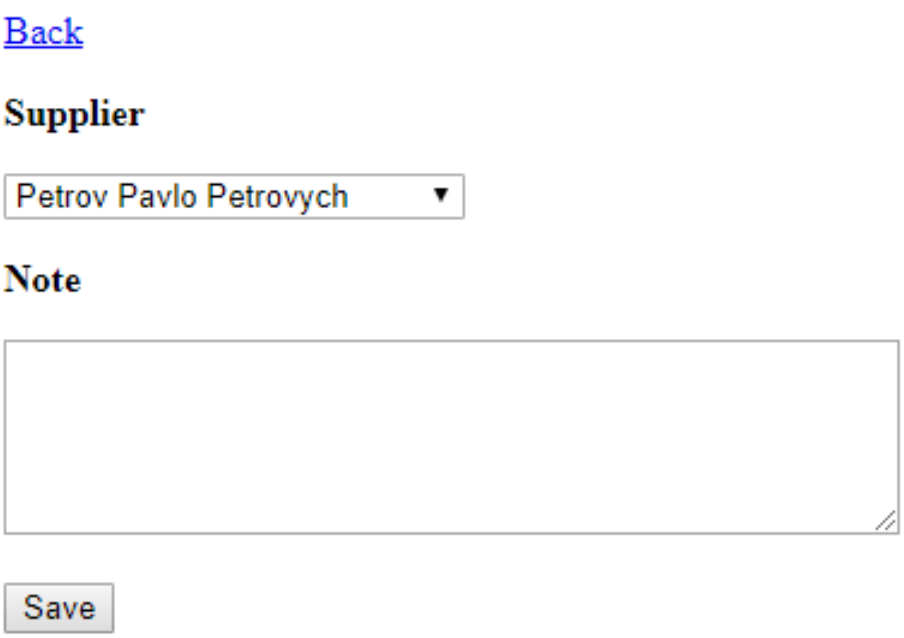

Figure 9.5

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

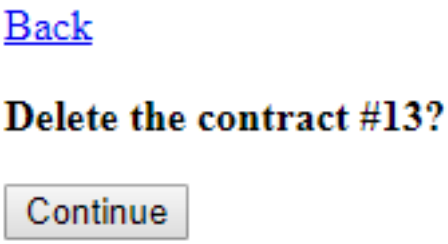

Figure 9.7

Lines 100 – 133, in its 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 }
```

```php
# if the action mode is info
# display data about supplied products for a selected contract
if (isset($_GET["action"]) && $_GET["action"] == "info") {
    $contract_number = $_GET["id"];
?>
    <h3>Supplied products by contract #<?= $contract_number ?></h3>
    <p>
    <a href="index.php">Hide</a>
    </p>
    <?php
    # retrieve data about selected products
    $sql = "SELECT supplied_product, supplied_amount, supplied_cost
        FROM supplied
        WHERE contract_number = {$contract_number}";
    $result = mysqli_query($conn, $sql);

    # check the size of a result set
    if (mysqli_num_rows($result) > 0) {
        ?>
        <table border="1">
            <tr>
                <th>Product</th>
                <th>Amount</th>
                <th>Cost</th>
            </tr>
        <?php
```

```php
            # display products if the contract is not empty
            while ($row = mysqli_fetch_assoc($result)) {
                ?>
                <tr>
                    <td><?= $row["supplied_product"] ?></td>
                    <td><?= $row["supplied_amount"] ?></td>
                    <td><?= $row["supplied_cost"] ?></td>
                </tr>
                <?php
            }
        } else {
            # if the result set is empty print the following message
            echo "Contract is empty";
        }
        ?>
    </table>
<?php
}
?>
```

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

9.4. 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 ?>
```

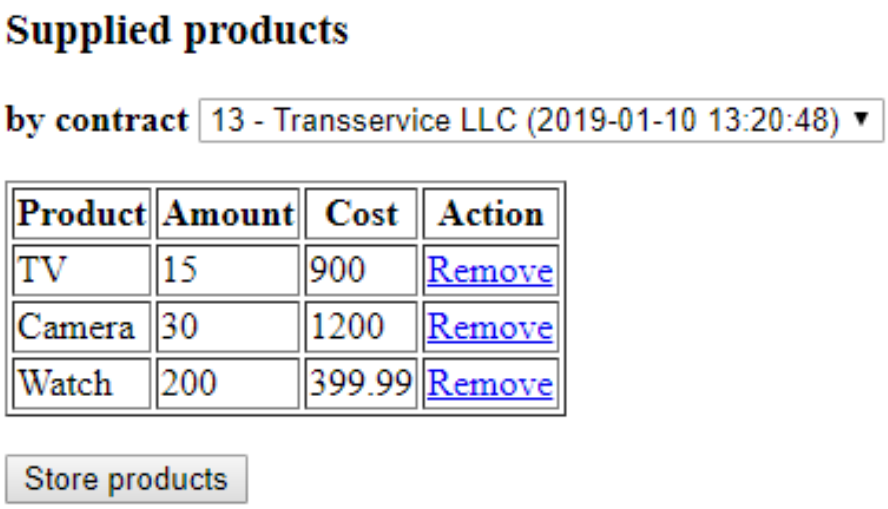

Figure 9.9

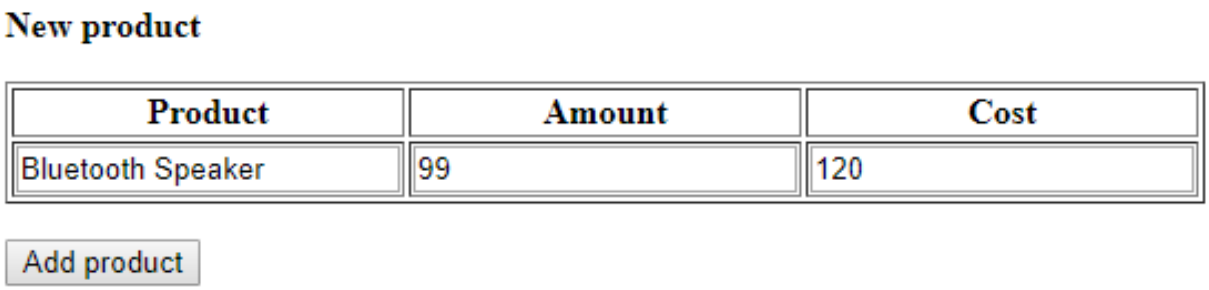

Figure 9.10

9.5. 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.

```php
# process request to store delivered products into the database
if (isset($_POST["save_products"])) {
    $contract_number = $_POST["contract_number"];

    # begin transaction
    mysqli_query($conn, "SET AUTOCOMMIT = 0");
    mysqli_query($conn, "START TRANSACTION");

    $failed = false;

    foreach ($_SESSION["supplied_products"] as $key => $value) {
        $amount = $value["amount"];
        $cost = $value["cost"];

        # keep result of each query inside the transaction
        $result = mysqli_query($conn, "INSERT INTO supplied (contract_number,
            supplied_product, supplied_amount, supplied_cost) VALUES (
            {$contract_number}, '{$key}', {$amount}, {$cost})");

        if (!$result) {
            $failed = true;

            # rollback the transaction if any query is failed
            mysqli_query($conn, "ROLLBACK");
            break;
        }
    }

    if (!$failed) {
        # commit the transaction if there are no failed queries
        mysqli_query($conn, "COMMIT");
    }
```

```php
    # restore autocommit property
    mysqli_query($conn, "SET AUTOCOMMIT = 1");

    # clear session array after products are stored into the database
    $_SESSION["supplied_products"] = NULL;

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

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 ?>
```

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

9.6. 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.