**ლექცია 14. მუშაობა მონაცემთა ბაზებზე**

**DBM- ფუნქციებთან მუშაობა**

PHP- პროგრამებს შეუძლიათ იმუშაონ ისეთ მძლავრ მონაცემთა ბაზებთან, როგორიცაა, მაგალითად, ORACLE, SQL, MySQL და სხვ. თუმცა მონაცემების შენახვა ბაზის ფაილების სახით და მათთან მუშაობა მაშინაც კი არის შესაძლებელი, როცა ზემოთ ჩამოთვლილი ბაზები ჩვენს განკარგულებაში არ იმყოფება. საქმე ისაა, რომ PHP- ს ე.წ. DBM-ფუნქციების მეშვეობით შეუძლია მონაცემთა ბაზის ფუნქციების ემულირება. ცხადია, ასეთი გზით შექმნილ ელემენტარულ ბაზებს ზემოთ მოყვანილი ბაზების ყველა შესაძლებლობა არ გააჩნიათ, მაგრამ რიგ პრაქტიკულ შემთხვევებში მათ წარმატებით შეუძლიათ, გადაწყვიტონ ჩვენ მიერ დასახული ამოცანები.

DBM -მონაცემთა ბაზის გახსნა-დახურვა

ამ მიზანს ემსახურება **dbmopen()** ფუნქცია. მას ესაჭიროება ორი არგუმენტის გადაცემა: DBM-ფაილებამდე გზის და ალმების სტრიქონის. განსახილველი ფუნქციით სარგებლობისას ვქმნით ცვლადს - გაღებული მონაცემთა ბაზის იდენტიფიკატორს, რომლითაც შემდგომ სარგებლობენ სხვა DBM-ფუნქციებიც.

შევნიშნოთ, რომ ამ ფუნქციის გამომყენებულ პროგრამებს უნდა ჰქონდეს იმ კატალოგთან შეღწევაზე ნებართვა, რომელშიც განთავსებულია მონაცემთა ბაზა.

ქვემოთ ჩამოთვლილი ალმები განსაზღვრავენ მონაცემთა ბაზასთან მუშაობის რეჟიმებს:

• r – მონაცემთა ბაზა იღება მხოლოდ წაკითხვისათვის;

• w \_ მონაცემთა ბაზა იღება წაკითხვისა და ჩაწერისთვის;

• c \_ დასაშვებია მონაცემთა ბაზის შექმნა (ანდა მისი წაკითხვა და კორექტირება, თუ იგი უკვე არსებობს);

• n \_ იქმნება ახალი მონაცემთა ბაზა (ნადგურდება ამ სახელის მქონე ბაზის შემცველობა, თუკი იგი უკვე არსებობს).

ქვემოთ მოყვანილ მაგალითში იღება მონაცემთა ბაზა, მაგრამ თუკი ასეთი სახელის მქონე ბაზა არ არსებობს, მაშინ იქმნება ახალი:

$dbh = dbmopen(“./data/products”, “c”) or die/(“ვერ მოხერხდა DBM-ის გაღება”);

ბაზებთან მუშაობას დასამთავრებლად ფუნქციაში მიეთითება გაღებული ბაზის იდენტიფიკატორი. მოცემულ შემთხვევაში მას ექნება სახე:

**dbmclose($dbh);**

**ბაზებში მონაცემების დამატება**

ამ მიზანს ემსახურება dbminsert() ფუნქცია. მოვიყვანოთ მისი გამოყენების მაგალითი:

ლისტინგი 1

<HTML>

<head>

<title> DBM\_ მონაცემთა ბაზაში მონაცემთა დამატება </title>

</head>

<body>

Adding produts now...

<?php

$dbh = dbmopen( "./data/products", "c") or die( "couldn't open DBM" );

dbminsert($dbh, "snic crewdriver", "23.20" );

dbminsert($dbh, "Tricorder", "55.50" );

dbminsert($dbh, "ORAC AI", "2200.500" );

dbminsert($dbh, "HAl 2000", "4500.50" );

dbmclose($dbh);

?>

</body>

</html>

ხაზი გავუსვათ შემდეგ გარემოებას. მონაცემთა ბაზაში ყოველი სახის ინფორმაცია, მათ შორის ციფრულიც, სტრიქონის სახით შეიტანება. ამ მოთხოვნის გამო ზედა მაგალითში ფასის ამსახველი მონაცემიც ბრჭყალებში ჩავსვით.

თუ ამის შემდეგ შეცდომით იგივე სახის მქონე ველის დამატებას შევეცდებით, ამას არავითარი შედეგი არ მოჰყვება:

სისტემა დააბრუნებს არა ოპერაციის წარმატებულად ჩატარების გამომხატველ “0” კოდს

(ან შეცდომის შესახებ მაუწყებელ “-1”-ს), არამედ – “1”-ს.

**ბაზის მონაცემთა კორექტირება**

ბაზაში მონაცემთა კორექტირება ხორციელდება dbmreplece() ფუნქციის გამოყენებით. ზედა პროგრამაში ჩამატების ფუნქცია სწორედ ამ ფუნქციით შევცვალოთ. ცხადია, ვცვლით ველების მნიშვნელობებსაც:

ლისტინგი 2

<HTML>

<head>

<title> DBM\_მონაცემთა ბაზაში მონაცემთა დამატება ან შეცვლა

</title>

</head>

<body>

Adding produts now...

<?php

$dbh = dbmopen( "./data/products", "c") or die( "couldn't open DBM" );

dbmreplace($dbh, "snic screwdriver", "25.20" );

dbmreplace($dbh, "Tricorder", "56.50" );

dbmreplace($dbh, "ORAC AI", "2209.50" );

dbmreplace($dbh, "HAl 2000", "4535.50" );

dbmclose($dbh);

?>

</body>

</html>

**ბაზიდან მონაცემების წაკითხვა**

ბაზიდან მონაცემები **dbmfetch()** ფუნქციის მეშვეობით წაიკითხება. მას არგუმენტად უნდა გადაეცეს გაღებული მონაცემთა ბაზის იდენტიფიკატორი და წასაკითხი ელემენტის სახელი. ფუნქცია მონაცემებს სტრიქონის სახით დაგვიბრუნებს. მაგალითად, **Tricorder** ელემენტის ფასის გასაგებად ვწერთ შემდეგ

ოპერატორს:

$price = dbmfetch ($dbh, “Tricorder”);

თუ ამ სახელის მქონე ელემენტი ბაზაში არ არსებობს, ფუნქცია დააბრუნებს ცარიელ სტრიქონს. მაგრამ როგორ მოვიქცეთ, თუკი ბაზაში შენახული ელემენტების სახელები ჩვენთვის უცნობია, ანდა ზუსტად არ გვახსოვს?

ვთქვათ, გვაინტერესებს ბაზაში შეტანილი ყოველი პროდუქტის ფასი. ასეთ შემთხვევებში დაგვეხმარება **dbmfirstkey()** ფუნქცია, რომელიც გვიბრუნებს ბაზაში პირველი ელემენტის სახელს. შესაძლებელია, ეს ელემენტი პირველად შეტანილი არც გახლდეთ (ასეთია DBM ბაზის შიდა ორგანიზების თავისებურება).

თუმცა ეს საქმეს მაინცდამაინც არ გვირთულებს - **dbmnexthey()** ფუნქციის მეშვეობით ჩვენ შეგვიძლია ბაზის სხვა ელემენტებსაც (აქ მათ გასაღებებს უწოდებენ) მივადგეთ. ელემენტის მნიშვნელობის წაკითხვა კი, როგორც ვიცით, ხორციელდება **dbmfetch()** ფუნქციით.

სამივე ფუნქციისათვის საჭირო გახლავთ მონაცემთა ბაზის იდენტიფიკატორის მითითება. მივიღოთ ინფორმაცია ბაზაში არსებული ყოველი საქონლის შესახებ:

ლისტინგი 3

<HTML>

<head>

<title> DBM\_მონაცემთა ბაზიდან მონაცემთა წაკითხვა </title>

</head>

<body>

Here at the Impossible Gadget Shop we're offering the following exciting products:

<p>

<table border=1 cellpadding ="5">

<tr>

<td aling="center"> <b>product</b></td>

<td aling="center"> <b>price</b></td>

</tr>

<?php

$dbh = dbmopen( "./data/products", "c") or die( "couldn't open DBM" );

$skey = dbmfirstkey($dbh);

while ($skey !="")

{

$value=dbmfetch($dbh,$key);

print "<tr><td align =\"left\">$skey </td>";

print "<td align = \"right\">\$$value </td></tr>";

$key =dbmnextkey($dbh,$key);

}

dbmclose($dbh);

?>

</table>

</body></html>

**ბაზაში ელემენტის არსებობის შემოწმება**

მონაცემების კორექტირების წინ უმჯობესია, შევამოწმოთ ბაზაში მისი არსებობა. ამ მიზნით ვიყენებთ **dbmexists()** ფუნქციას:

if (dbmexists ($dbh, “Tricorder”))

print dbmfetch ($dbh, “Tricorder”))

**მონაცემთა ბაზიდან ელემენტის ამოგდება**

ხორციელდება მარტივად **dbmdelete()** ფუნქციის მეშვეობით.

მაგალითად:

dbmdelete($dbh, “Tricorder”)

**ბაზაში უფრო რთული სტრუქტურების შენახვა**

რადგანაც ბაზაში ინფორმაცია მხოლოდ სტრიქონის სახით ინახება, პრინციპში შესაძლებელია (მაგრამ საკმაოდ რთულდება) მასში მასივებისა და ობიექტების შენახვა. ამ მიზნით იყენებენ სპეციალურ ფუნქციებს.

თუმცა როცა საქმე ასეთი სტრუქტურების გამოყენებამდე მიდის, მაშინ უმჯობესია ვისარგებლოთ უფრო მძლავრი ბაზების შესაძლებლობებით.

**მონაცემთა ბაზებთან კავშირი MySQ-ის მაგალითზე**

პირველი, რაც უნდა გავაკეთოთ, ეს გახლავთ სერვერთან დაკავშირება. PHP-ში ამ მიზანს ემსახურება **mysql-connect()** ფუნქცია. მას გააჩნია სამი არგუმენტი:

1. კომპიუტერის სახელი,

2. მომხმარებლის სახელი,

3. პაროლი.

თუ სერვერთან დაკავშირება მოხერხდა, ფუნქცია გვიბრუნებს მიერთების იდენტიფიკატორს. მას ცვლადის მნიშვნელობად იმახსოვრებენ და ბაზასთან შემდგომი მუშაობისთვის იყენებენ.

ვაჩვენოთ აღნიშნული ფუნქციის გამოყენებით ბაზასთან შეერთების მაგალითი:

$link = mysql\_connect (“locallhost”, “root”, “nickel”);

if ( ! $link )

die (“couldn’t connect to MySQL”);

აღვნიშნოთ, რომ mysql\_connect() ფუნქციაში კომპიუტერის სახელად “locallhost” -ის გამოყენებით კავშირს ვამყარებთ ჩვენსავე კომპიუტერზე განთავსებულ SQL ბაზასთან. Apache Web-სერვერთან მიერთება mysql-pconnect() ფუნქციის მეშვეობითაც არის შესაძლებელი. ასეთ შემთხვევაში სერვერთან შეერთება არ წყდება პროგრამის მუშაობის დამთავრებისთანავე ანდა **mysql\_close()** ფუნქციის გამოძახებით.

**მონაცემთა ბაზის არჩევა**

სერვერთან შეერთების შემდეგ ვირჩევთ მონაცემთა ბაზას **mysql\_select\_db()** ფუნქციის მეშვეობით. ამ ფუნქციას პირველ არგუმენტად გადავცემთ ბაზის სახელს, მეორე \_ არააუცილებულ არგუმენტში მიეთითება სერვერთან მიერთების იდენტიფიკატორი. თუ ეს იდენტიფიკატორი არ ვუჩვენეთ, დუმილით აირჩევა ბოლო მიერთების იდენტიფიკატორი. მაგალითად:

$database = ”sample”;

mysql\_select\_db($sample) or die (“couldn’t open $sample”);

**შეცდომების დამუშავება**

აქამდე ფუნქციის შესრულებისას შეცდომის შემთხვევაში die ფუნქციის მეშვეობით ვწყვეტდით პროგრამის მუშაობას. მაგრამ შესაძლებელია შეცდომის შესახებ აწვრილებითი ინფორმაციის მიღებაც, რაც მის აღმოსაფხვრელად გამოგვადგება. კერძოდ, mysql\_errno()

ფუნქციით განისაზღვრება შეცდომის ნომერი, ხოლო mysql\_error()-ით – სტრიქონი მისი აღწერით.

ვაჩვენოთ mysql\_error()-ფუნქციის გამოყენების მაგალითი:

ლისტინგი 4

<HTML>

<head>

<title>monacemTa bazis gaReba, mierTeba da amorCeva </title>

</head>

<body>

<?php

$user = "harry";

$pass = "elbomonkey";

$db = "sample";

$link = mysql\_connect( "localhost", $user, $pass );

if ( ! $link )

die( "Coudn't connect to MySQL");

print “Successfully connected to server<P>”;

mysql\_select\_db( $db) or die (“Could/t open $db: “.mysql\_error());

print “Successfully selected database \”$db\”<P>”;

mysql\_close( $link);

?>

</body>

</html>

თუ ამ პროგრამაში $db ცვლადს, არარსებული ბაზის სახელს, (ვთქვათ, “sample2””-ს) მივანიჭებთ, die() ფუნქცია გამოგვიყვანს შემდეგ შეტყობინებას:

Couldn’t open sample2: Access denied for user: ‘harry@localhost’ to database

‘sample2’

**ცხრილში მონაცემების ჩამატება**

დავუშვათ, ვქმნით Web-კვანძს, რომელშიც მომხმარებლებს შეუძლიათ თავიანთვის დომენური სახელების ყიდვა. Sample მონაცემთა ბაზაში შევქმნათ ცხრილი domains სახელითა და 4 ველით:

create table domains ( id INT NOT NULL AUTO\_INCREMENT,

PRIMARY EY ( id ),

domain VARCHAR (20),

sex CHAR (1),

mail VARCHAR (20) );

ამ ცხრილში კონკრეტული მონაცემების ჩასამატებლად საჭიროა ბაზას მივმართოთ SQL-მოთხოვნით. ამისათვის PHP-ში გათვალისწინებულია mysql-query() ფუნქცია, რომელსაც გადაეცემა მოთხოვნის სტრიქონი და არააუცილებელი მიერთების იდენტიფიკატორი:

ლისტინგი 5

<html>

<head>

<title> cxrilSi Canaweris damateba

</title>

</head>

<body>

<?php

$user = "harry";

$pass = "elbomonkey";

$db = "sample";

$link = mysql\_connect( "localhost", $user, $pass );

if ( ! $link )

die( "Coudn't connect to MySQL");

mysql\_select\_db( $db, $link)

or die ("Couldn't open $db: ".mysql\_error() );

$query=”INSERT INTO domains ( domain, sex, mail )

values( ‘123xyz.com’, ‘F’, sharp@adomain.com’ )”;

mysql\_query( $query, $link)

or die ("Couldn't add data to \”domains\” table: “.mysql\_error() );

mysql\_close( $link);

?>

</body>

</html>

მივაქციოთ ყურადღება: ბაზაში სტრიქონის ჩამატებისას id-ველისათვის მნიშვნელობა არ გვიჩვენებია. გავიხსენოთ, გასაღებური ველის მნიშვნელობა სისტემის მიერ ავტომატურად განისაზღვრება. ამ პროგრამას რამდენჯერაც შევასრულებთ, ცხრილს იმდენჯერვე დაემატება ერთი და იგივე მონაცემების შემცველი ახალი ჩანაწერი (განსხვავება მხოლოდ id ველის მნიშვნელობაში იქნება).

ქვემოთ მოგვყავს პროგრამა, რომელიც ცხრილში ამატებს მომხმარებლის მიერ ფორმაში შეტანილ მონაცემებს:

ლისტინგი 6

<HTML>

<HEAD>

<TITLE> ბაზაში მომხმარებლის მიერ მოწოდებული ინფორმაციის

დამატება </title>

</head>

<BODY>

<?php

if ( isset( $domain ) && isset( $sex ) && isset( $domain ) )

{

// check user input here!

$dberror = "";

$ret = add\_to\_database( $domain, $sex, $mail, $dberror);

if ( ! $ret )

print "Error: $dberror<BR>";

else

print “Thank you very much”;

}

else {

write\_form();

}

function add\_to\_database( $domain, $sex, $mail, &$dberror)

{

$user = "harry";

$pass = "elbomonkey";

$db = "sample";

$link = mysql\_pconnect( "localhost", $user, $pass );

if ( ! $link )

{

$dberror = "Couldn't connect to MySQL server";

return false;

}

if (! mysql\_seleqt\_db( $db, $kink ) )

{

$dberror =mysql\_error();

return faise;

}

$query = "INSERT INTO domains ( domain, sex, mail )

values( '$domain','$sex','$mail' )";

if ( ! mysql\_query( $query, $link ) )

{

$dberror = mysql\_error();

return false;

}

return true;

}

function write\_form()

{

global$PHP\_SELF;

print"<form action=\'$PHP \_SELF\" method=\"POST\">\n";

print"<input type=\"text\" name=\"domain\"> ";

print"The domain you would like<p>\n";

print"<input TYPE=\"text\" name\"mail\"> ";

print"Your mail address<p>\n" ;

print"<select name=\"sex\">\n";

print"\t<option value=\"F\" Female\n";

print"\t<option value=\"M\"> Male\n";

print"\select>\n";

print"<input type=\"submit\" value=\"submit! \">\n</form>\n";

}

?>

</body>

</html>

თავდაპირველად პროგრამა ამოწმებს **$domain, $mail** და **$sex** ცვლადების არსებობას (ისინი ფორმიდან გადმოიცემა). დადებითი პასუხის შემთხვევაში გამოიძახება **add\_to\_database()** ფუნქცია.

(მანამდე კი განისაზღვრება ჯერჯერობით ცარიელი მნიშვნელობის მქონე **$dberror**

ცვლადი).

თუ აღმოჩნდა, რომ ფორმა ჯერ არ გადმოცემულა (მივაქციოთ ყურადღება - ფორმა გადმოეცემა ამავე პროგრამას) ან მასში თუნდაც ერთი ცვლადის მნიშვნელობა არ იქნება განსაზღვრული, გამოიძახება **write-form()**ფუნქცია. **add\_to\_database()** ფუნქციას 4 არგუმენტი გააჩნია - მომხმარებლის მიერ გადმოცემულ 3 მნიშვნელობას ემატება **$dberror** ცვლადთან დაკავშირებული **$dberror** სტრიქონიც. რომელიმე ოპერაციის უშედეგოდ დასრულებისას ხდება შეცდომის შესახებ ინფორმაციის ჩაწერა ჩვენ მიერ გამოცხადებულ **$dberror** გარე ცვლადშიც (და არა მარტო მის ასლში).

როგორც ვხედავთ, **add\_to\_database()** ფუნქციაში ხორციელდება მცდელობები:

* სერვერთან დაკავშირების,
* ბაზის შერჩევის,
* შQL -მოთხოვნის გაცემის.

თუ რომელიმე მათგანი წარუმატებლად დასრულდა, შეცდომის შესახებ ინფორმაცია გადაეცემა **$dberror** ცვლადს, ხოლო **add\_to\_database**() ფუნქცია PHP-პროგრამას დაუბრუნებს false მნიშვნელობას.

საპირისპირო შემთხვევაში **add\_to\_database**() ფუნქცია პროგრამას უბრუნებს true მნიშვნელობას. ნებისმიერი შემთხვევისთვის პროგრამას გამოჰყავს შესაბამისი შეტყობინება (მოწმდება $ret ცვლადი)

**ავტომატურად კორექტირებადი ველის მნიშვნელობის გაგება**

ამ მიზნით დასაშვებია SQL -მოთხოვნის გამოყენება, მაგრამ თუ გვსურს ინფორმაცია ჩანაწერის შექმნისთანავე მივიღოთ, უნდა ვისარგებლოთ mysql\_insert\_id() ფუნქციით, რომელიც გვიბრუნებს ბოლო ჯერზე ცხრილისთვის დამატებული ჩანაწერის გასაღებური ველის მნიშვნელობას.

მაგალითად, როცა გვსურს მომხმარებელს შევატყობინოთ ნომერი, რომელიც მის შეკვეთას მიენიჭა, ვიყენებთ ასეთ პროგრამულ ფრაგმენტს:

$query=”INSERT INTO domains ( domain, sex, mail ) values (‘$domain’,

‘$sex’, ‘$mail’ )”;

mysql\_query( $query, $link );

$id=mysql\_insert\_id ();

print “Thank you. Your tranzaction number is $id. Please quote it in any queries.”;

**ბაზიდან მონაცემების წაკითხვა**

ამჯერად ვიყენებთ შელეცტ ტიპის SQL-მოთხოვნას. მისი წარმატებით შესრულებისას mysql\_query() ფუნქცია აბრუნებს მოთხოვნის შედეგის იდენტიფიკატორს, რომელიც შეიძლება გადავცეთ მიღებული შედეგის დამამუშავებულ ფუნქციებს.

**SQL-მოთხოვნით მოძებნილი ჩანაწერების რიცხვის განსაზღვრა**

ამ ინფორმაციის მიღება შესაძლებელია mysql\_num\_rows() ფუნქციით. იგი იყენებს ერთადერთ არგუმენტს - მანამდე შესრულებული მოთხოვნის იდენტიფიკატორს. მაგალითად:

ლისტინგი 7

<HTML>

<head>

<title>striqonebis ricxvis gansazRvra</title>

</head>

<body>

<?php

$user = "harry";

$pass = "elbomonkey";

$db = "sample";

$link = mysql\_connect( "localhost", $user, $pass );

if ( ! $link )

die ( "Could't connect to MySQL");

mysql\_select\_db($db, $link)

or die ("Couldn't open $db: ".mysql\_error() );

$result=sql\_query("SELECT \*FROM domains");

$num\_rows=mysql\_num\_rows( $result);

print "There are currently $num\_rows rows in the table<P>";

mysql\_close ($link);

?>

</body>

</html>

**მოთხოვნის შედეგების ჩათვალიერება**

ეს პროცესი ციკლში შეიძლება განვახორციელოთ. მოძებნილი ჩანაწერებისათვის PHP ქმნის ე.წ შინაგან მაჩვენებელს. როცა მიმდინარე ჩანაწერთან ვმუშაობთ, ეს მაჩვენებელი მიგვითითებს მომდევნო ჩანაწერის პოზიციაზე. ამასთან, mysqe-fetch() ფუნქციით შესაძლებელია თითოეული ჩანაწერისათვის მივიღოთ მისი ველებისგან შემდგარი მასივი. ცხადია, ამ ფუნქციასაც უნდა გადაეცეს მოთხოვნის იდენტიფიკატორი. ჩანაწერების ჩათვალიერების ბოლოს ფუნქცია გვიბრუნებს false-ს:

ლისტინგი 8

<HTML>

<head>

<title>cxrilis yvela Canaweris gamoyvana</title>

</head>

<body>

<?php

$user = "harry";

$pass = "elbomonkey";

$db = "sample";

$link = mysql\_connect( "localhost", $user, $pass );

if ( ! $link )

die ( "Could't connect to MySQL");

mysql\_select\_db($db, $link)

or die ("Couldn't open $db: ".mysql\_error() );

$result=sql\_query("SELECT \*FROM domains");

$num\_rows=mysql\_num\_rows( $result);

print "There are currently $num\_rows rows in the table<P>";

print "<table border=1>\n";

while ( $a\_row =mysql\_fetch ($result))

{

print "<tr>\n";

foreach ($a\_row as $field)

print "\t<td>$field</td>\n";

print"</tr>\n";

}

print "</table>\n";

mysql\_close ($link);

?>

</body>

</html>

WHILE-ციკლში mysql\_fetch() ფუნქციის მიერ დაბრუნებულ მნიშვნელობას ვანიჭებთ $a\_row ცვლადს.

WHILE-ციკლში გვაქვს კიდევ ერთი ციკლი foreach. მისი მეშვეობით ხდება მასივის ჩათვალიერება და მისი თითოეული ელემენტის გამოყვანა ცხრილის უჯრედში.

შევნიშნოთ, რომ ჩანაწერის ველებს სახელითაც შეიძლება მივმართოთ. ამასთან, შეიძლება გამოვიყენოთ შემდეგი ორი გზიდან ერთ-ერთი:

* პირველი გულისხმობს WHILE-ის სათაურში mysql\_fetch()ფუნქციის ნაცვლად

mysql\_fetch\_array -ის გამოყენებას (იგი ასოცირებულ მასივს გვიბრუნებს);

* მეორე \_ სტრიქონის აღქმას ობიექტად და მისი თვისებების mysql\_fetch\_object() ფუნქციით წაკითხვას:

print “<TABLE BORDER=1>\n”;

while ( $a\_row = mysql\_fech\_array ($result) )

{

print “<TR>\n”;

print “<TD>$a\_row [mail]</TD><TD>$a\_row [domain]</TD>\n”;

print “</TR>”\n”;

}

print “</TABLE>\n”;

print “<table border=1>\n”;

while ( $a\_row=mysql\_fetch\_object ($result) )

{

print “<tr>\n”;

print “<td> $a\_row -> mail</td><td>$a\_row -> domain </td>\n”;

print “</tr>\n”;

}

print “</table>\n”;

**მონაცემების შეცვლა**

mysql-query() ფუნქციას ვიყენებთ ცხრილში მონაცემების შესაცვლელადაც. მოთხოვნაში ფიგურირებს UPDATE ინსტრუქცია, რომელსაც შეიძლება, ვთქვათ, ასეთი სახე ჰქონდეს:

UPDATE ცხრილის-სახელი SET ველის-სახელი=ახალი-მნიშვნელობა

Where პირობის შესრულება;

UPDATE -ინსტრუქციის წარმატებით შესრულება ჯერ კიდევ არ ნიშნავს მონაცემების ფაქტობრივად შეცვლას, რისი შემოწმებაც შეიძლება მოხდეს mysql\_affected\_rows() ფუნქციით (ეს ფუნქციაც მხოლოდ მიერთების იდენტიფიკატორს საჭიროებს არგუმენტად და აქაც, თუ ამ იდენტიფიკატორს არ მივუთითებთ, გამოყენებული იქნება უკანასკნელი მიერთების იდენტიფიკატორი). შევნიშნოთ, რომ ამ ფუნქციის გამოყენება შეიძლება მონაცემებში ნებისმიერი ცვლილებების შეტანისას. ქვემოთ ნაჩვენებია პროგრამა, რომელიც მონაცემთა ბაზის ადმინისტრატორს საშუალებას აძლევს, ცვლილებები შეიტანოს domains ცხრილის domain ველში:

ლისტინგი 9

<HTML>

<head>

<title>mysql- ფუნქციის გამოყენება ცხრილში მონაცემების შესაცვლელად

</title>

</head>

<body>

<?php

$user = "harry";

$pass = "elbomonkey";

$db = "sample";

$link = mysql\_connect( "localhost", $user, $pass );

if ( ! $link )

die ( "Could't connect to MySQL");

mysql\_select\_db($db, $link)

or die ("Couldn't open $db: ".mysql\_error() );

if ( isset ($domain) && isset ($id))

{

$query="UPDATE domains SET domain = '$domain' where id=$id";

$result = mysql\_query ($query);

if (! $result)

die ("Couldn't update: ".mysql\_error());

print "<h1>Table updated ". mysql\_affected\_rows().

"row(s) hanged</h1><p>";

?>

<form action = "<? print $PHP\_SELF ?>" method = 'POST'>

<select name = "id">

<?

$result=mysql\_query("SELECT domain, id FROM domains");

while ($a\_row = mysql\_fetch\_object ($result))

{

print "<OPTION VALUE = \"$a\_row->id\"";

if ( isset ($id) && $id == $a\_row->id)

print "SELECTED";

print" > $a\_row->domain\n";

}

mysql\_close ($link);

?>

</select>

<input type = "text" name = "domain">

</form>

</body>

</html>

სერვერთან შეერთებისა და მონაცემთა ბაზის არჩევის შემდეგ ვამოწმებთ $domain და $id ცვლადების არსებობას. დადებითი პასუხის შემთხვევაში ვაყალიბებთ SQL-მოთხოვნას. მასში domain ველის მნიშვნელობას ვცვლით იმ ჩანაწერებისთვის, რომლებისთვისაც id ველის მნიშვნელობა ემთხვევა (ფორმაში განსაზღვრულ) $id ცვლადის მნიშვნელობას.

აღვნიშნოთ, რომ შეცდომის შესახებ შეტყობინებას ვერ მივიღებთ, თუ $id -ისთვის შევირჩევთ არარსებულ მნიშვნელობას, ან როცა domain ველის მნიშვნელობა დაემთხვევა ჩვენ მიერ შეთავაზებულს. უბრალოდ, ასეთ შემთხვევებში mysql\_affected\_rows() ფუნქცია დაგვიბრუნებს “0”-ს.

საპირისპირო შემთხვევებში კი (მაგალითად, ზემოთ მოყვანილი პროგრამისათვის) დაგვიბრუნდება მნიშვნელობა “1”, რომელიც ეკრანზეც აისახება.

პროგრამას ეკრანზე გამოჰყავს ფორმა, რომლის დახმარებითაც შეიძლება ცხრილში ცვლილებები შევიტანოთ. Domain და id ველების შესახებ ინფორმაციას გვაწვდის mysql\_query() funqcia ფუნქცია, რომელიც HTML -ტექსტში სიის ფორმირებისათვის გამოიყენება. სიაში ვირჩევთ მონაცემებს, მათი შემდგომი კორექტირების მიზნით. თუ ფორმა უკვე გადავეცით და ამორჩეული id მნიშვნელობა ემთხვევა მიმდინარე ველის მნიშვნელობას, Option ელემენტში შეგვყავს Selected სტრიქონი, რათა ახალი მნიშვნელობა გამოირჩეს ფორმაში.