MySQL

În acest curs vom crea şi integra o bază de date într-un script PHP. Avantajele stocării datelor într-o bază de date, spre deosebire de salvarea datelor în fişiere, includ faptul că sistemele de gestiune a bazelor de date:

- pot furniza un acces mai rapid la date decât fişierele;
- pot fi interogate cu uşurință pentru a extrage mulțimi de date care îndeplinesc anumite criterii;
- dețin mecanisme predefinite pentru tratarea accesului concurent;
- furnizează acces aleator la date;
- furnizează mecanisme predefinite de asigurare a securității.

O bază de date permite obținerea rapidă a unor statistici utile (aflarea clientului care a cheltuit cel mai mult, a produsului celui mai bine vândut).

SGBD-ul pe care îl vom utiliza în acest curs este MySQL. Cursul se bazează pe noțiunile prezentate în cadrul cursurilor de „Baze de date" şi „Sisteme de gestiune a bazelor de date", la care facem trimitere pentru conceptele şi terminologia bazelor de date relaționale, proiectarea şi arhitectura acestora.

Arhitectura unei baze de date *web*

Pornind de la arhitectura internă a unei baze de date, trecem la studiul arhitecturii externe a unui sistem de baze de date *web* şi vom discuta metodologia pentru dezvoltarea unui astfel de sistem de baze de date.

Operația principală a unui server *web* este prezentată în Figura 1. Sistemul constă din 2 obiecte: un *browser web* şi un *server web*, între care este necesară o legătură de comunicație. *Browser*-ul *web* lansează o cerere către server, iar server-ul trimite înapoi un răspuns. Această arhitectură este adecvată unui server care trimite pagini statice. Arhitectura care furnizează un website care lucrează cu o bază de date este mai complexă.

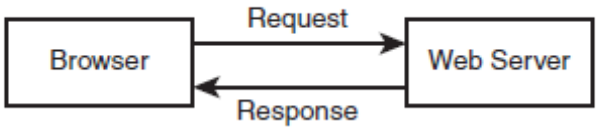

Figura 1

Aplicațiile cu baze de date *web* pe care le vom prezenta urmează o structură generală (Figura 2), care conține un *browser web*, un server *web*, motorul de *scripting* şi server-ul de baze de date.

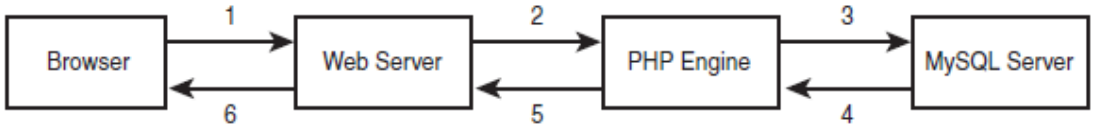

Figura 2

O tranzacție obişnuită într-o bază de date *web* constă din următoarele etape. Pentru o aplicație care lucrează cu o bază de date pentru o bibliotecă, aceste etape pot fi exemplificate astfel:

1. Un utilizator lansează, prin intermediul unui *browser web*, o cerere HTTP pentru o anumită pagină *web*. De exemplu, cu ajutorul unui formular HTML, utilizatorul solicită căutarea tuturor cărților scrise de Liviu Rebreanu. Pagina de căutare a rezultatelor se numeşte *results.php*.
2. Serverul *web* primeşte cererea pentru *results.php*, regăseşte fişierul şi îl transmite motorului PHP pentru procesare.
3. Motorul PHP începe parsarea script-ului. Scriptul conține o comandă pentru conectarea la baza de date şi execută o cerere (căutare de cărți). PHP deschide o conexiune către server-ul MySQL şi trimite cererea corespunzătoare.
4. Serverul MySQL primeşte cererea pentru baza de date, o procesează şi trimite rezultatul (lista de cărți) înapoi motorului PHP.
5. Motorul PHP încheie execuția script-ului, care de obicei presupune şi formatarea rezultatului cererii în HTML. Apoi, returnează rezultatul HTML către server-ul web.
6. Server-ul web trimite codul HTML browser-ului, unde utilizatorul va putea vizualiza lista de cărți solicitată.

Procesul este, în general, acelaşi indiferent de motorul de *scripting* sau de serverul de baze de date folosit. Adeseori, serverul *web*, motorul PHP şi

serverul de baze de date rulează toate pe aceeaşi maşină. Însă, nu este mai puțin obişnuit ca serverul de baze de date să ruleze pe o maşină diferită, din motive de securitate, capacitate crescută, sau distribuire a activităților. Din punct de vedere al dezvoltării, această abordare este similară, însă oferă avantaje din punct de vedere al performanțelor.

Pe măsură ce o aplicație creşte în dimensiune şi complexitate, ea va fi separată în niveluri (*tiers*), de cele mai multe ori un nivel al bazei de date care asigură interfața cu MySQL, un nivel al logicii aplicației care conține nucleul acesteia şi un nivel de prezentare care gestionează ieşirile în format HTML. Arhitectura din figura 2 se păstrează, diferența fiind aceea că secțiunea PHP devine mai structurată.

Crearea bazei de date web în *MySQL*

În cele ce urmează, vom prezenta modul în care se configurează o bază de date MySQL pentru a fi utilizată pe un site *web*. Exemplele vor face referință la un magazin de cărți online, care are o bază de date ale cărei scheme relaționale sunt:

```
Customers(CustomerID, Name, Address, City)
Orders(OrderID, CustomerID, Amount, Date)
Books(ISBN, Author, Title, Price)
Order_Items(OrderID, ISBN, Quantity)
Book_Reviews(ISBN, Reviews
```

Instalarea lui *MySQL* pe server-ul *web* include:

- instalarea fişierelor
- stabilirea unui utilizator care să ruleze MySQL ca administrator
- configurarea căii
- rularea script-ului *mysql_install_db*
- stabilirea parolei *root*
- ştergerea utilizatorului anonim şi testarea bazei de date
- pornirea server-ului MySQL prima dată şi configurarea ca acesta să pornească automat.

Crearea bazei de date şi a utilizatorilor

MySQL poate avea mai multe baze de date. De obicei, vom avea câte o bază de date pentru fiecare aplicație. În exemplul nostru, aceasta se va numi *books*.

Crearea bazei de date se efectuează simplu:

```
mysql> create database <dbname>;
```

Crearea utilizatorilor şi stabilirea privilegiilor

Utilizatorul *root* ar trebui utilizat doar în scopuri administrative, din motive de securitate. Pentru fiecare utilizator care doreşte să folosească sistemul trebuie creat un cont şi o parolă. Nu este necesar ca acestea să fie aceleaşi cu numele de utilizatori şi parolele din afara MySQL (cele din sistemul de operare). Acelaşi principiu se aplică lui *root*.

Atribuirea de parole utilizatorilor nu este obligatorie, dar se recomandă. În scopul configurării unei baze de date *web*, se recomandă stabilirea cel puțin a câte unui utilizator pentru fiecare aplicație web. Motivul pentru care se face această recomandare are legătură cu privilegiile.

Este necesar să avem un utilizator care, prin intermediul *script*-urilor PHP, se conectează la MySQL. Pentru aceasta se aplică principiul privilegiului celui mai mic: se acordă doar privilegiile necesare operațiilor pe care respectivul utilizator va trebui să le efectueze.

Presupunem că utilizatorul va trebui să execute doar comenzi LMD. Prin urmare, stabilirea privilegiilor acestuia se realizează astfel:

```
mysql> grant select, insert, delete, update
-> on books.*
-> to student identified by 'student123';
```

Dacă se utilizează un serviciu de web hosting, de obicei este acordat acces şi la alte privilegii asupra bazei de date pe care serviciul o creează. De obicei, primim acelaşi nume de utilizator şi parolă pentru a fi utilizate în linie de comandă (crearea tabelelor ş.a.m.d) şi pentru conexiunile prin intermediul script-ului (interogarea bazei de date). Utilizarea aceluiaşi nume de utilizator şi a aceleiaşi parole este mai puțin sigură. Putem configura un utilizator cu nivelul corespunzător de privilegii după cum urmează:

```
mysql> grant select, insert, update, delete, index, alter,
create, drop
-> on books.*
-> to student identified by 'student123';
```

Utilizarea bazei de date corespunzătoare

Primul pas după conectare constă în specificarea bazei de date pe care dorim să o utilizăm:

```
mysql> use dbname;
```

Comanda *use* poate fi evitată dacă specificăm baza de date la conectare:

```
mysql -D dbname -h hostname -u username -p
```

Script-ul pentru crearea tabelelor pentru aplicația referitoare la un magazin online de cărți este http://193.226.51.37/web/curs5/books.sql.

Un fişier SQL poate fi rulat cu ajutorul comenzii (trebuie furnizată cale completă către fişierul books.sql):

```
> mysql -h host -u student -D books -p < books.sql
```

Redirecționarea fişierelor presupune că va fi posibil ca acestea să poată fi editate într-un editor de text înainte de a fi executate.

Observații:

- *AUTO_INCREMENT* este o caracteristică specială a MySQL care poate fi utilizată asupra coloanelor întregi. Aceasta presupune că, dacă respectiva valoare nu este furnizată atunci când sunt inserate linii în tabel, MySQL va genera un identificator unic, mai mare cu 1 decât valorile existente în coloană. Putem avea o singură astfel de coloană în fiecare tabel.
- *UNSIGNED* specificat după un tip întreg presupune că acesta poate avea doar valori mai mari sau egale decât 0.
- Tipul de date *char* specifică şiruri de caractere de dimensiune fixă. Tipul de date *varchar* utilizează doar spațiul necesar, deci furnizează o optimizare în termeni de spațiu, însă tipul de date *char* este mai rapid.
- Tipul de date *int* reține numere întregi, iar *float* stochează numere reale în virgulă mobilă.
- Pentru datele calendaristice se utilizează tipul *date*.
- Tipul de date *TINYINT UNSIGNED* reține valori întregi între 0 şi 255.
- Tipul de date *text* permite stocarea textelor de dimensiuni mari.

Accesarea bazei de date *MySQL* utilizând *PHP*

Vom relua cei 6 paşi corespunzători arhitecturii unei baze de date *web*.

Mai întâi, se creează formularul de căutare HTML: http://193.226.51.37/web/curs5/search_html.txt.
Acesta produce http://193.226.51.37/web/curs5/search.html

Script-ul care va fi apelat atunci când este acționat butonul Search este http://193.226.51.37/web/curs5/results_php.txt

Rezultatul în urma unei căutări cu ajutorul script-ului se poate observa introducând în căutare „*Java 2 for Professional Developers*".

În continuare, vom explica ce face acest script şi modul în care lucrează el.

Interogarea unei baze de date de pe *web*

Într-un script utilizat pentru a accesa o bază de date de pe *web*, au loc următoarele acțiuni:

1. Verificarea şi filtrarea datelor care au fost introduse de către utilizator.
2. Stabilirea unei conexiuni la baza de date corespunzătoare.
3. Interogarea bazei de date.
4. Recuperarea rezultatelor.
5. Prezentarea rezultatelor către utilizator.

Aceştia sunt paşii care au fost urmăriți în fişierul *results.php*.

Verificarea şi filtrarea datelor de intrare

Script-ul începe prin a elimina orice spațiu liber pe care utilizatorul l-ar fi putut introduce din greşeală la începutul sau sfârşitul termenului său de căutare:

```
$searchterm=trim($_POST['searchterm']);
```

Următorul pas este de a căuta dacă utilizatorul a introdus un termen de căutare şi a selectat un tip de căutare:

```
if (!$searchtype || !$searchterm) {
      echo "You have not entered search details. Please go back
      and try again.";
      exit;
}
```

Filtrarea intrărilor din partea utilizatorului include filtrarea caracterelor de control. În acest sens, sunt utile funcțiile *addslashes*(), *stripslashes*() şi *get_magic_quotes_gpc*(). În MySQL trebuie introduse secvențe *escape* asupra datelor atunci când acestea provin din partea utilizatorului.

În acest caz, se utilizează valoarea returnată de funcția *get_magic_quotes_gpc*(), care verifică dacă introducerea apostrofurilor se realizează automat. În caz contrar, se utilizează funcția *addslash*() pentru a introduce secvențele *escape* corespunzătoare:

```
if (!get_magic_quotes_gpc()) {
```

```
        $searchtype = addslashes($searchtype);
        $searchterm = addslashes($searchterm);
}
```

Funcția *stripslashes*() se utilizează asupra datelor care se întorc din baza de date. Dacă este activată opțiunea *magic quotes*, datele vor conține caractere *slash* atunci când se întorc din baza de date, deci trebuie eliminate.

În exemplul nostru, utilizăm *htmlspecialchars*()pentru a codifica acele caractere care au semnificații speciale în HTML. Datele curente de test nu includ caractere &, <, > sau „, însă unele titluri de cărți pot conține &. Prin utilizarea acestei funcții, se elimină posibilele erori.

Stabilirea unei conexiuni

Biblioteca PHP pentru conectarea la MySQL se numeşte *mysqli* (*i* = improved). Atunci când se utilizează această bibliotecă, folosim fie o sintaxă procedurală, fie una orientată pe obiecte.

Pentru a realiza o conexiune la server-ul MySQL, vom avea următoarea linie de cod în script:

```
@ $db = new mysqli('localhost', 'student', 'student123',
'books');
```

Această linie de cod instanțiază clasa *mysqli* şi creează o conexiune la *localhost* cu numele de utilizator şi parola specificate. Conexiunea este configurată pentru a utiliza baza de date *books*.

Utilizând această abordare orientată pe obiecte, putem invoca metode asupra acestui obiect pentru a accesa baza de date. Dacă este preferată abordarea procedurală, linia de cod corespunzătoare este următoarea:

```
@ $db = mysqli_connect('localhost', 'bookorama', 'bookorama123',
'books');
```

Această funcție returnează o resursă, care reprezintă conexiunea la baza de date. Resursa va fi apoi transferată tuturor celorlalte funcții *mysqli*, similar modului în care lucram cu fişiere.

Majoritatea funcțiilor *mysqli* au o interfață orientată-obiect şi una procedurală. În general, numele funcțiilor din versiunea procedurală încep cu *mysql_* şi necesită ca argument resursa obținută cu ajutorul funcției *mysqli_connect*(). Conexiunile la baza de date constituie o excepție de la această regulă deoarece ele se pot realiza de către constructorul obiectului *mysqli*.

Rezultatul încercării de conectare trebuie verificat, deoarece nimic nu va funcționa dacă nu s-a obținut o conexiune validă. Verificarea se poate realiza astfel, atât în versiunea procedurală cât şi în cea orientată obiect:

```
if (mysqli_connect_errno()) {
      echo 'Error: Could not connect to database. Please try
            again later.';
      exit;
}
```

Funcția *mysql_connect_errno*() returnează 0 dacă nu au apărut erori la conectare, altfel returnează numărul erorii.

Remarcăm că linia de cod care realizează conectarea la baza de date începe cu @ pentru suprimarea erorilor. În acest mod, putem trata erorile corespunzător. Acelaşi lucru se poate realiza şi cu ajutorul excepţiilor.

Există o limită a conexiunilor MySQL care pot fi active la un moment dat. Parametrul MySQL *max_conections* determină această limită. Scopul acestui parametru şi al lui *MaxClients* din *Apache* este acela de a determina server-ul să respingă noile conexiuni, prevenind ca resursele să fie utilizate complet în perioade aglomerate.

Ambele valori ale acestor parametri pot fi modificate în fişierele de configurare: *httpd.conf* (pentru *MaxClients*) şi *my.conf* (pentru *max_connections*).

Alegerea bazei de date

În linia de comandă specificăm baza de date care va fi utilizată astfel:

```
use books;
```

Acest lucru este necesar şi atunci când conexiunea se realizează pe *web*. Baza de date va fi specificată ca parametru al constructorului *mysqli* sau al funcției *mysqli_connect*(). Dacă dorim să schimbăm baza de date, putem utiliza funcția *mysqli_select_db*(), care poate fi accesată:

```
$db->select_db(dbname)
```

sau

```
mysqli_select_db(db_resource, db_name)
```

Interogarea bazei de date

Pentru a efectua o interogare se utilizează funcția mysql_query(). Înainte de aceasta, se stabileşte cererea care va fi executată.

```
$query = "select * from books where ".$searchtype." like
'%".$searchterm."%'";
```

În acest caz este căutată valoarea introdusă de către utilizator ($searchterm) în câmpul specificat de către utilizator ($searchtype).

Cererea care este trimisă către MySQL nu trebuie să conţină „;”, spre deosebire de cele lansate direct în MySQL.

Cererea poate fi executată astfel:

```
$result = $db->query($query);
```

sau

```
$result = mysqli_query($db, $query);
```

Versiunea OO returnează un obiect, iar cea procedurală o resursă. În cazul unei erori, funcţia returnează *false*.

Regăsirea rezultatelor cererii

Există funcţii care permit obţinerea de informaţii din obiectul sau resursa rezultată.

În exemplul nostru, sunt numărate liniile returnate de cerere şi, de asemenea, este utilizată funcţia *mysqli_fetch_assoc*().

Atunci când este utillizată abordarea OO, numărul de linii returnate este stocat în atributul *num_rows* al obiectului rezultat şi poate fi accesat astfel:

```
$num_results = $result->num_rows;
```

În varianta procedurală, funcţia *mysqli_num*() returnează această informaţie pentru resursa furnizată ca parametru:

```
$num_results = mysqli_num_rows($result);
```

Această valoare ne permite să procesăm sau să afişăm rezultatul cererii, utilizând o instrucţiune de ciclare:

```
for ($i=0; $i <$num_results; $i++) {
      // process results
}
```

Instrucţiunea de ciclare nu se va executa dacă nu a fost returnată nicio linie. În fiecare iteraţie a acestei instrucţiuni de ciclare , va trebui apelat `$result -> fetch_assoc()` (sau `mysqli_fetch_assoc()`). Această funcţie ia fiecare linie din mulţimea rezultat şi o returnează ca vector, în care fiecare atribut este cheie şi îi corespunde valoarea de pe linia respectivă:

```
$row = $result->fetch_assoc();
```

sau

```
$row = mysqli_fetch_assoc($result);
```

Dat acest vector, putem parcurge şi afişa fiecare câmp:

```
echo "<br />ISBN: ";
```

```
echo stripslashes($row['isbn']);
```

Pot fi utilizate câteva variante pentru a obține informații din rezultat. În locul unui vector cu chei, rezultatele pot fi regăsite într-un vector cu enumerare utilizând *mysql_fetch_row*():

```
$row = $result->fetch_row($result);
```

sau

```
$row = mysqli_fetch_row($result);
```

Valorile atributelor se află în fiecare dintre valorile tabloului *$row*[0], *$row*[1] etc.

De asemenea, o înregistrare poate fi regăsită în cadrul unui obiect cu ajutorul funcției *mysql_fetch_object*():

```
$row = $result->fetch_object();
```

sau

```
$row = mysqli_fetch_object($result);
```

Putem accesa fiecare atribut prin *$row -> title*, *$row -> author* etc.

Deconectarea de la o bază de date

Putem elibera mulțimea rezultat utilizând:

```
$result->free();
```

sau

```
mysqli_free_result($result);
```

Apoi, pentru închiderea bazei de date scriem:

```
$db->close();
```

sau

```
mysqli_close($db);
```

Utilizarea acestei funcții nu este obligatorie deoarece script-ul va închide conexiunea la terminarea execuției.

Introducerea de informații în baza de date

Inserarea de informații noi în baza de date este asemănătoare modului în care acestea sunt obținute din baza de date. Sunt urmați aceiaşi paşi: crearea unei conexiuni, trimiterea unei cereri şi verificarea rezultatelor.

Cererea care este trimisă corespunde, de data aceasta, unei comenzi INSERT.

Considerăm formularul
http://193.226.51.37/web/curs5/newbook.html.
Codul acestuia este
http://193.226.51.37/web/curs5/newbook_html.txt.

Rezultatele acestui formular sunt transmise script-ului *insert_book.php*, care preia informațiile, realizează câteva validări şi încearcă să le scrie în baza de date.

Acest script este: http://193.226.51.37/web/curs5/insert_book_php.txt
Adăugăm corect o carte şi observăm ce se obține.

Observăm că script-ul *insert_book.php* este similar celui care regăsea date din baza de date. Verificăm dacă au fost completate toate câmpurile formularului şi le formatăm corect pentru inserarea în baza de date cu *addslashes*():

```
if (!get_magic_quotes_gpc()) {
      $isbn = addslashes($isbn);
      $author = addslashes($author);
      $title = addslashes($title);
      $price = doubleval($price);
}
```

Deoarece prețul este stocat în baza de date ca număr real, nu trebuie să îi adăugăm caractere „\”. Putem obține acelaşi efect de filtrare a caracterelor necorespunzătoare asupra unui câmp numeric utilizând funcția *doubleval*() (de exemplu, aceasta va trata simbolurile monetare care ar fi putut fi adăugate din greşeală).

Conexiunea la baza de date se realizează instanțiind obiectul *mysqli* şi va fi creată o cerere care va fi trimisă la baza de date şi executată:

```
$query = "insert into books values
('".$isbn."', '".$author."', '".$title."', '".$price."')";
$result = $db->query($query);
```

O diferență între utilizarea lui SELECT şi a lui INSERT este dată de folosirea funcției *mysqli_affected_rows*() sau a metodei corespunzătoare:

```
echo $db->affected_rows." book inserted into database.";
```

Această funcție se utilizează în cazul instrucțiunilor de modificare a bazei de date; în cazul unei cereri SELECT, funcția corespunzătoare este *mysqli_num_rows*().

Utilizarea instrucțiunilor „pregătite”

Biblioteca *mysqli* permite utilizarea cererilor pregătite. Acestea permit o execuție mai rapidă atunci când este executată de mai multe ori aceeaşi cerere cu date diferite. De asemenea, aceste cereri constituie un mijloc de prevenire a atacurilor *SQL Injection*.

Conceptul de bază al unei cereri pregătite este acela că trimitem un *template* al cererii pe care vom dori să o executăm server-ului MySQL, iar apoi trimitem datele separat. Putem trimite mai multe date unei cereri, funcționalitate utilă în cadrul inserărilor de tip *bulk*.

În cadrul script-ului *insert_books.php* putem utiliza instrucțiuni pregătite astfel:

```
$query = "insert into books values(?, ?, ?, ?)";
$stmt = $db->prepare($query);
$stmt->bind_param("sssd", $isbn, $author, $title, $price);
$stmt->execute();
echo $stmt->affected_rows.' book inserted into database.';
$stmt->close();
```

La crearea cererii, în loc de a furniza valori ale atributelor se utilizează caracterul „?” pentru fiecare dată. Linia de cod *$db -> prepare* (*mysqli_stmt_prepare*()) construieşte un obiect sau resursă instrucțiune, care va fi utilizată pentru a realiza procesarea efectivă.

Obiectul instrucțiune are o metodă numită *bind_param*() (*mysql_stmt_bind_param*()). Scopul acestei metode este de a comunica lui PHP variabilele care ar trebui înlocuite caracterelor „?”. Primul parametru este un şir de caractere care specifică formatul, similar funcției *printf*. Valoarea „sssd” specifică faptul că avem 4 parametri, tipurile de date ale primilor 3 fiind *string* iar al celuilalt *double*. Alte caractere posibile în acest şir care specifică formattul sunt : *i* (numere întregi), *b* (*blob*).

Apelul la *$stmt -> execute* (*mysqli_stmt_execute*()) efectuează cererea. Ulterior, poate fi accesat numărul de linii afectate şi se poate încheia instrucțiunea.

De ce este utilă o astfel de instrucțiune pregătită? Putem modifica valorile celor 4 variabile legate şi reexecuta instrucțiunea fără a o mai pregăti. Această funcționalitate este utilă la ciclarea în cazul inserărilor *bulk*.

Similar legării parametrilor, se pot lega rezultatele. Pentru cererile de tip SELECT se poate utiliza *$stmt -> bind_result()* (sau *mysqli_stmt_bind_result*()) pentru a furniza lista variabilelor în care recuperăm coloanele din rezultat. La fiecare apel al *$stmt -> fetch*() (sau

mysqli_stmt_fetch()), valorile coloanelor din următoarea linie din mulțimea rezultat sunt atribuite acestor variabile legate. De exemplu, în script-ul anterior avem:

```
$stmt->bind_result($isbn, $author, $title, $price);
```

Această instrucțiune leagă cele 4 variabile de cele 4 coloane care vor fi returnate de către cerere.

După apelul

```
$stmt->execute();
```

se poate apela în instrucțiunea de ciclare:

```
$stmt->fetch();
```

La fiecare apel al acesteia , următoarea linie rezultat va fi regăsită în cele 4 variabile legate.

Putem utiliza funcțiile *mysqli_stmt_bind_param*() şi *mysqli_stmt_bind_result*() în cadrul aceluiaşi script.