

Universitatea "Lucian Blaga" din Sibiu
Facultatea de Științe Economice

Sisteme Informatice de Asistare a Deciziilor
Aplicații în PHP-MySQL

Îndrumar de laborator

Pentru Învățământ la Distanță

2022-2023

Introducere



Obiectivele laboratorului

Laboratorul urmărește familiarizarea studenților cu termenii specifici limbajelor de programare web, însușirea noțiunilor de bază necesare în gestiunea bazelor de date distribuite pe web, precum și crearea și dezvoltarea abilităților practice de a utiliza aceste tehnologii web, utile în proiectarea și dezvoltarea aplicațiilor economice.



Competențe conferite

După parcurgerea materialului studenții vor fi capabili:

- să lucreze cu un server WEB și să implementeze arhitectura client-server;
- să creeze un șablon static pentru o pagină web, pe care să îl populeze dinamic din PHP;
- să folosească variabile, constante și structuri de control PHP;
- să prelucereze formularele trimise prin metodele GET sau POST;
- să lucreze cu fișiere pe server;
- să creeze baze de date distribuite;
- să integreze o bază de date MySQL într-o aplicație PHP;
- să manipuleze datele dintr-o tabelă, folosind principalele operații create, read, update, delete;
- să valideze corect datele trimise dintr-o pagină WEB.



Resurse și mijloace de lucru

– *mijloace informatice necesare parcurgerii laboratorului și rezolvării problemelor propuse: limbajul de scripting PHP și sistemul de gestiune a bazelor de date MySQL;*



Cerințe preliminare

Discipline necesare a fi parcurse și eventual promovate înaintea disciplinei curente:

- Informatică
- Baze de date
- Tehnologii Web pentru Realizarea Aplicațiilor Financiar-Contabile



Durata medie de studiu individual – 20-25 ore



Evaluarea

Componenta notei finale:

- ponderea evaluării finale(examen oral în fața calculatorului) – 40%;
- verificarea temelor de curs&laborator – 10%;
- verificările de laborator(proiect) – 50%;



STRUCTURA LABORATOR:

- I. **Unitatea de învățare 1 - Modul de funcționare al tehnologiei client-server și instalarea soft-urilor necesare**
- II. **Unitatea de învățare 2 - Baze de date MySQL.**
Comenzi utilizate în lucrul cu baze de date MySQL.
- III. **Unitatea de învățare 3 - Limbajul PHP -**
variabile, funcții, structuri de control
- IV. **Unitatea de învățare 4 - Funcții PHP de conectare la baza de date MySQL**
- V. **Unitatea de învățare 5 - Aplicație practică de tipul unui site dinamic**

Unitatea de învățare 1



Modul de funcționare al tehnologiei client-server și configurarea unui server web

I.1. Funcționarea rețelelor de calculatoare

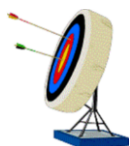
- **Comunicarea client – server**
- **Protocolul HTTP**
- **Avantajele modelului client – server**

I.2. Tehnologia Web – Server Apache / MySQL / PHP

- **Instalarea server web**
- **Utilizarea server web**

Competențele unității de învățare

După parcurgerea acestui modul studenții sunt capabili:



- să identifice tipul și arhitectura rețelelor de calculatoare precum și principalele browsere web;
- să identifice tehnologiile web ce se pot utiliza în realizarea sistemelor informatice web destinate evidenței financiar-contabilă.



Durata medie de parcurgere a acestei unități de învățare este de 4-5 ore

I.1. Rețele de calculatoare

Calculatoarele mici au un raport preț/calitate mult mai bun decât cele mari. Sistemele mari de calcul (calculatoare de mărimea unei camere) sunt cam de zece ori mai rapide decât calculatoarele personale, dar costă de o mie de ori mai mult. Acest dezechilibru i-a determinat pe mulți proiectanți să construiască sisteme distribuite formate din calculatoare personale, câte unul pentru fiecare utilizator, datele din rețea fiind păstrate pe unul sau mai multe **servele de fișiere** partajate. În acest model utilizatorii sunt numiți **clienți**, iar întregul aranjament poartă numele de **model client-server**. Acest model este ilustrat în **Figura 1**

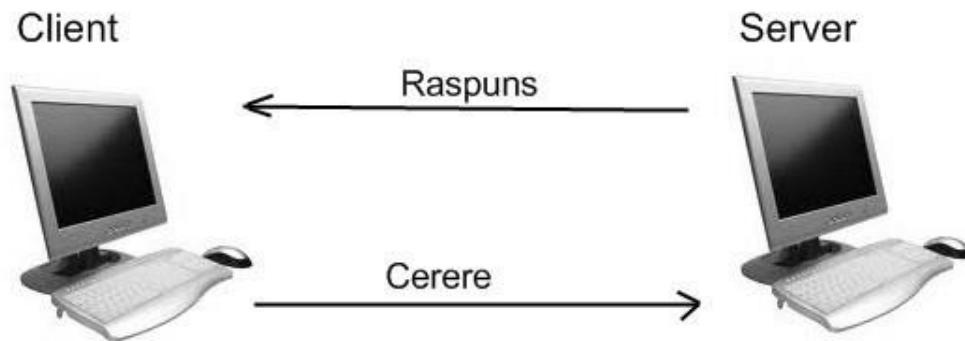




Figura 1 – Modelul Client - Server

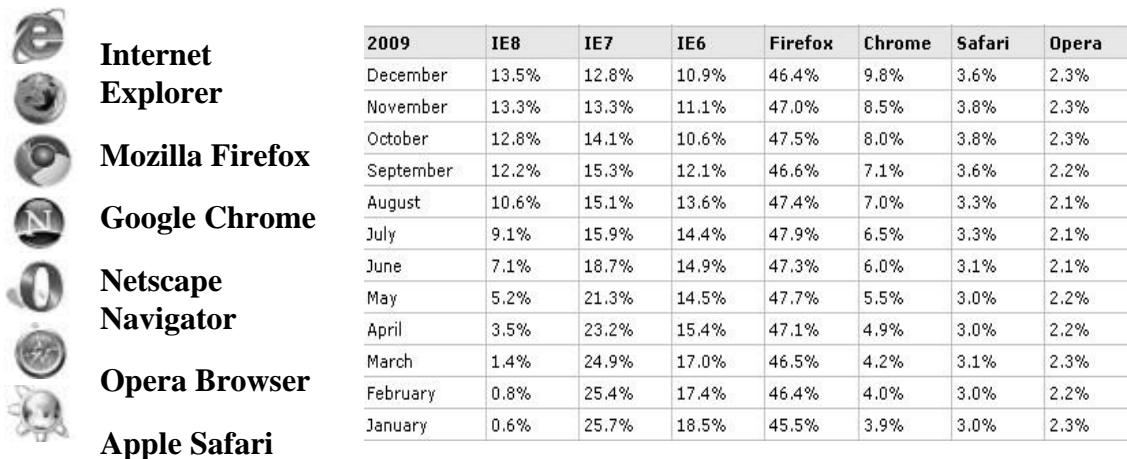
Clientul este partea din aceasta arhitectura care va initia comunicarea. Este cel care va trimite o cerere catre un server pentru a primi un raspuns, niste informatii personalizate pentru cererea care tocmai a facut-o, date in general. Clientul poate fi un browser web

(Internet Explorer , Mozilla Firefox ) care se conecteaza la un server web, poate fi un client de e-mail (Thunderbird, Microsoft Outlook) care se conecteaza la un server de email, trimite datele de autentificare pentru un cont de e-mail si cererea de primire a mesajelor noi, poate fi un client de FTP (TotalCommander, Cute FTP) care va trimite unui server de FTP cererea de stocare pe acel server a unui fisier, urmat de fisierul in sine si va primi ca raspuns confirmarea primirii acestuia sau un mesaj de eroare corespunzator. Asadar clientul este cel care va actiona si va determina un intreg lant de actiuni din partea server-ului: initiaza cererea catre server, asteapta raspunsul de la server, primeste raspunsul de la server si in final il returneaza utilizatorului posibil intr-un mod formatat.

◦ Browsere web

Un browser web este o aplicatie [software](#) ce permite utilizatorilor sa afiseze text, grafica, video, muzica si alte informatii localizate pe o pagina din [World Wide Web](#), dar si sa comunice cu ofertantul de informatii, respectiv cu serverul web pe care se afla stocata informatia.

Unele dintre cele mai utilizate browsere web sunt descrise in **Figura 2**, precum si procentajul de utilizatori pe care il are fiecare browser. Acest procent devine important in momentul in care creem o aplicatie web, deoarece aceasta trebuie sa functioneze corect pe toate browserele majore:



Konqueror (UNIX)

Sursa: www.w3schools.com

Figura 2 – Principalele browsere web si cota lor de piata

Serverul are ca principala activitate a sta si astepta. Serverul nu va actiona niciodata pe cont propriu, nu va transmite date decat daca este intrebat si decat daca sunt urmate anumite reguli de comunicare. Cand este pornit, un server va lua pozitia de asteptare de conexiuni (numita tehnic: listening state), de regula acesta asculta pe un anume port primirea conexiunilor. La primirea unei astfel de conexiuni, deci implicit a unei cereri, el va face toate demersurile necesare pentru a returna rezultatul asteptat. Daca este un server web, va intoarce clientului (browserul web) codul html al paginii care a fost ceruta, daca este un server de e-mail va returna clientului o lista cu toate email-urile pe care le-a primit de la ultima cerere, daca este un server de MySQL va prelua interogarea SQL primita o va executa si va returna setul de date rezultat. Aşadar un server stă şi aşteaptă conexiuni pe care le va servi cererile de îndată ce au fost primite.

Regulile şi convenţiile utilizate în conversaţia dintre server si client sunt cunoscute sub numele de **protocol**. In principal, un protocol reprezintă o înţelegere între părţile care comunica, asupra modului de realizare a comunicarii. Cel mai des folosit protocol pentru comunicarea pe internet este **HTTP** (Hypertext Transfer Protocol). Protocolul HTTP este un protocol de tip text, fiind protocolul "implicit" al WWW (World Wide Web). Adica, daca un **URL** (Uniform Resource Locator) nu contine partea de protocol, aceasta se considera ca fiind http. HTTP presupune ca pe calculatorul destinatie ruleaza un program

care intelege protocolul. Fisierul trimis la destinatie poate fi un document **HTML** (abreviatie de la **HyperText Markup Language**), un fisier grafic, de sunet, animatie sau video, de asemenea un program executabil pe server-ul respectiv sau si un editor de text.

HTTP ofera o tehnica de comunicare prin care paginile web se pot transmite de la un computer aflat la distanta spre propriul computer. Daca se apeleaza un link sau o adresa de web cum ar fi **http://www.example.com**, atunci se cere calculatorului host sa afiseze o pagina web (de exemplu **index.php**). In prima faza numele (adresa) **www.example.com** este convertit de protocolul DNS intr-o adresa IP. Urmeaza transferul prin protocolul TCP pe portul standard 80 al serverului HTTP, ca raspuns la cererea HTTP-GET. Informatii suplimentare ca de exemplu indicatii pentru browser, limba dorita s.a.m.d. se pot adauga in header-ul (antetul) pachetului HTTP. In urma cererii HTTP-GET urmeaza din partea serverului raspunsul cu datele cerute, ca de ex.: pagini in (x)HTML, cu fisiere atasate ca imagini, fisiere de stil (CSS), scripturi (Javascript), dar pot fi si pagini generate dinamic (SSI, JSP, PHP si ASP.NET). Daca dintr-un anumit motiv informatiile nu pot fi transmise, atunci serverul trimite inapoi un mesaj de eroare. Modul exact de desfasurare a acestei actiuni (cerere si raspuns) este stabilit in specificatiile HTTP.

Header	Mesaj	CRC
Ip Browser	Continut util	Biti de verificare

Deseori utilizatorul doreste sa transmita informatii speciale la website. Aici HTTP pune la dispozitie doua posibilitati:

- Transferul datelor in combinatie cu o cerere pentru o resursa (HTTP-metoda "GET")
- Transferul datelor in combinatie cu o cerere speciala (HTTP-metoda "POST")

\$_GET	\$_POST
Numele si valoarea variabilelor se vede in	Numele si valoarea variabilelor nu se vede
URL	in URL
Permite bookmarkuirea paginii	Nu permite bookmarkuirea paginii
Valoarea dintr-o variabila nu poate depasi 100 de caractere	Valoarea dintr-o variabila nu are limita de caractere

Datele transferate vin deseori codate (**urlencode**). La metoda GET se utilizeaza partea de cerere Uniform Resource Identifiers (URI) cu simbolul ?. Aceasta metoda se utilizeaza pentru a transfera o lista de parametri, pe care partea opusa trebuie sa o ia in considerare la prelucrarea cererii.

Observatie. Protocolul HTTP combinat cu protocolul SSL (Secure Socket Layer) / TLS (Transport Layer Security) permite criptarea informatiei si identificarea sigura a

serverului formând un alt protocol denumit HTTPS (Hypertext Transfer Protocol Secure) care este folosit la tranzacții financiare pe Internet.

📖 Pentru informații detaliate despre acest subiect se recomandă citirea cărții scrise de

Andrew S. Tanenbaum – ‘Rețele de Calculatoare’



Să ne reamintim...

⌘ Avantajele modelului client – server

- ✓ accesul la informație de la distanță;
- ✓ costuri scăzute;
- ✓ fiabilitate maximă;
- ✓ scalabilitate.

I.2. Tehnologia Web – Server Apache / MySQL / PHP

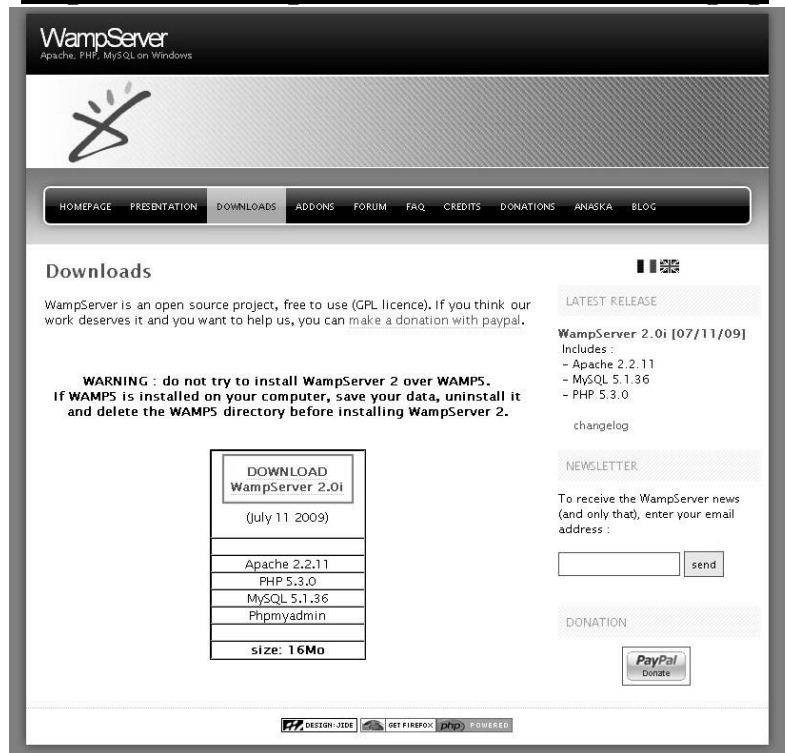
Există mai multe tipuri de tehnologii web, dintre cele mai utilizate sunt: PHP/MySQL, ASP/ MSSQL, Ruby etc.

Pentru crearea unui server pe calculatorul personal se pot folosi programe speciale ce permit instalarea si configurarea unui server web cu doar cateva clickuri. Un exemplu de astfel de pachet este WampServer (pentru Windows) sau LampServer (pentru Linux) ce permite crearea unui server pe calculatorul personal, server denumit **localhost**.

☺ Exemplul 1 – instalarea si utilizarea WampServer

Pasul 1. Kitul WampServer2.0i.exe se poate downloada gratuit de la adresa urmatoare

<http://www.wampserver.com/en/download.php>



Pasul 2. Dati dublu click pe fisierul downloadat **WampServer2.0i.exe** si urmati instructiunile de pe ecran. Totul este automat. Pachetul WampServer este livrat cu cele mai noi versiuni de Apache, MySQL si PHP.

Pasul 3. Porniti serverul Apache dand click dreapta pe icon-ul de langa ceas si apoi selectati **Start All Services**



Pasul 4. Daca se doreste configurarea serverului se face din fisierul **httpd.conf** (initial serverul vine cu proprietatile default care sunt suficiente si nu necesita modificare) iar apoi dupa ce a fost modificat trebuie restartat serverul folosind meniul de la **Pasul 3.**

! ATENTIE. Modificarea fisierului de configurare necesita cunostinte de administrare de retea.

Exemplu de cod din fisierul de configurare httpd.conf

```
#
# ServerAdmin: Your address, where problems with the server should be
# e-mailed. This address appears on some server-generated pages, such
# as error documents. e.g. admin@your-domain.com
#
ServerAdmin webmaster@localhost

#
# ServerName gives the name and port that the server uses to identify itself. #
This can often be determined automatically, but we recommend you specify #
it explicitly to prevent problems during startup.
#
# If your host doesn't have a registered DNS name, enter its IP address here.
#
ServerName localhost:80

#
# DocumentRoot: The directory out of which you will serve your #
documents. By default, all requests are taken from this directory, but #
symbolic links and aliases may be used to point to other locations.
```

```
#
DocumentRoot "C:/wamp/www"

#
# Each directory to which Apache has access can be configured with respect
# to which services and features are allowed and/or disabled in that #
# directory (and its subdirectories).
#
# First, we configure the "default" to be a very restrictive set of #
# features.
#
<Directory />
    Options FollowSymLinks
    AllowOverride None
    Order deny,allow
    Deny from all
    Satisfy all
</Directory>
```

Pasul 5. După ce setup-ul va termina de instalat aplicația un director “**www**” este creat (de obicei c:\wamp\www). În acest director se pot crea subdirectoare pentru proiectele dumneavoastră și puteți pune fișierele PHP în el. Apoi pentru a vedea acest director puteți da click pe linkul “**Localhost**” din WampServer sau puteți deschide un browser și accesa adresa **http://localhost** ca în **Figura 3**

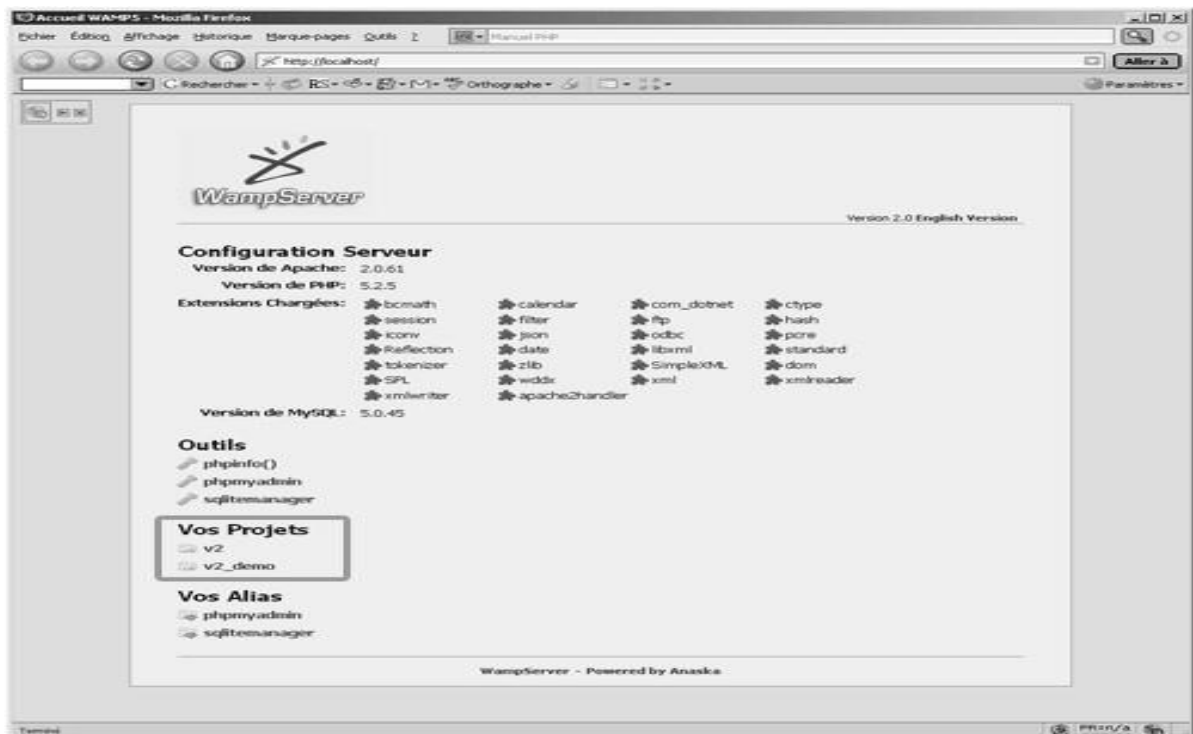


Figura 3 – Accesarea fișierelor PHP de pe serverul local (localhost)



Să ne reamintim...

¶ În cadrul laboratorului vom folosi un server Apache / MySQL / PHP, printre principalele avantaje ale acestor tehnologii fiind:

- fiabilitate ridicată;
- tehnologia este ”*free*”;
- updatare periodică a versiunilor;
- comunitate dedicată;
- multe locuri de muncă/proiecte care necesită cunoașterea acestor tehnologii.

Unitatea de învățare 2



Baze de date MySQL

II.1. Prezentarea sistemului de gestiune a bazelor de date MySQL

- Prezentarea conceptului de baza de date
- Ce este MySQL?

II.2. Folosirea modului de administrare phpMyAdmin

II.3. Comenzi utilizate in lucrul cu baze de date MySQL

- Query – uri
SELECT, INSERT, DELETE, UPDATE
- Cheie Primara si Cheie Străină
- Query – uri avansate
JOIN, SUBQUERY, UNION
- Cuvinte cheie și funcții MySQL **DISTINCT, LIKE, SUBSTRING, NOW**

II.4. Protejarea la SQL Injection

II.1. Prezentarea sistemului de gestiune a bazelor de date MySQL

O baza de date (DB) reprezinta o modalitate de stocare a unor informatii si date pe un suport extern (un dispozitiv de stocare), cu posibilitatea regasirii rapide a acestora. De obicei o baza de date este memorata intr-unul sau mai multe fisiere (exemplu fisierele de tip text cu extensia **.dat**). Bazele de date sunt manipulate cu ajutorul sistemelor de gestiune a bazelor de date. Cel mai raspandit tip de baze de date este cel relational, in care datele sunt memorate in tabele. Pe langa tabele, o baza de date relationala mai poate contine: indecsi, proceduri stocate, declansatori, utilizatori si grupuri de utilizatori, tipuri de date, mecanisme de securitate si de gestiune a tranzactiilor etc. Alte tipuri de baze de date sunt modelul ierarhic, modelul orientat pe obiecte si, mai nou, modelul XML (eXtensible Markup Language).

☺ Exemplul 2 – Structura unei baze de date

Intr-o baza de date relationala exista tabele in care sunt memorate datele. Fiecare tabela are unul sau mai multe campuri in care se pot pastra valori (vezi **Figura 4**)

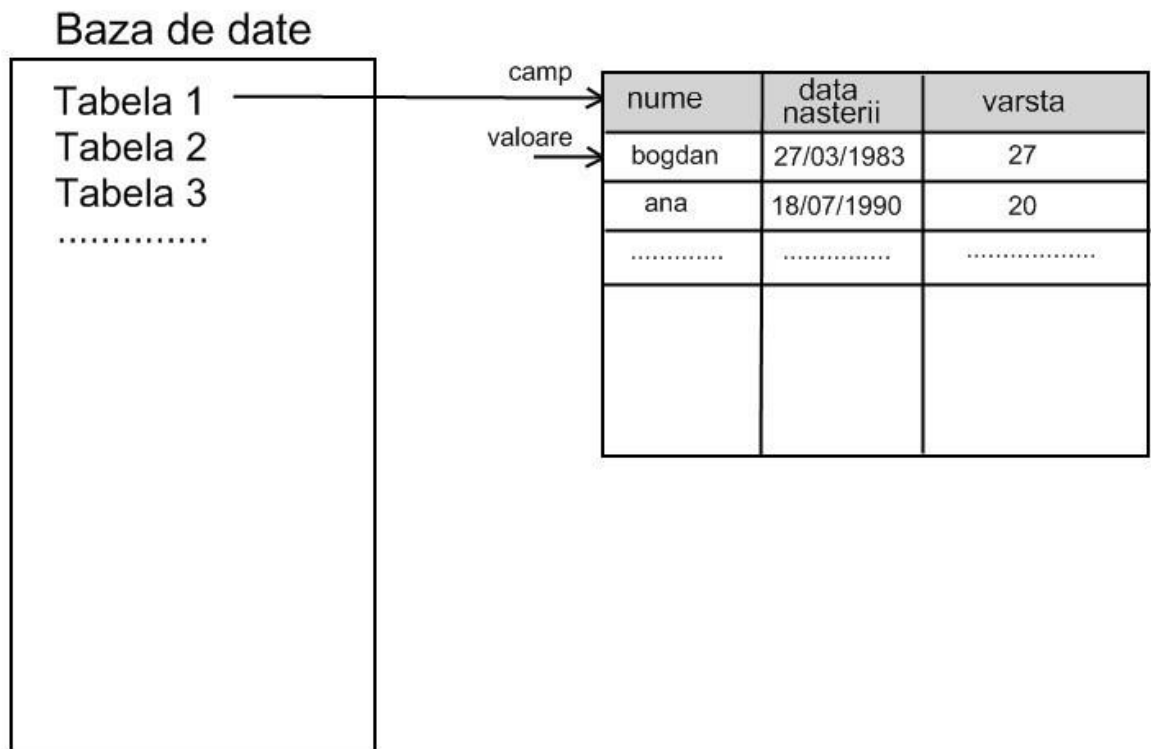


Figura 4 – Structura unei baze de date

In **Figura 4** campurile sunt reprezentate pe coloane (nume, data nasterii, varsta), iar valorile pe rand (bogdan, 27/03/1983, ana, s.a.m.d.).

Trebuie creat un camp (coloana) individual pentru fiecare tip de date pe care vrem sa-l stocam (nume, data, varsta etc.). Principalele **tipuri de campuri** ale unei tabele sunt urmatoarele:

INT – contine numere de tip intreg (ex. 27, 35)

FLOAT – contine numere reale (ex 1234.32, 23416.04)

VARCHAR – sir de caractere (ex. Bogdan, Str. Stefan Cel Mare)

DATE – variabila de tip data (ex. 27/03/1983,01-01-2010)

DATETIME – variabila de tip data completa (ex: 27-06-1967 14:10:00)

TIMESTAMP – variabila de tip data in format calculator (ex: 20050506133000)

Pe de alta parte un rand contine valorile specificate pentru aceste campuri. Fiecare rand va avea o valoare (chiar daca este NULL – nici o valoare) pentru fiecare coloana, in exemplul nostru valorile sunt : bogdan, ana, 27/03/1983, 27 etc.

Pe langa tipul coloanei mai putem specifica **dimensiunea** acesteia (numarul de caractere pe care acesta il poate accepta) de la 1 la 2^{32} in bazele de date de tip MyISAM. Tipul MyISAM este cel mai folosit tip alaturi de ISAM, HEAP, innoDB, Berkley DB. Daca incercam sa introducem o valoare mai mare decat numarul de caractere specificat atunci valoarea va fi trunchiata la numarul de caractere specificat.

Exemplu:

Daca doresc sa introduc valoarea '*bogdan*' intr-un camp prenume de tip VARCHAR(4) atunci in baza va aparea doar *bogd*, restul informatiei pierzandu-se.

Mai putem specifica tipul de caractere pe care campul il primeste, asa numitul **Collation**. Default valoarea este *latin1_swedish_ci*, si contine caractere latine. O alta valoare foarte folosita este *utf8_swedish_ci* care este folosita pentru caractere speciale de tipul literelor chirilice (Rusia), caractere speciale din Cehia, Polonia, Ungaria etc.

DEFAULT specifica valoarea pe care o va lua campul in mod automat (valoarea poate fi NULL, 0 sau orice alta valoare dorim).



Să ne reamintim...

⌘ Avantajele folosirii unei baze de date

- Bazele de date sunt folosite cand dorim stocarea de informatie in categorii logice, de exemplu vrem sa stocam informatia despre angajatii unei companii. Cu ajutorul unei baze de date poti imparti compania in diferite departamente(tabele) pentru a ajuta la pastrarea logica a informatiei. Exemple de tabele ar putea fi: Angajati, Supervizori, Clienti etc. Fiecare tabela ar contine la randul ei coloane specifice, de exemplu la tabela Angajati am putea avea urmatoarele campuri: Nume, Data, Pozitie, Varsta, Salariu etc.
- De asemenea accesul la bazele de date este foarte usor si rapid, iar o cantitate mare de informatie poate fi stocata ocupand un spatiu relativ mic pe server.

◦ Ce este MySQL?

SQL (Structured Query Language) permite accesul si manipularea bazelor de date. Un limbaj SQL poate executa interogari (query-uri) asupra bazelor de date, poate returna valori din baza de date, poate insera valori in baza de date, updata campuri din tabele s.a.m.d.



MySQL este un sistem de gestiune a bazelor de date relational, produs de compania suedeza MySQL AB si distribuit sub Licenta Publica Generala GNU. Este cel mai popular SGBD open-source la ora actuala. Desi este folosit foarte des impreuna cu limbajul de programare PHP, cu MySQL se pot construi aplicatii in orice limbaj major. Exista multe scheme API disponibile pentru MySQL ce permit scrierea aplicatiilor in numeroase limbaje de programare pentru accesarea bazelor de date MySQL, cum are fi: C, C++, C#, Java, Perl, PHP, Python, FreeBasic, etc. Licenta GNU GPL nu permite incorporarea MySQL in softuri comerciale; cei care doresc sa faca acest lucru pot achizitiona, contra cost, o licenta comerciala de la compania producatoare, MySQL AB.

Pentru a administra bazele de date MySQL se poate folosi modul linie de comanda sau, prin descarcare de pe internet, o interfata grafica: MySQL Administrator si MySQL Query Browser. Un alt instrument de management al acestor baze de date este aplicatia gratuita, scrisa in PHP, **phpMyAdmin**.

MySQL poate fi rulat pe multe dintre platformele software existente: AIX, FreeBSD, GNU/Linux, Mac OS X, NetBSD, Solaris, SunOS, Windows 9x/NT/2000/XP/Vista.



Pentru informatii detaliate despre acest subiect se recomanda accesarea site-ului oficial MySQL <http://www.mysql.com/> unde puteţi găsi o descriere detaliată a acestui sistem de gestiune a bazelor de date precum şi sintaxa corectă a interogărilor şi o mulţime de exemple cu explicaţiile de rigoare.

II.2. Folosirea modului de administrare phpMyAdmin

Pe localhost daca accesam din browser adresa <http://localhost/> si apoi dam click pe linkul [phpmyadmin](http://localhost/phpmyadmin) sau direct <http://localhost/phpmyadmin> vom putea rula aplicatia phpMyAdmin ce permite executarea de interogari asupra bazei de date.

La logare vom fi intrebati de datele de conectare la serverul de MySQL (user / parola).



Figura 5 – Logare la phpMyAdmin

Daca le inseram corect vom fi redirectati la pagina urmatoare de unde putem selecta **Databases** si va aparea lista cu toate bazele de date existente pe serverul de MySQL iar din lista care apare (vezi **Figura 6**) selectam baza de date dorita sau putem crea una noua scriind numele bazei de date in campul **Create new database** si apoi apasand butonul **Create**.

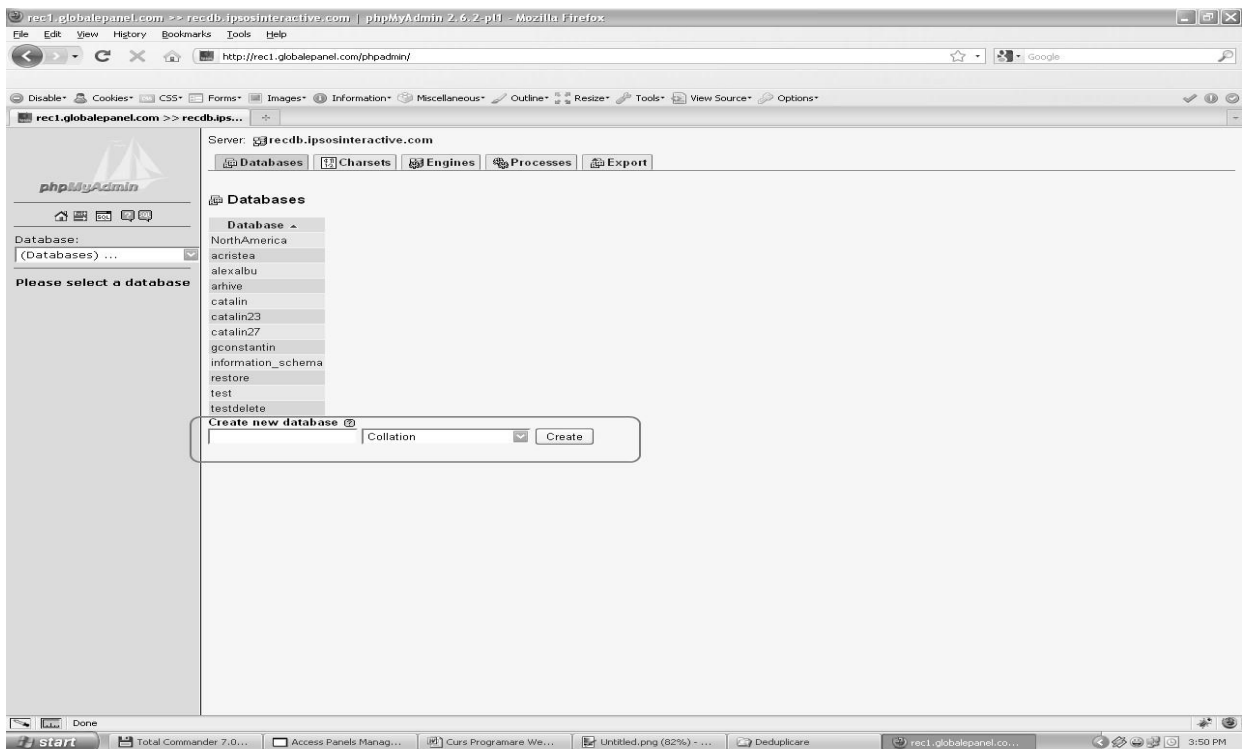


Figura 6 – Selectare/Creare baza de date in phpMyAdmin

Si vom fi redirectati catre lista cu tabelele din baza de date curenta (vezi **Figura 7**), tab-ul **Structure** din aplicatie

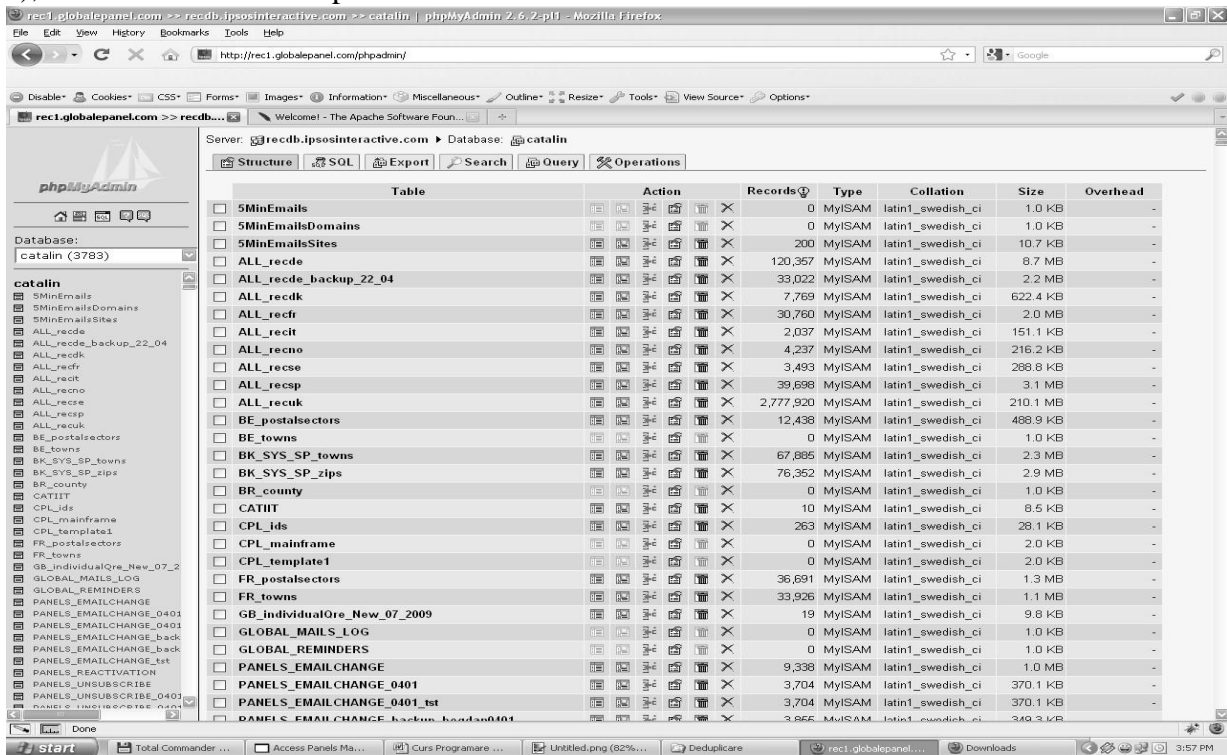


Figura 7 – phpMyAdmin lista tabele

Celelate taburi din phpMyAdmin sunt:

SQL – unde putem scrie query-uri SQL

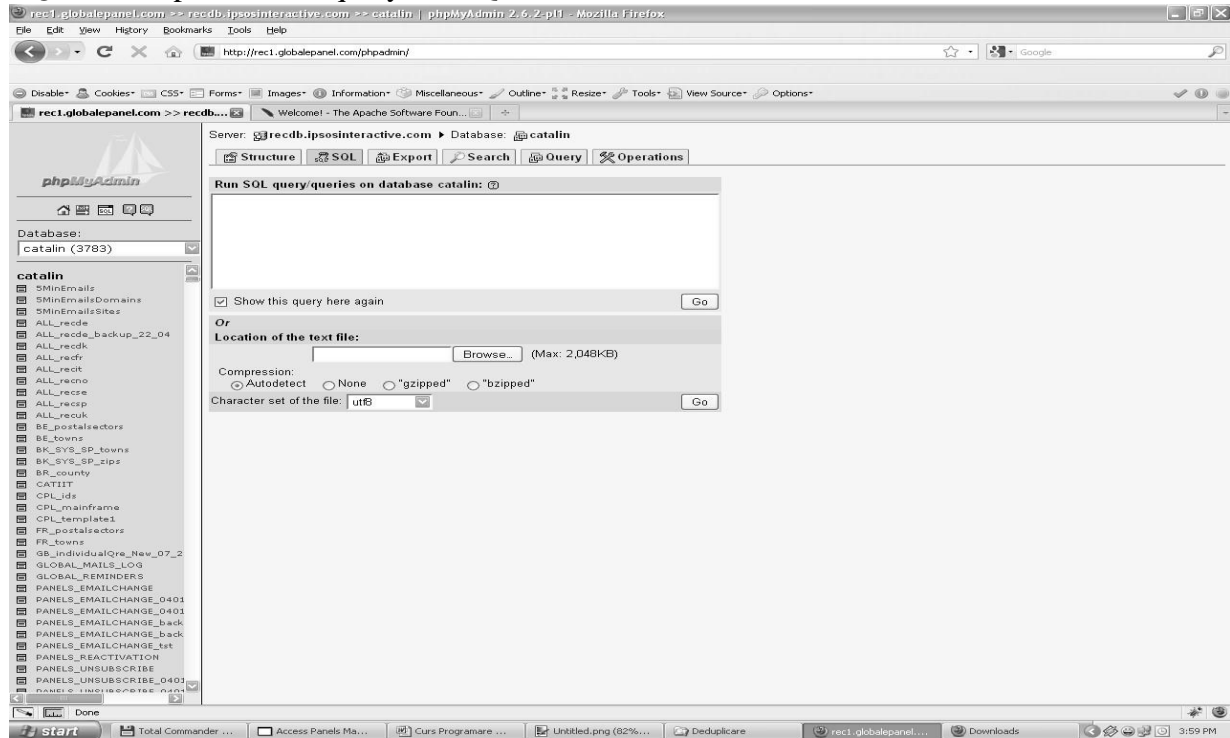


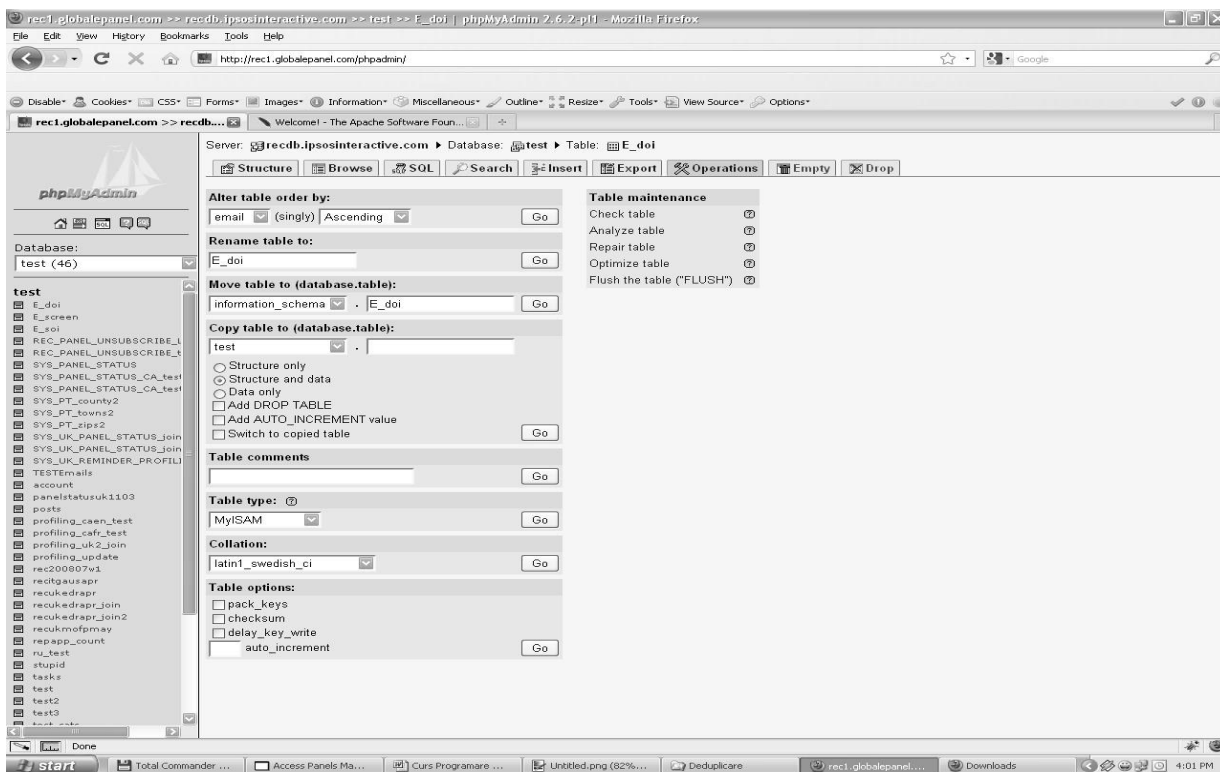
Figura 8 – phpMyAdmin tab SQL

Export – unde putem exporta informatia dintr-o tabela in fisiere cu diferite formate (Excel .xls, CSV .csv, text .txt etc).

Search – care permite cautarea informatiei din baza de date folosind o interfata

Query – asemanator tab SQL cu o alta interfata

Operations - de unde putem copia o tabela, redenumi o tabela existenta, verifica statusul tablei etc.



Pentru mai multe detalii despre aceasta aplicație ”free” puteți vizita pagina oficială la adresa de mai jos:

http://www.phpmyadmin.net/home_page/index.php

II.3. Comenzi utilizate în lucrul cu baze de date MySQL

1) Creare baza de date - CREATE DATABASE `nume_baza_de_date`

O baza de date in MySQL este implementata ca un director ce contine fisiere ce corespund tabelor din baza de date. O baza de date poate avea orice nume (mai mic de 64 de caractere) , cu conditia ca daca numele este un numar trebuie comentat (ex. `719`). De asemenea anumite succesiuni de caractere trebuiesc evitate 0x00, 1e, Me etc.

2) Stergere baza de date – DROP [IF EXISTS] DATABASE `nume_baza_de_date`

Sterge o baza de date si toate tabelele din aceasta.



ATENȚIE: Ștergerea unei baze de date va duce la pierderea informației din toate tabelele din acea bază de date.

3) Creare tabelă - CREATE TABLE `nume_tabela`

Regulile de formare a denumirii tabelelor sunt asemanatoare cu cele ale bazelor de date. In query-ul de creare a tabelii trebuie specificata si structura acesteia cu campurile necesare si tipul acestora. De asemenea se poate stabili si dimensiunea maxima pe care o pot lua valorile din acel camp. Daca o valoare va depasi aceasta dimensiune atunci valoarea va fi truncheata la dimensiunea maxima admisa.

☺ Exemplul 3 – Structura unei tabele

```
CREATE TABLE `angajati` (  
  `Id` int(11) default NULL,  
  `nume` varchar(255) default NULL,  
  `data_nasterii` date default NULL,  
  `pozitie` varchar(255) default NULL,  
  `departament` varchar(255) default NULL,  
  `salariu` int(11) default NULL  
)
```

OBSERVAȚIE: Daca nu se specifica tipul tabelii, atunci aceasta este de tipul default (in MySQL este MyISAM). Putem specifica insa si tipul tabelii, in functie de acesta tabela are diverse proprietati.

//tabela de tip MyISAM

```
CREATE TABLE tblMyISAM (  
  id INT NOT NULL AUTO_INCREMENT,  
  PRIMARY KEY (id),  
  value_a TINYINT  
) TYPE=MyISAM
```

//tabela de tip HEAP

```
CREATE TABLE tblHeap (
    id INT NOT NULL AUTO_INCREMENT,
    PRIMARY KEY (id),
    value_a TINYINT
) TYPE=Heap

//schimbarea tipului tabelii din MyISAM in InnoDB

ALTER TABLE tblMyISAM CHANGE TYPE=InnoDB
```

4) Stergere tabela – DROP TABLE `nume_tabela`

Sterge o tabela si toate informatiile din ea.



ATENȚIE: Ștergerea unei tabele va duce la pierderea tuturor informațiilor din tabela respectivă.

5) Golire tabela – TRUNCATE TABLE `nume_tabela`

Golește tabela ștergând toată informația din aceasta dar păstrând tabela.

6) Inserarea informatiei intr-o tabela - INSERT INTO `nume_tabela` (camp1, camp2, ...) VALUES (val1, val2, ..)

Comanda INSERT INTO este folosita pentru inserarea datelor intr-o tabela. De fiecare data cand rulam aceasta comanda vom adauga un nou rand in tabela

Exemplu:

```
INSERT INTO `angajati` VALUES (1, 'Bogdan', '1983-03-27', 'Programator', 'Soft', 5000);
```

```
INSERT INTO `angajati` VALUES (2, 'Ana', '1987-12-12', 'Secretara', 'Marketing', 2500);
```


INSERT INTO `angajati` VALUES (3, 'George', '1977-09-14', 'Director', 'Management', 8500);

INSERT INTO `angajati` VALUES (6, 'Cami', '1983-06-06', 'Vanzator', 'Sales', 5000);

Rezultatul acestui query va fi urmatorul

Id	Nume	Data Nasterii	Pozitie	Departament	Salariu
1	Bogdan	27/03/1983	Programator	Soft	5000
2	Ana	12/12/1987	Secretara	Marketing	2500
3	George	14/09/1977	Director	Management	8500
4	Sandra	15/12/1987	Programator	Soft	4000
5	Mircea	18/05/1956	Vanzator	Sales	3000
6	Cami	06/06/1983	Vanzator	Sales	5000
...

7) Selectare informatie din tabela -

Comanda SELECT este folosita pentru selectarea datelor dintr-o tabela. Rezultatul este stocat intr-o tabela rezultat, denumita set de rezultate.

☺ **Exemplul 4** – Explicație funcționare comanda SELECT

SELECT [nume_coloana1],[nume_coloana2] **FROM** `nume_tabela` **WHERE** [conditie1] **AND/OR** [conditie2] **GROUP BY** [nume_coloana] **HAVING** [conditie4] **ORDER BY** [nume_coloana] **ASC/DESC**

Exemplu:

Fie tabela ANGAJATI cu urmatoarea structura

Id	Nume	Data Nasterii	Pozitie	Departament	Salariu
1	Bogdan	27/03/1983	Programator	Soft	5000
2	Ana	12/12/1987	Secretara	Marketing	2500
3	George	14/09/1977	Director	Management	8500
4	Sandra	15/12/1987	Programator	Soft	4000
5	Mircea	18/05/1956	Vanzator	Sales	3000
...

Explicație parametri SELECT:

• [nume_coloana] – coloana pe care dorim să o selectăm (putem selecta oricâte coloane dorim). Dacă dorim să selectam toate coloanele putem folosi simbolul * (all).

Dacă dorim să selectăm Numele și Salariul fiecărui individ din companie

SELECT Nume, Salariu FROM angajati

Va returna următorul rezultat:

Showing rows 0 - 4 (5 total, Query took 0.0005 sec)

SQL query:
SELECT nume, salariu
FROM angajati_RB
LIMIT 0, 30

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

	nume	salariu
<input type="checkbox"/>	Bogdan	5000
<input type="checkbox"/>	Ana	2500
<input type="checkbox"/>	George	8500
<input type="checkbox"/>	Sandra	4000
<input type="checkbox"/>	Mircea	3000

Check All / Uncheck All With selected:

Dacă dorim toată informația din tabela angajati

SELECT * FROM angajati

Va returna întreaga informație

Showing rows 0 - 4 (5 total, Query took 0.0008 sec)

SQL query:
 SELECT *
 FROM angajati_RB
 LIMIT 0, 30

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
 in horizontal mode and repeat headers after 100 cells

			Id	nume	data_nasterii	pozitie	departament	salariu
<input type="checkbox"/>			1	Bogdan	1983-03-27	Programator	Soft	5000
<input type="checkbox"/>			2	Ana	1987-12-12	Secretara	Marketing	2500
<input type="checkbox"/>			3	George	1977-09-14	Director	Management	8500
<input type="checkbox"/>			4	Sandra	1987-12-15	Programator	Soft	4000
<input type="checkbox"/>			5	Mircea	1956-05-18	Vanzator	Sales	3000

Check All / Uncheck All With selected:

• **WHERE** – reprezintă condiția pentru a selecta acea coloană

De exemplu dacă vrem să aflăm numele și salariul tuturor persoanelor cu salariul mai mare de 3500 RON.

SELECT Nume, Salariu FROM angajati WHERE Salariu>3500







Acest query va returna rezultatul de mai jos




Showing rows 0 - 2 (3 total, Query took 0.0005 sec)

SQL query:
 SELECT Nume, Salariu
 FROM angajati_RB
 WHERE Salariu >3500
 LIMIT 0 , 30

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
 in horizontal mode and repeat headers after 100 cells

	Nume	Salariu
<input type="checkbox"/>  	Bogdan	5000
<input type="checkbox"/>  	George	8500
<input type="checkbox"/>  	Sandra	4000

Check All / Uncheck All With selected:   

De asemenea la condiția WHERE putem avea condiții compuse folosind operatorii logici AND (SI), OR (SAU) și ! (NOT).

Valoare 1	Operator	Valoare 2	Rezultat
ADEVARAT (1)	AND	ADEVARAT (1)	ADEVARAT (1)
ADEVARAT (1)	AND	FALS (0)	FALS (0)
ADEVARAT (1)	OR	ADEVARAT (1)	ADEVARAT (1)
ADEVARAT (1)	OR	FALS (0)	FALS (0)
	!	ADEVARAT (1)	FALS (0)
	!	FALS (0)	ADEVARAT (1)

Daca dorim sa aflam numele si salariul tuturor persoanelor cu salariul mai mare de 3500 RON dar mai mic de 6000 RON.

SELECT Nume, Salariu FROM angajati WHERE Salariu>3500 AND Salariu<6000

Acest query va returna rezultatul următor:

Showing rows 0 - 1 (2 total, Query took 0.0006 sec)

SQL query:
 SELECT Nume, Salariu
 FROM angajati_RB
 WHERE Salariu >3500
 AND Salariu <6000
 LIMIT 0, 30

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
 in horizontal mode and repeat headers after 100 cells

	Nume	Salariu
<input type="checkbox"/>	Bogdan	5000
<input type="checkbox"/>	Sandra	4000

Check All / Uncheck All With selected:

• **GROUP BY** – este folosit când dorim să împărțim în grupuri informația din tabela

Dacă dorim să aflăm cât se cheltuiește cu salariile angajaților, pe fiecare departament în parte

SELECT Departament, SUM(Salariu) FROM angajati GROUP BY Departament

Acest query va returna următorul rezultat:

Showing rows 0 - 3 (4 total, Query took 0.0005 sec)

SQL query:
 SELECT Departament, SUM(Salariu)
 FROM angajati_RB
 GROUP BY Departament

[Edit] [Create PHP Code]

Show : 30 row(s) starting from record # 0
 in horizontal mode and repeat headers after 100 cells

Departament	SUM(Salariu)
Management	8500
Marketing	2500
Sales	3000
Soft	9000



ATENȚIE: Cu GROUP BY se pot folosi și funcțiile agregate de tipul SUM(), MIN(), MAX(), AVG(), COUNT()

- **SUM()** – calculează suma câmpurilor din grup

SELECT Departament, SUM(Salariu) FROM angajati GROUP BY Departament

- **MIN()** – alege minimul valorilor dintre toate valorile din grup

SELECT Departament, MIN(Salariu) FROM angajati GROUP BY Departament

Showing rows 0 - 3 (4 total, Query took 0.0006 sec)

SQL query:


```
SELECT Departament, MIN( Salariu )
FROM angajati
GROUP BY Departament
LIMIT 0 , 30
```

Show: 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

Departament	MIN(Salariu)
Management	8500
Marketing	2500
Sales	3000
Soft	4000

- **MAX()** – alege maximul valorilor dintre toate valorile din grup

SELECT Departament, MAX(Salariu) FROM angajati GROUP BY Departament

 Showing rows 0 - 3 (4 total, Query took 0.0008 sec)

SQL query:


```
SELECT Departament, MAX( Salariu )  
FROM angajati  
GROUP BY Departament  
LIMIT 0 , 30
```

Show: row(s) starting from record #
in mode and repeat headers after cells

Department	MAX(Salariu)
Management	8500
Marketing	2500
Sales	5000
Soft	5000

• **AVG()** – returnează media valorilor din grup

SELECT Departament, AVG(Salariu) FROM angajati GROUP BY Departament

 Showing rows 0 - 3 (4 total, Query took 0.0028 sec)

SQL query:

```
SELECT Departament, AVG( Salariu )
FROM angajati
GROUP BY Departament
LIMIT 0 , 30
```


Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

← T →

Departament	AVG(Salariu)
Management	8500.0000
Marketing	2500.0000
Sales	4000.0000
Soft	4500.0000

• **COUNT()** – returnează numărul de valori din grup

SELECT Departament, COUNT(*) FROM angajati GROUP BY Departament

 Showing rows 0 - 3 (4 total, Query took 0.0009 sec)

SQL query:

```
SELECT Departament, COUNT( * )
FROM angajati
GROUP BY Departament
LIMIT 0 , 30
```

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

← T →

Departament	COUNT(*)
Management	1
Marketing	1
Sales	2
Soft	2

• **HAVING** – este folosit când dorim să punem condiție pe informația returnată de **GROUP BY**

Dacă dorim să aflăm suma salariilor angajaților pe fiecare departament, dar doar pentru departamentele ce cheltuiesc pe salariile angajaților mai mult de 8000 RON pe luna

SELECT Departament, SUM(Salariu) as suma FROM angajati GROUP BY Departament HAVING suma>8000

Acest query va returna urmatorul rezultat:

Showing rows 0 - 1 (2 total, Query took 0.0005 sec)

SQL query:
SELECT Departament, SUM(Salariu) AS suma
FROM angajati_RB
GROUP BY Departament
HAVING suma >8000
LIMIT 0 , 30

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

Departament	suma
Management	8500
Soft	9000

• **ORDER BY** – este folosit când dorim să sortăm rezultatul query-ului ascendent **ASC** sau descendent **DESC**

Dacă dorim să aflăm suma salariilor angajaților, pe fiecare departament, sortat descrescător în funcție de sumă

SELECT Departament, SUM(Salariu) as suma FROM angajati GROUP BY Departament ORDER BY suma DESC

Acest query va returna rezultatul:

Showing rows 0 - 3 (4 total, Query took 0.0006 sec)

SQL query:
 SELECT Departament, SUM(Salariu) AS suma
 FROM angajati_RB
 GROUP BY Departament
 ORDER BY suma DESC
 LIMIT 0, 30

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
 in horizontal mode and repeat headers after 100 cells

Departament	suma
Soft	9000
Management	8500
Sales	3000
Marketing	2500

8) Updatarea informatiei - UPDATE `nume_tabela` SET [camp]='valoare_noua'

Comanda UPDATE este folosita pentru modificarea datelor dintr-o tabela. De fiecare data cand rulam aceasta comanda vom modifica informatia din tabela

De exemplu daca angajatului Cami i se creste salariul la 6000 de RON

UPDATE `angajati` set salariu="6000" WHERE nume="Cami"

Dupa acest query in baza vom avea

Id	Nume	Data Nasterii	Pozitie	Departament	Salariu
1	Bogdan	27/03/1983	Programator	Soft	5000
2	Ana	12/12/1987	Secretara	Marketing	2500
3	George	14/09/1977	Director	Management	8500
4	Sandra	15/12/1987	Programator	Soft	4000
5	Mircea	18/05/1956	Vanzator	Sales	3000
6	Cami	06/06/1983	Vanzator	Sales	6000
...



ATENȚIE: Dacă nu se specifică o condiție asupra căror înregistrări se va face update-ul toate înregistrările din acea tabelă, de la câmpul respectiv, vor fi updateate. De exemplu query-ul

UPDATE `angajati` set salariu="6000"

Va avea ca rezultat updatarea tuturor salariilor la 6000 RON.

9) Stergerea informatiei din tabela - DELETE FROM `nume_tabela` WHERE [conditie]

Comanda DELETE este folosita pentru stergerea datelor dintr-o tabela. De fiecare data cand rulam aceasta comanda vom sterge informatia din tabela

De exemplu daca angajatului Cami este concediat

DELETE FROM `angajati` WHERE nume="Cami"

Dupa acest query in baza vom avea

Id	Nume	Data Nasterii	Pozitie	Departament	Salariu
1	Bogdan	27/03/1983	Programator	Soft	5000
2	Ana	12/12/1987	Secretara	Marketing	2500
3	George	14/09/1977	Director	Management	8500
4	Sandra	15/12/1987	Programator	Soft	4000
5	Mircea	18/05/1956	Vanzator	Sales	3000

ATENȚIE: Daca nu se specifica o conditie asupra caror inregistrari se va face stergerea, toate inregistrările din acea tabela vor fi sterse. De exemplu query-ul



DELETE FROM `angajati`

Va avea ca rezultat ștergerea tuturor datelor din tabela ANGAJATI.



TEMA

Să se creeze o tabela *sex_angajati* cu următoarea structura Id - int(11) si sex - varchar(255) și să se insereze valorile din tabelul de mai jos

Id		Sex	
1	M	2	F
3	M	4	F
5	M		
6			F

CHEIE PRIMARĂ (PRIMARY KEY) ȘI CHEIE STRAINĂ (FOREIGN KEY)

• **Cheie Primara [PK]** - Una sau mai multe coloane ale caror valori identifica in mod unic toate liniile unui tabel. De obicei id-ul sau adresa de email este considerata cheie primara deoarece are valoare unica pentru fiecare inregistrare. Cheia primara este utilizata pentru a face referinte la o singura linie. Fara cheia primara, actualizarea sau stergerea de linii specifice devine foarte dificila, deoarece nu exista nici o modalitate garantat sigura pentru a face referire numai la liniile ce vor fi afectate. Desi cheile primare nu reprezinta o conditie necesara, cei mai multi designeri de baze de date se asigura ca toate tabelele pe care le creaza au o cheie primara, astfel incit viitoarea manipulare a datelor sa fie posibila si usor de efectuat.

Orice coloana dintr-un tabel poate fi stabilita drept cheie primara, atita timp cit respecta urmatoarele conditii:

- 1) Doua linii nu pot avea aceeasi valoare a cheii primare
- 2) Fiecare linie trebuie sa aiba o valoare a cheii primare (coloana nu poate admite valori NULL)
- 3) Coloana care contine valorile cheilor primare nu poate fi niciodata modificata sau actualizata
- 4) Valorile cheilor primare nu pot fi niciodata refolosite. Daca o linie este stearsa din tabel, cheia ei nu poate fi atribuita altor linii noi

De obicei, cheile primare sunt definite intr-o singura coloana dintr-un tabel. Acest lucru nu este inasa o conditie necesara si mai multe coloane pot fi utilizate impreuna drept cheie primara. Atunci cind se utilizeaza mai multe coloane, regulile enumerate mai sus

trebuie aplicate tuturor coloanelor, iar valorile din toate coloanele considerate laolalta trebuie sa fie unice (coloanele individuale nu trebuie sa aiba valori unice).

```
CREATE TABLE `angajati` (  
  `Id` int(11) default NULL,  
  `nume` varchar(255) default NULL,  
  `data_nasterii` date default NULL,  
  `pozitie` varchar(255) default NULL,  
  
  `departament` varchar(255) default NULL,  
  `salariu` int(11) default NULL,  
  PRIMARY KEY (Id)  
)
```

• **Cheie Străină [FK]** - Facem precizarea ca in majoritatea situatiilor practice apar legaturi intre tabele care se materializeaza prin coincidenta valorilor cheilor primare si externe(straine).

Orice coloana dintr-un tabel poate fi stabilita drept cheie straina, atata timp cat respecta urmatoarele conditii:

- 1) Prin definitie, fiecare valoare a unei chei externe trebuie sa se regaseasca printre multimea valorilor cheii candidate corespondente; reciproca nu este obligatorie
- 2) O cheie externa este simpla daca si numai daca cheia candidata corespondenta este simpla, si este compusa daca si numai daca cheia candidata corespondenta este compusa;
- 3) Fiecare atribut component al unei chei externe trebuie sa fie definit pe acelasi domeniu al componentei corespondente din cheia candidata;
- 4) O valoare a unei chei externe reprezinta o referinta catre un tuplu care contine aceeasi valoare pentru cheia candidata corespondenta; se pune astfel *problema integritatii referintei*: o baza de date nu trebuie sa contina valori invalide pentru chei externe, altfel spus daca *B* refera pe *A* atunci *A* trebuie sa existe.

```
CREATE TABLE `locatie_angajati` (  
  `locatie` int(11) default NULL,  
  `Id_angajat` int(11) default NULL,  
  PRIMARY KEY (locatie),  
  FOREIGN KEY(Id_angajat) REFERENCES angajati (Id)  
)
```

◦ QUERY-uri avansate

1) **JOIN** (sau **INNER JOIN**) permite selectarea valorilor din doua sau mai multe tabele dupa o conditie (denumita conditia de join)

SELECT coloana1, coloana2,.. FROM tabela1 INNER JOIN tabela2 ON tabela1.coloanaX = tabela2.coloanaX

Este echivalent cu query-ul de mai jos (mai simplu)

SELECT coloana1, coloana2,.. FROM tabela1, tabela2 WHERE tabela1.coloanaX = tabela2.coloanaX

Ambele producand acelasi rezultat, respectiv pentru tabelele din exemplul nostru avem query-ul

SELECT nume, sex FROM `angajati` a INNER JOIN `sex_angajati` b ON a.id = b.id

Showing rows 0 - 5 (6 total, Query took 0.0003 sec)

SQL query:

```
SELECT nume, sex
FROM `angajati` a, `sex_angajati` b
WHERE a.id = b.id
LIMIT 0, 30
```

Show: 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

nume	sex
Bogdan	M
Ana	F
George	M
Sandra	F
Mircea	M
Cami	F

2) **LEFT JOIN** – functioneaza ca JOIN doar ca numai pentru elementele din tabela din stanga (prima tabela).

SELECT * FROM angajati a LEFT JOIN sex_angajati b ON a.id = b.id

De exemplu daca din tabela **angajati** ar lipsi intrarea 6 iar din tabela **sex_angajati** ar lipsi intrarea 5 atunci query-ul ar returna rezultatul urmator

Showing rows 0 - 4 (5 total, Query took 0.0000 sec)

SQL query:

```
SELECT *
FROM angajati a
LEFT JOIN sex_angajati b ON a.id = b.id
LIMIT 0, 30
```

Show: 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

Id	nume	data_nasterii	pozitie	departament	salariu	Id	sex
1	Bogdan	1983-03-27	Programator	Soft	5000	1	M
2	Ana	1987-12-12	Secretara	Marketing	2500	2	F
3	George	1977-09-14	Director	Management	8500	3	M
4	Sandra	1987-12-15	Programator	Soft	4000	4	F
5	Mircea	1956-05-18	Vanzator	Sales	3000	NULL	NULL

respectiv doar inregistrarile ce se afla in tabela **angajati**.

3) RIGHT JOIN – functioneaza ca LEFT JOIN doar ca numai pentru elementele din tabela din dreapta (cea de-a doua tabela).

SELECT * FROM angajati a RIGHT JOIN sex_angajati b ON a.id = b.id

Folosind acelasi exemplu ca la punctul anterior vom obtine rezultatul de mai jos, respectiv doar inregistrarile ce se afla in cea de-a doua tabela **sex_angajati**.

i Showing rows 0 - 4 (5 total, Query took 0.0002 sec)

SQL query:

```
SELECT *
FROM angajati a
RIGHT JOIN sex_angajati b ON a.id = b.id
LIMIT 0, 30
```

Show: 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

Id	nume	data_nasterii	pozitie	departament	salariu	Id	sex
1	Bogdan	1983-03-27	Programator	Soft	5000	1	M
2	Ana	1987-12-12	Secretara	Marketing	2500	2	F
3	George	1977-09-14	Director	Management	8500	3	M
4	Sandra	1987-12-15	Programator	Soft	4000	4	F
NULL	NULL	NULL	NULL	NULL	NULL	6	F



Pentru mai multe detalii și exemple despre comanda JOIN (inner, outer, left, right) puteți consulta linkul <http://dev.mysql.com/doc/refman/5.0/en/join.html>

4) SUBQUERY – permite alegerea elementelor dintr-o tabela ce respecta o condiție în altă tabelă

SELECT * FROM angajati a WHERE a.id NOT IN (SELECT b.id FROM sex_angajati b)

Folosind acelasi exemplu ca la punctul anterior vom obtine rezultatul de mai jos, respectiv doar inregistrarile ce se afla in prima tabela **angajati** dar care nu se gasesc in cea de-a doua tabela **sex_angajati**.

Showing rows 0 - 0 (1 total, Query took 0.0009 sec)

SQL query:

```
SELECT *
FROM angajati a
WHERE a.Id NOT
IN (

SELECT b.Id
FROM sex_angajati b
)
LIMIT 0 , 30
```

[Edit] [Explain SQL] [Create PHP Code] [Refresh]

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

	Id	nume	data_nasterii	pozitie	departament	salariu
<input type="checkbox"/>	5	Mircea	1956-05-18	Vanzator	Sales	3000

Check All / Uncheck All With selected:



Pentru mai multe detalii si exemple despre SUBQUERIES puteți consulta linkul de mai jos <http://dev.mysql.com/doc/refman/5.1/en/subqueries.html>

- 5) **UNION** – este folosit la combinarea rezultatelor a multiple query-uri SELECT. Numele coloanelor primului select sunt folosite la rezultatul returnat.



ATENȚIE! SELECT - urile trebuie să aibă același număr de coloane și același tip de coloană pe o anumită poziție (de exemplu: dacă prima coloană din prima tabelă este de tip INT atunci și prima coloană din cea de-a doua tabelă trebuie să fie de tip INT).

(*SELECT * FROM angajati*) UNION (*SELECT * FROM angajati2*)

Query-ul va returna rezultatul urmator, respectiv suma rezultatelor celor doua SELECT-uri.

OBSERVATIE. Cu ajutorul comenzii UNION se pot concatena rezultatele a mai multe query-uri SELECT, nu neaparat doar doua.

Showing rows 0 - 7 (8 total, Query took 0.0006 sec)

SQL query:

```
SELECT *
FROM angajati
UNION
SELECT *
FROM angajati2
```

Show : 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

	Id	nume	data_nasterii	pozitie	departament	salariu
<input type="checkbox"/>	1	Bogdan	1983-03-27	Programator	Soft	5000
<input type="checkbox"/>	2	Ana	1987-12-12	Secretara	Marketing	2500
<input type="checkbox"/>	3	George	1977-09-14	Director	Management	8500
<input type="checkbox"/>	4	Sandra	1987-12-15	Programator	Soft	4000
<input type="checkbox"/>	5	Mircea	1956-05-18	Vanzator	Sales	3000
<input type="checkbox"/>	1	Cristina	1983-03-27	Asistenta	Soft	2000
<input type="checkbox"/>	2	Georgiana	1987-12-12	Secretara	Marketing	1500
<input type="checkbox"/>	3	Alin	1977-09-14	Director	Management	8500



Pentru mai multe detalii si exemple despre comanda UNION puteți consulta linkul de mai jos <http://dev.mysql.com/doc/refman/5.0/en/union.html>

◦ Cuvinte cheie și funcții MySQL

÷ Cuvinte cheie

1) **DISTINCT** – permite selectarea doar a valorilor distincte

SELECT DISTINCT coloana FROM tabela

De exemplu din tabela angajati va selecta doar departamentele distincte folosind query-ul

SELECT DISTINCT department FROM `angajati`

Showing rows 0 - 3 (4 total, Query took 0.0009 sec)

SQL query:

```
SELECT DISTINCT department
FROM `angajati`
WHERE 1
LIMIT 0, 30
```

Show: 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

	department
<input type="checkbox"/>	Soft
<input type="checkbox"/>	Marketing
<input type="checkbox"/>	Management
<input type="checkbox"/>	Sales

- 2) **LIKE** – permite compararea valorii cu un pattern simplu si va rezulta 1 daca este adevarat sau 0 daca este fals

SELECT coloana FROM tabela WHERE coloana LIKE "text%"

De exemplu daca dorim sa selectam din tabela angajati doar departamentele executive (Marketing, Management) ce au in comun faptul ca incep cu literele "Ma", query-ul va fi

SELECT * FROM `angajati` WHERE department LIKE "Ma%"

Si va returna rezultatul de mai jos:

Showing rows 0 - 1 (2 total, Query took 0.0005 sec)

SQL query:

```
SELECT *
FROM `angajati`
WHERE departament LIKE "Ma%"
LIMIT 0, 30
```

Show: 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

			Id	nume	data_nasterii	pozitie	departament	salariu
<input type="checkbox"/>			2	Ana	1987-12-12	Secretara	Marketing	2500
<input type="checkbox"/>			3	George	1977-09-14	Director	Management	8500

÷ Funcții MySQL

3) **SUBSTRING** - permite returnarea unui substring din stringul existent

SELECT SUBSTRING(camp, pozitia initiala, numar de caractere) ;

De exemplu query-ul

SELECT SUBSTRING('Bogdan', 1, 3) ;

va returna rezultatul de mai jos, respectiv va lua din stringul *Bogdan* substring-ul incepand de la primul caracter si avand lungimea de trei caractere, respectiv va returna sirul de caractere *Bog*

Showing rows 0 - 0 (1 total, Query took 0.0005 sec)

SQL query:

```
SELECT SUBSTRING( 'Bogdan', 1, 3 );
```

Show: 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

← T →

SUBSTRING('Bogdan',1,3)
Bog

Show: 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

Alte functii pentru stringuri foarte utilizate sunt **LOCATE** (care permite cautarea unui substring intr-un string si returneaza pozitia acestuia daca este gasit si 0 altfel), **LENGTH** (care returneaza dimensiunea unui string) etc.

Pentru a returna doar numele de familie din string-ul 'Bogdan Roman' folosim query-ul următor:

```
SELECT SUBSTRING( 'Bogdan Roman', LOCATE( ' ', 'Bogdan Roman' ) , LENGTH( 'Bogdan Roman' ) - LOCATE( ' ', 'Bogdan Roman' ) + 1 )
```

Care va returna rezultatul de mai jos, respectiv numele de familie

Showing rows 0 - 0 (1 total, Query took 0.0005 sec)

SQL query:

```
SELECT SUBSTRING( 'Bogdan Roman', LOCATE( ' ', 'Bogdan Roman' ) , LENGTH( 'Bogdan Roman' ) - LOCATE( ' ', 'Bogdan Roman' ) + 1 )
```

Show: 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

← T →

SUBSTRING('Bogdan Roman', LOCATE(' ', 'Bogdan Roman') , LENGTH('Bogdan Roman') - LOCATE(' ', 'Bogdan Roman') + 1)
Roman



Pentru lista completa de funcții pentru string-uri, inclusiv exemple, puteți merge la adresa <http://dev.mysql.com/doc/refman/5.0/en/string-functions.html>

4) **NOW()** – este utilizat pentru a specifica timpul si data curenta a serverului

SELECT NOW();

Showing rows 0 - 0 (1 total, Query took 0.0004 sec)

SQL query: `SELECT NOW()`

Show: 30 row(s) starting from record # 0
in horizontal mode and repeat headers after 100 cells

NOW()
2010-10-20 09:56:41

INSERT INTO `angajati2`(`Id`, `nume`, `data_nasterii`, `pozitie`, `departament`, `salariu`) VALUES ('9','Andreea',now(),'Copil','Testare','1000');



Pentru lista completă de funcții pentru dată, inclusiv exemple, puteți merge aici <http://dev.mysql.com/doc/refman/5.1/en/date-and-time-functions.html>

☺ Exemplul 5 – Utilizarea efectivă a query-urilor în aplicații

Sa se creeze o baza de date cu toti studentii dintr-o facultate. Baza de date va avea doua tabele **SERIA** (*codSerie INT(2) NOT NULL, denSerie varchar(15) NOT NULL, nrGrupe INT(2) DEFAULT 0, an INT(1) DEFAULT 1*) si **GRUPA** (*codGrupa INT(2)*

NOT NULL, denGrupa varchar(15) NOT NULL, nrStudenti INT(3) DEFAULT 0, codSerie INT(2), media FLOAT NOT NULL,). Sa se populeze cele doua tabele cu inregistrari, astfel incat sa putem selecta denumirile si numarul studentilor din grupele care sunt in seria A sau in seria C, selectam denumirea, numarul studentilor si media pentru toate grupele care au o medie mai mare de nota 9.00 si sa selectam pentru fiecare grupa din seriile cu cod par: nume grupa, cod grupa, nr studenti, nume serie din care face parte.

*/*creare tabele serie si grupa*/*

```
CREATE TABLE serie_rb( codSerie INT(2) NOT NULL,
                        denSerie varchar(15) NOT NULL,
                        nrGrupe INT(2) DEFAULT 0, an
                        INT(1) DEFAULT 1, PRIMARY
                        KEY(codSerie)
                        )
```

```
CREATE TABLE grupa_rb( codGrupa INT(2) NOT NULL,
                        denGrupa varchar(15) NOT NULL,
                        nrStudenti INT(3) DEFAULT 0,
                        codSerie INT(2),
                        media FLOAT NOT NULL,
                        PRIMARY KEY(codGrupa),
                        FOREIGN KEY(codSerie) REFERENCES serie_rb(codSerie)
                        )
```

*/*populare tabele*/*

```
INSERT INTO serie_rb(codSerie,denSerie,nrGrupe,an) VALUES (1,'SeriaA',4,1);
INSERT INTO serie_rb(codSerie,denSerie,nrGrupe,an) VALUES (2,'SeriaB',3,1);
INSERT INTO serie_rb(codSerie,denSerie,nrGrupe,an) VALUES (3,'SeriaC',4,2);
INSERT INTO serie_rb(codSerie,denSerie,nrGrupe,an) VALUES (4,'SeriaFinala',3,2);
INSERT INTO grupa_rb(codGrupa,denGrupa,nrStudenti,codSerie,media) VALUES
(1,'Grupa1',10,1,10.00);
INSERT INTO grupa_rb(codGrupa,denGrupa,nrStudenti,codSerie,media) VALUES (2,'Grupa2',11,2,10);
INSERT INTO grupa_rb(codGrupa,denGrupa,nrStudenti,codSerie,media) VALUES (3,'Grupa3',12,3,9);
INSERT INTO grupa_rb(codGrupa,denGrupa,nrStudenti,codSerie,media) VALUES (4,'Grupa4',13,4,8);
INSERT INTO grupa_rb(codGrupa,denGrupa,nrStudenti,codSerie,media) VALUES (5,'Grupa5',14,1,9.50);
INSERT INTO grupa_rb(codGrupa,denGrupa,nrStudenti,codSerie,media) VALUES
(6,'GrupaFinala',15,2,10);
```

*/*afisare rezultat*/*

```
SELECT * FROM serie_rb;
SELECT * FROM grupa_rb;
```

*/*selectati denumirile si numarul studentilor din grupele care sunt in seria A sau in seria C*/*

```
SELECT denGrupa, nrStudenti FROM serie_rb a, grupa_rb b WHERE (a.denSerie = 'SeriaA' OR
a.denSerie = 'SeriaC') AND a.codSerie = b.codSerie;
```

*/*selectati denumirea, numarul studentilor si media pentru toate grupele care au o media mai mare de 9.00*/*

SELECT denGrupa, nrStudenti, media FROM grupa_rb WHERE media > 9.00;

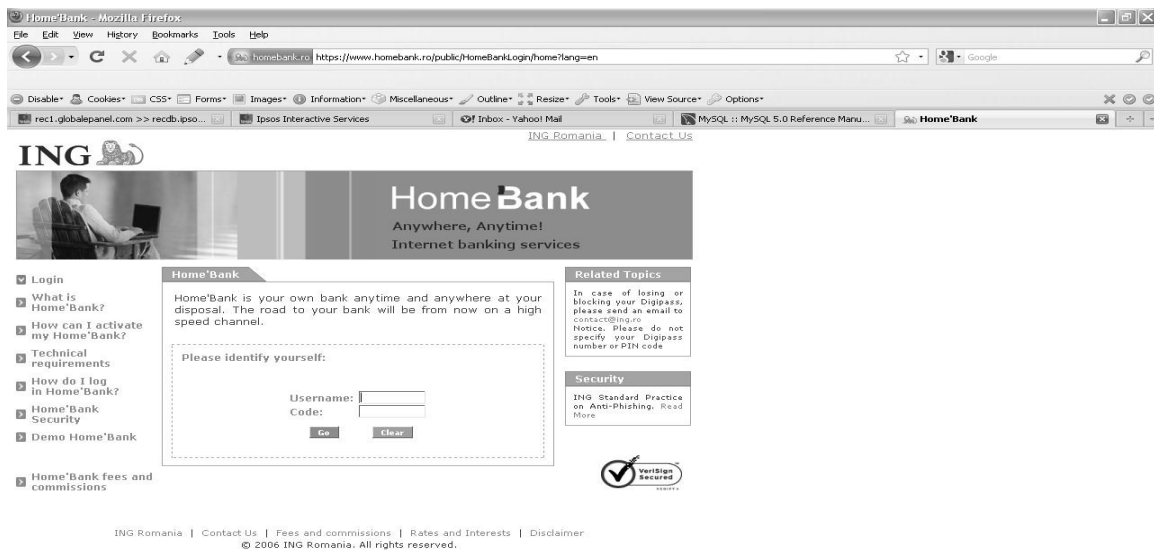
/*selectati pentru fiecare grupa din seriile cu cod par: nume grupa, cod grupa, nr studenti, nume serie din care face parte */

SELECT denGrupa, codGrupa, nrStudenti, denSerie FROM serie_rb a, grupa_rb b WHERE mod(a.codSerie,2) = 0 AND a.codSerie = b.codSerie;



TEMĂ

Să se creeze o bază de date pentru o bancă de tipul celor folosite pe site-urile de home banking <https://www.homebank.ro/>.



Baza de date trebuie sa conțină minim 3 tabele. Să se populeze cele două tabele cu următoarele înregistrări:

• **CONT**

nrCont VARCHAR(15) NOT NULL	expl VARCHAR(30)	valDebit INT(2)	valCredit INT(2)
401	Furnizori	0	1000
411	Clienti	2000	0
5121	Banca	1500	3000

• **CEC**

idCec VARCHAR(15) NOT NULL	valCec INT	dataCec date
1	500	2001-11-4
2	600	2001-12-5
3	100	2001-11-10

• **POZITIE_CEC**

idCec VARCHAR(15) NOT NULL	pozCec INT(3) NOT NULL	valPoz INT	contDeb VARCHAR(10) NOT NULL	contCred VARCHAR(10) NOT NULL
1	1	200	401	5121
1	2	400	401	5121
2	1	300	5121	411
2	2	0	401	411
2	3	100	5121	411
3	1	100	401	411

Se cer următoarele:

- 1) Să se selecteze pentru fiecare cec identificatorul, și suma reală a pozițiilor sale (de ex. pt cec-ul nr 2 suma este 400)
- 2) Să se selecteze pentru fiecare poziție a cec-ului emis pe 5-12-2001, valoarea poziției, valoarea declarată a cec-ului din care face parte, denumirea contului debitor și denumirea contului creditor cu care a fost înregistrată (MONTHNAME returnează denumirea lunii)
- 3) Sa se selecteze pentru fiecare pozitie a cec-ului emis pe 5-12-2001, valoarea pozitiei, valoarea declarata a cec-ului din care face parte, denumirea contului debitor si denumirea contului creditor cu care a fost inregistrata
- 4) Sa se selecteze id-ul cecului din care face parte, pozitia (pozCec), si valoarea pentru a doua pozitie ca valoare din sistem (pozitia care are campul valPoz = 300)

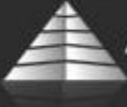
II.4. Protejarea la SQL Injection

Daca se preia informatie de la utilizatorul unui website prin intermediul unui formular si se doreste inserarea informatiei intr-o baza de date exista o sansa ca utilizatorul, daca este rau intentionat (hacker), sa poata sterge sau modifica informatia din baza de date fara permisiunea noastra.

EXEMPLU DE ATAC prin SQL Injection

Va rugam completati datele personale pentru transport

Name	<input type="text" value="george"/>
Prenume	<input type="text" value="popescu"/>
Parola	<input type="password" value="....."/>
Sex	<input type="text" value="Barbat"/> ▼
Telefon	<input type="text"/>
Email	<input type="text"/>

 **ABSOLUTE SCHOOL**
Împreună, pentru un viitor mai bun

De exemplu: dacă noi dorim să selectăm din baza de date informația indicată de utilizatorul site-ului când se loghează pe site

```
SELECT * FROM `clienti` WHERE `parola` = ' $valoare_provenita_de_la_utilizator'
```

Dacă utilizatorul introduce în căsuță parola sa de exemplu **parola_valida** atunci query-ul este

```
SELECT * FROM `clienti` WHERE `parola` = 'parola_valida'
```

Dacă user-ul respectiv are parola **parola_valida** atunci scoatem informația despre acest user din baza de date.

Totuși dacă utilizatorul este rău intenționat poate introduce în căsuța de login textul **parola_invalida' OR 1 OR client='parola_invalida**

Atunci query-ul devine

```
SELECT * FROM `clienti` WHERE `parola` = 'parola_invalida' OR 1 OR client='parola_invalida'
```

Care va verifica

- 1) dacă parola clientului este '**parola_invalida**' ceea ce este fals
- 2) dacă 1 care este mereu adevărat
- 3) dacă numele clientului este '**parola_invalida**' ceea ce este fals

și vom avea rezultatul FALS(0) OR ADEVARAT(1) OR FALS(0) = ADEVARAT(1) deci va permite accesul la datele acelui client.

Exista, însă, metode de protecție împotriva acestor atacuri:

- I. **ESCAPAREA DATELOR** – exista o functie PHP `mysql_real_escape_string` (\$informatie) care pune un \ in fata fiecarui apostrof si acesta nu va mai trece peste conditia de mai devreme

```
SELECT * FROM `clienti` WHERE `parola` = 'parola_invalida\' OR 1 OR client=\'parola_invalida'
```

Care va verifica daca parola clientului este **parola_invalida\' OR 1 OR client=\'parola_invalida** ceea ce nu este adevarat deci va returna fals.

- II. **INCAPSULAREA** datelor – toate valorile din query-uri trebuiesc puse intre apostroafe pentru a evita aceleasi atacuri

```
SELECT * FROM `clienti` WHERE `parola` = parola_invalida
```

Atunci va returna fals daca parola nu este corecta dar daca se introduce

```
SELECT * FROM `clienti` WHERE `parola` = parola_invalida OR 1
```

Va verifica :

1) daca parola este cea corecta ceea ce nu este adevarat 2) daca 1 care este mereu adevarat si vom avea rezultatul FALS(0) SAU ADEVARAT(1) = ADEVARAT (1)

ATENȚIE: Acest aspect de protecție a informației este foarte important pentru programatorii web, făcând diferența între un programator web bun și unul slab. Iar o dată ce unui client i-a dispărut toată informația de pe site, este garantat că nu va mai dori să facă afaceri cu dumneavoastră.



Unitatea de învățare 3



Limbajul PHP - variabile, funcții, structuri de control

III.1. Prezentarea limbajului de programare PHP

- **Limbaje de programare server-side și client-side**
- **Prezentare generala PHP**

III.2. Introducere în PHP

- **Fișier PHP**
- **Variabile PHP**
- **Operatori PHP**
- **Structuri PHP**
 - + de control If ... Else, Switch
 - + repetitive For, While
- **Funcții PHP**



III.1. Prezentarea limbajului de programare PHP

◦ Limbaje de programare server-side și client-side

La momentul în care o pagină web este încărcată de către browser, există în acea pagină codul specific unui limbaj de programare ce rulează pe serverul web (așa numitele limbaje de programare server-side ca PHP, ASP, .NET etc.), cod specific al unui limbaj de programare client-side (JavaScript, jScript) și cod HTML.

Ordinea de parcurgere și încărcare a unei pagini web este următoarea:

- 1) Se parcurge și se încarcă codul scris în limbajul de programare server-side, acest cod se încarcă, o singura dată, la început, iar dacă vrem să facem modificări în el trebuie să retrimitem informația către server și să reîncărcăm răspunsul. Acest proces este denumit și **refresh** –ul paginii web.
- 2) Se parcurge și se încarcă codul din HTML
- 3) Se parcurge și se încarcă codul scris în limbajul de programare client-side care aranjează în pagina elementele HTML modificându-le proprietățile, stilurile, valorile etc.

! ATENȚIE. Browserele noi (Google Chrome , Apple Safari ) folosesc o tehnologie diferită denumită **WebKit** ce permite încărcarea mai rapidă a paginilor web prin încărcarea în paralel a HTML-ului și JavaScript-ului. Din acest motiv pot apărea erori la schimbarea proprietăților elementelor HTML.

În exemplul de mai jos (**Exemplul 5**) putem observa structura unei pagini web, cu toate cele 3 limbaje incluse în același fișier.

- Limbajul HTML – textul normal
- Limbajul PHP – textul *italic*
- Limbajul JavaScript – textul **bold**

ATENȚIE: Pentru o scriere mai ușoară și corectă a codului, diversele limbaje trebuie structurate și împărțite în fișiere separate.



☺ **Exemplul 6** – Structura unei pagini web

```

<?php
.....
if(isset($_POST["submitted"]) && $_POST["submitted"]=='yes'){ if(isset($_POST["user"]) &&
    $_POST["user"]!='' && isset($_POST["pass"]) &&
    $_POST["pass"]!=''){
    .....
        }

    } if ($_SESSION["pass"] ==
    sha1($user_password)){ header('Location: p3.php');
    }else{
        $eroare = 'User / parola incorecta';
    }
}

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Absolute School - Programare Web</title>
<script type="text/javascript" src="./js/funcs.js"></script>
<link rel="stylesheet" type="text/css" href="<?php echo INCLUDE_PATH?>/css/main.css" />
</head> <script>
function check_form(){
    .....

    if(user_ok && pass_ok)
        document.forma.submit();

    else alert('Va rugam completati campul '+eroare);
}

</script>
<body>

<form method="post" name="forma" id="forma" >
    <div style="color:#193153;width:100px;">User</div><div> <input type="text" id="user"
name="user" value="<?php echo $user;?>" /></div>
    .....
    <input type="hidden" name="submitted" id="submitted" value="yes" />
    <input type="button" name="buton" value="Trimite" onClick="check_form()" />
</form>
<div style="color:#FF0000;"><?php echo $eroare;?></div><br /><br />
Noi suntem <a href="http://www.absoluteschool.net/" target="_blank">ABSOLUTE SCHOOL</a>

</body>
</html>

```

PHP – prezentare generală

PHP: Hypertext Pre – Processor este un limbaj de programare server-side folosit la scară largă în programarea web.

Din acest motiv codul PHP este integrat în documentul sursă și interpretat de către un server web pe care este instalat un modul PHP, care generează documentul web în format HTML și îl trimite browserului de pe computerul client.

Principalele avantaje ale PHP-ului sunt:



- este un limbaj de programare stabil
- este software open source
- este gratuit de downloadat și folosit de la website-ul oficial <http://www.php.net>
- este o alternativa viabilă împotriva competitorilor (ASP, Ruby etc.)
- suporta multe baze de date (MySQL, Informix, Oracle, Sybase, Solid, PostgreSQL, MSSQL etc.)
- poate rula pe diferite platforme (Windows, Linux, Unix etc.)
- este compatibil cu aproape toate serverele folosite în ziua de azi (Apache, IIS, etc.)
- este relativ ușor de învățat și rulează eficient pe serverul web

III.2. Introducere în PHP

◦ Fișierele PHP

Sunt fișiere text ce au extensia **.php**, **.php3**, **.phtml**, **.php5**, **.phps**. Extensia este importantă pentru server care trebuie să știe să interpreteze conținutul acestor fișiere ca fiind cod PHP.

Un script PHP începe întotdeauna cu tagul **<?php** și se încheie cu tagul **?>**

```
<?php
// pentru a comenta un singur rand in PHP se foloseste echo
"Salut!";

/* pentru a comenta un bloc de mai multe
   randuri in PHP se foloseste */
?>
```

Scriptul PHP se poate insera oriunde in document, de asemenea un document poate avea mai multe blocuri de script PHP

```
<?php echo
"Inceput"; ?>
```

```
<!--cod HTML -->
```

```
<?php
echo "Sfarsit";
?>
```

Pe serverele ce au forma scurta (shorthand in limba engleza) enabled putem scrie blocul PHP intre tagurile

```
<? echo
"Salut!"; ?>
```

Sau pentru afisare putem folosi direct

```
<?="Salut!";?>
```

ATENȚIE: Pentru o compatibilitate maximă se recomandă folosirea tag-ului standard **<?php** în locul formei scurte **<?**.



◦ Variabile PHP

O variabila este folosita pentru a pastra informatia text, numerica sau vectori (array). Dupa ce a fost declarata o variabila ea poate fi folosita peste tot in script. Toate variabilele din PHP incep cu simbolul \$. In PHP variabilele se declara la atribuire (spre deosebire de alte limbaje de programare ca C, C++, JavaScript unde se declara variabile la inceput si se atribuie pe parcurs).

Modul corect de declarare a unei variabile in PHP este urmatorul:

```
$nume = "bogdan"; // variabila de tip text sau string
```

```
$varsta="17";      // variabila de tip numeric
```



```

$lista_nume = array( //variabila de tip vector pe un nivel
    '1'=>'bogdan',
    '2'=>'andreea',
    '3'=>'catalin'

);

$lista_date_personale = array( //variabila de tip vector pe mai multe nivele nivel

    '1'=>array(
        'nume'=>'bogdan',
        'varsta'=>'27',
    ),
    '2'=>array(
        'nume'=>'andreea',
        'varsta'=>'25',
    ),
    '3'=>array(
        'nume'=>'catalin',
        'varsta'=>'30',
    ),

);

$conditie = true; //variabila de tip boolean (poate lua valorile Adevarat/Fals)

```

Spre deosebire de alte limbaje de programare se poate observa ca PHP converteste automat variabila la tipul dorit (de exemplu daca dorim ca **\$nume** sa fie de tip text, doar ii atribuim un text si variabila devine de tip text).

Reguli de denumire a variabilelor in PHP:

- 1) trebuie sa inceapa cu litera sau _
- 2) poate contine doar caractere alfanumerice (litere **0-9** si cifre **a-zA-Z**) si _
- 3) o variabila nu poate contine spatii in numele sau

OBSERVAȚIE: Există o serie de variabile predefinite PHP ce permit stocarea de informatii (\$_GET, \$_POST care colecteaza valori din formular, \$_SESSION care colecteaza valorile stocate in sesiuni, \$_COOKIES etc.). Regula de formare a acestora este urmatoarea

`$_VARIABILA_PREDEFINITA['nume_element_html']`

☺ Exemplul 7 – Captarea datelor dintr-un formular pe `$_GET` sau `$_POST`

Exemple de astfel de formulare se găsesc în paginile de contact de pe multe website-uri, în general cele care vând produse sau servicii fiind o modalitate facilă de dialog între proprietarii și utilizatorii website-ului (site-ul de mâncare chinezească WU XING <http://www.wuxing.ro/ro/trimite-mesaj/> este un exemplu în acest sens)

• `$_GET`

```
<?php if(isset($_GET["fname"]) && $_GET["fname"]!=""){ echo
    $_GET["fname"]." are ".$_GET["age"]." ani!<br />"; }else{
?>
<form method="get">
<div style="width:200px;">Nume: </div><div style="width:200px;"><input type="text" name="fname"
width="200"/></div>
<div style="width:200px;">Varsta: </div><div style="width:200px;"><input type="text" name="age"
width="200"/></div>
<div style="width:200px;"><input type="submit" value="Trimite Datele pe GET" width="200"/></div>
</form>
<?php } ?>
```

• \$_POST

```
<?php if(isset($_POST["fname"])) && $_POST["fname"]!=""){ echo
    $_POST["fname"]." are ".$_POST["age"]." ani!<br />"; }else{
?>
<form method="post">
<div style="width:200px;">Nume: </div><div style="width:200px;"><input type="text" name="fname"
width="200"/></div>
<div style="width:200px;">Varsta: </div><div style="width:200px;"><input type="text" name="age"
width="200"/></div>
<div style="width:200px;"><input type="submit" value="Trimite Datele pe GET" width="200"/></div>
</form>
<?php } ?>
```

◦ Operatori PHP

Operatori Aritmetici

Operator	Descriere	Exemplu	Rezultat
+	Adunare	x=2 x+2	4
-	Scadere	x=2 5- x	3
*	Inmultire	x=4 x*5	20
/	Impartire	15/5 5/2	3 2.5
%	Modulo n	5%2 10%8 10%2	1 2 0
++	Incrementare [+1]	x=5 x++	x=6
--	Decrementare [-1]	x=5 x- -	x=4

Operatori de atribuire

Operator	Exemplu	Similar cu
=	x=y	x=y
+=	x+=y	x=x+y
-=	x-=y	x=x-y
=	x=y	x=x*y
/=	x/=y	x=x/y
.=	x.=y	x=x.y

%=	x%=y	x=x%y
----	------	-------

Operatori de comparare

Operator	Descriere	Exemplu
==	Egal cu	5==8 returns fals
!=	Diferit	5!=8 intoarce adevarat
<>	Diferit	5<>8 intoarce adevarat
>	Mai mare decat	5>8 intoarce fals
<	Mai mic decat	5<8 intoarce adevarat
>=	Mai mare sau egal cu	5>=8 intoarce fals
<=	Mai mic sau egal cu	5<=8 intoarce adevarat

Operatori logici

Operator	Descriere	Exemplu
&&	SI	x=6 si y=3 (x < 10 && y > 1) intoarce adevarat
	SAU	x=6 si y=3 (x==5 y==5) intoarce fals
!	NOT	x=6 si y=3 !(x==y) intoarce adevarat

*tabele preluate de pe <http://www.w3schools.com>

◦ Structuri PHP

• condiționale

IF...ELSE — o anumită porțiune de cod se execută doar dacă este adevărată condiția, altfel se execută altă porțiune de cod

Sintaxa:

if (conditie) cod ce se executa daca este adevarata conditia; else cod ce se executa daca este falsa conditia;

Exemplu:

```
<?php
$d=date("D"); if($d=="Sat"
|| $d=="Sun"){
    echo "E week-end ☺!";
}else{
    echo "Nu e inca week-end ☹!";
}
?>
```

IF...ELSEIF...ELSE — această instrucțiune se folosește pentru a alege din mai multe opțiuni posibile.

Sintaxa:

if (conditie1) cod ce se execută dacă este adevărată condiția; elseif (conditie2) cod ce se execută dacă este adevărată condiția; else cod ce se execută dacă este falsă condiția;

Exemplu:

```
<?php
$d=date("D"); if($d=="Sat"
|| $d=="Sun"){ echo "E
week-end ☺!";
}elseif($d=="Fri"){
    echo "Nu e inca week-end dar este aproape!";
}else{ echo "Nu e inca week-end ☹!";
}
?>
```

SWITCH – similar cu IF...ELSEIF...ELSE alege una din multele opțiuni posibile.

Sintaxa:

```
switch (variabila)  
{ case valoare1: cod ce este executat daca variabila=valoare1; break;  
case valoare2: cod ce este executat daca variabila=valoare2; break;  
default: cod ce este executat daca variabila e diferita de valoare1 si de  
valoare2; }
```

Exemplu:

```
<?php  
switch ($loc)  
{  
    case 1:  
        echo "Medalia de aur";  
        break;  
    case 2:  
        echo "Medalia de argint";  
        break;  
    case 3:  
        echo "Medalia de bronz";  
        break;  
    default:  
        echo "Nu a terminat pe podium";  
}  
?>
```

• repetitive

WHILE – execută o porțiune de cod de un număr specificat de ori, sau atâta timp cât o condiție este adevărată.

Sintaxa:

```
while (conditie)  
{  
    executa cod;  
}
```

Exemplu:

```
<?php  
$i=10;  
while($i>0){  
    echo "Au mai ramas " . $i . " secunde pana la anul nou!<br />";  
    $i--;  
}
```

```
echo "LA MULTI ANI!";  
?>
```

FOR – execută o porțiune de cod de un număr specificat de ori, se folosește când știm exact de câte ori trebuie executat codul.

Sintaxa:

```
for (initial; conditie; final)  
{  
    cod de executat;  
}
```

Exemplu:

```
<?php  
  
$nota = array('10','9','7');  
echo "Elevul Bogdan are urmatoarele note ";  
for ($i=0; $i<=2; $i++)  
{  
    echo $nota[$i] . " ";  
}  
  
?>
```

◦ **Funcții PHP**

◦ **FUNCTII PREDEFINITE**

```
$nume = "bogdan";
```

• **funcții pentru folosirea variabilelor**

- 1) `isset ($nume)` – determina daca o variabila este setata (are valoare) sau e NULL Ex:
`isset($nume) ? true`
- 2) `is_string($nume)` – determina daca o variabila este string
Ex: `is_string($nume) ? true`
- 3) `is_array($nume)` – determina daca o variabila este array
Ex: `is_array($nume) ? false`



Lista completă de funcții PHP pentru string-uri o puteți găsi accesând link-ul de mai jos: <http://www.php.net/manual/en/ref.var.php>

• funcții pentru variabile de tip text(string)

- 1) **echo** – afisare variabila Ex: echo \$nume; ➤ Bogdan
- 2) **.** – concatenare de text
Ex: echo \$nume.” scrie un text.”; ➤ Bogdan scrie un text.
- 3) **strlen** (\$string) – returneaza numarul de caractere
Ex: strlen(\$nume) ➤ 6
- 4) **strpos** (\$string) – cauta un sir de caractere intr-un string si returneaza pozitia pe care il gaseste
Ex: strpos(\$nume,'gd') ➤ 2



ATENȚIE: Prima pozitie dintr-un string este 0 si nu 1

- 5) **substr** (\$string, \$start, \$dimensiune) – scoate un substring de dimensiune \$dimensiune dintr-un string existent incepand cu caraterul \$start

Ex: substr(\$nume,1,3) ➤ ogd

- 6) **str_replace** (\$vechi, \$nou, \$string) – inlocuieste in stringul \$string sirul de caractere \$vechi cu sirul de caractere \$nou

Lista completa de functii PHP pentru stringuri o puteti gasi accesand linkul de mai jos: <http://www.php.net/manual/en/ref.strings.php>

• funcții pentru variabile de tip data

- 7) **date()** – returneaza data curenta in functie de parametrii Y- an, m – luna, d – zi
Ex: date('Y-m-d') ➤ 2010-06-18

8) **time()** – returneaza numarul de secunde care au trecut de la data de 1 Ianuarie 1970 00:00:00

Ex: time() 1276868215

• funcții pentru trimitere email

9) **mail()** – trimite email catre o adresa specificata **mail (destinatar , subject, mesaj, [headere], [parametrii])**

÷ *Exemplu de email plain/text:*

```
if(isset($_GET["fname"]) && $_GET["fname"]!=""){
    echo "Emailul a fost trimis";

    $emailaddress = 'romanbogdan2783@gmail.com';
    $emailsubject = 'Aveti un email';
    $emailsubject = 'Aveti un mesaj de la ' . $_GET["fname"];

    mail($emailaddress, $emailsubject, $msg);
}
```

ATENȚIE: Acest tip de email nu este recomandat sa fie trimis din cauza lipsei headerelor va fi rejectat de filtrul anti-spam al adresei de email (@gmail, @yahoo etc.)



÷ *Exemplu de email HTML:*

```
$inbox = "romanbogdan2783@gmail.com";

$mailTo = $inbox;
$emailsubject = $text[$_language]['contact']['mail_subject'];
$body = "Message from ".$fname . " " . $lname . " " . $cname . " :<br>".$question;

function send_mail($emailaddress, $fromaddress, $emailsubject, $body, $attachments=false)
{
    $eol="\r\n";
    $mime_boundary=md5(time());

    # Common Headers
    $headers = 'From: Paradigma Solutions<'.$fromaddress.'>'.$eol;
    $headers .= 'Reply-To: '.$fromaddress.'<'.$fromaddress.'>'.$eol;
    $headers .= 'Return-Path: restaurant<'.$fromaddress.'>'.$eol; // these two to set reply address
    $headers .= "Message-ID: <". $now . " Admin@ " . $_SERVER['SERVER_NAME'] . ">".$eol;
    $headers .= "X-Mailer: PHP v".phpversion().$eol; // These two to help avoid spam-filters

    # Boundry for marking the split & Multitype Headers
```

```

$headers .= 'MIME-Version: 1.0'. $eol;
$headers .= "Content-Type: multipart/related; boundary=\"". $mime_boundary. "\". $eol;

$msg = "";

if ($attachments !== false)
{
    for($i=0; $i < count($attachments); $i++)
    {
        if (is_file($attachments[$i]["file"]))
        {
            # File for Attachment
            $file_name = substr($attachments[$i]["file"], (strpos($attachments[$i]["file"], "/")+1));

            $handle=fopen($attachments[$i]["file"], 'rb');
            $f_contents=fread($handle, filesize($attachments[$i]["file"]));
            $f_contents=chunk_split(base64_encode($f_contents)); //Encode The Data For Transition using
            base64_encode();
            fclose($handle);

            # Attachment
            $msg .= "--". $mime_boundary. $eol;
            $msg .= "Content-Type: ".$attachments[$i]["content_type"]."; name=\"". $file_name. "\". $eol;
            $msg .= "Content-Transfer-Encoding: base64". $eol;
            $msg .= "Content-Disposition: attachment; filename=\"". $file_name. "\". $eol. $eol; // !! This line
            needs TWO end of lines !! IMPORTANT !!
            $msg .= $f_contents. $eol. $eol;

        }
    }
}

# Setup for text OR html
$msg .= "Content-Type: multipart/alternative". $eol;

# Text Version
$msg .= "--". $mime_boundary. $eol;
$msg .= "Content-Type: text/plain; charset=iso-8859-1". $eol;
$msg .= "Content-Transfer-Encoding: 8bit". $eol;
$msg .= strip_tags(str_replace("<br>", "\n", $body)). $eol. $eol;

/* # HTML Version
$msg .= "--". $mime_boundary. $eol;
$msg .= "Content-Type: text/html; charset=iso-8859-1". $eol;
$msg .= "Content-Transfer-Encoding: 8bit". $eol;
$msg .= $body. $eol. $eol;*/

# Finished
$msg .= "--". $mime_boundary. "--". $eol. $eol; // finish with two eol's for better security. see Injection.

```

```
# SEND THE EMAIL
ini_set(sendmail_from,$fromaddress); // the INI lines are to force the From Address to be used !
mail($emailaddress, $emailsubject, $msg, $headers); ini_restore(sendmail_from);
//echo "mail send";
} send_mail ($mailto, $email, $emailsubject, $body,

$attachments);
```



Pentru mai multe detalii puteți studia manualul pentru aceasta funcție de la adresa <http://www.php.net/manual/en/function.mail.php>

• funcții pentru upload de fișiere

10) **move_uploaded_file()** – uploadeaza fisierul din formular pe server

move_uploaded_file (nume_fisier , destinatie)

Exemplu:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<!--<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />-->
<title>Absolute School</title>
<script type="text/JavaScript" src="js/funcs.js"></script>
<link rel="stylesheet" type="text/css" href="css/main.css" />
</head>
<body style="color:#193153;">
<?php if(isset($_POST["fname"])) && $_POST["fname"]!=""){
    $errMsg = 'noError';
    if (($_FILES["file"]["size"] < 2000000) && ($_FILES["file"]["error"] == 0)){
        $file = $_FILES["file"]["name"];
    }else{
        $errMsg = 'eroare';
    }
    if($errMsg='noError'){
        move_uploaded_file($_FILES["file"]["tmp_name"],"download/".$
$_FILES["file"]["name"]); echo "Fisierul a fost uploadat cu
succes";
    }else{ echo "Fisierul nu a fost uploadat cu succes"; }
}
}
}
```

?>

```
<form method="post" enctype="multipart/form-data">
<div style="width:100px;">Nume: </div><div style="width:200px;"><input type="text" name="fname"
width="200"/></div>
<div style="width:100px;">Fisier: </div><div style="width:200px;"><input type="file" name="file"
id="file" /></div>
<div style="width:300px;padding-top:15px;"><input type="submit" value="UPLOAD"
width="200"/></div> </form>

<div style="margin-top:300px;background-color:#193153;width:100%;text-align:center;"> </div>
<?php } ?>
</body>
</html>
```



Pentru mai multe detalii puteți studia manualul pentru aceasta funcție de la adresa <http://www.php.net/manual/en/function.move-uploaded-file.php>

◦ FUNCȚII DEFINITE DE UTILIZATOR

```
function nume_funcie(var1,var2,...){

    //cod ce trebuie executat return

    $rezultat;

}
```

ATENȚIE: O variabilă poate fi **globală** dacă este definită în afara funcției sau **locală** dacă este definită într-o funcție



😊 Exemplul 8 – Funcție PHP

```
<?php
$a = 5; $b = 6; function
inmultire($a,$b){
```

```

        $total=$a * $b;
        return $total;
    }

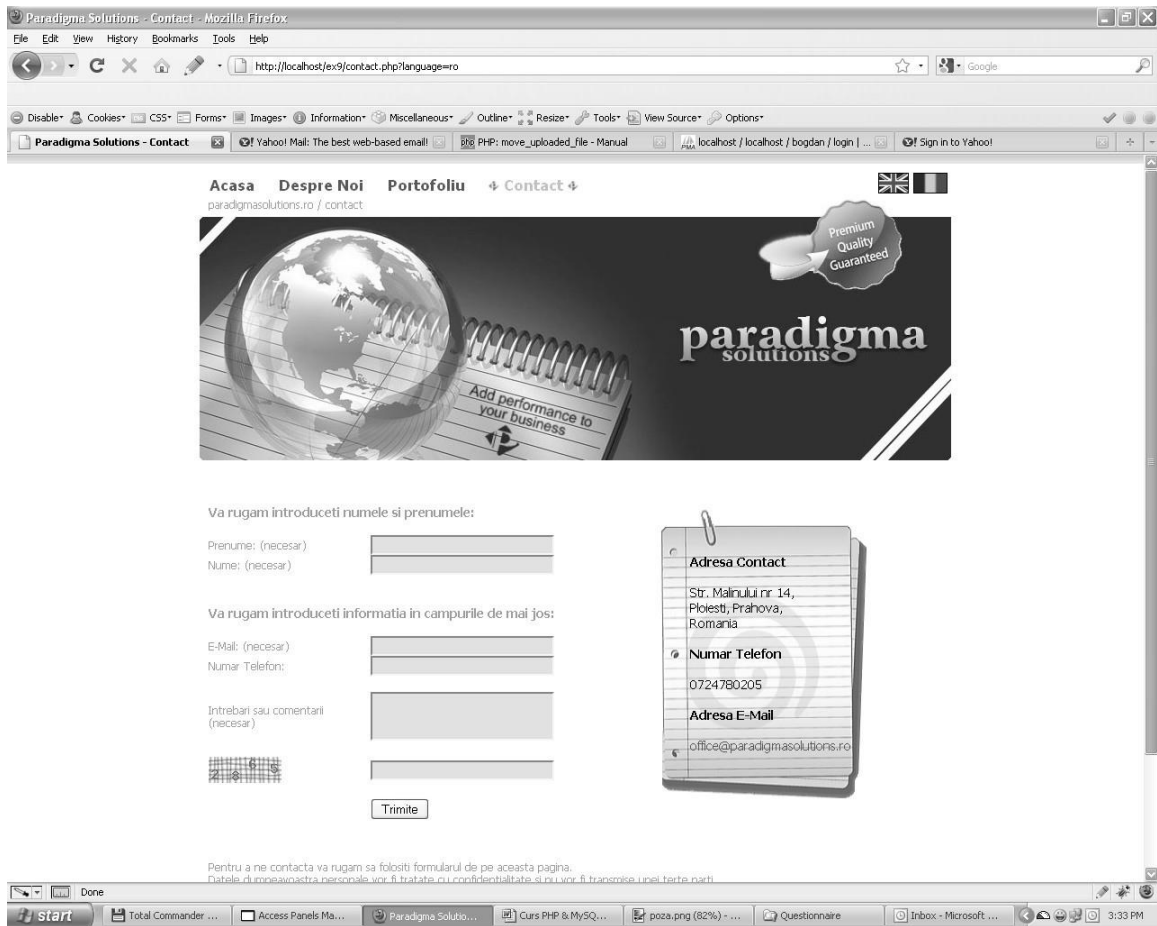
    echo $a." * ".$b." = ".inmultire($a,$b);

    function cub ($a){ $i=1;$total
        = 1;
        for($i=1;$i<=3;$i++){
            $total = $total * $a;
        }
        return $total;
    }

    echo $a."^3 = ". cub ($a);
    ?>

```

☺ **Exemplul 9** – Exemplu website dinamic de prezentare a unei firme de web design
 Paradigma Solutions (<http://www.paradigmasolutions.110mb.com/>)



layout.php

<?php

```
function layTop(){
global $_TITLE, $_OTHER, $_CURR, $_language, $text;

if(!isset($_TITLE))
    $_TITLE = "";
if(!isset($_OTHER))
    $_OTHER = "";
if(!isset($_CURR))
    $_CURR = "";
if(!isset($_language))
    $_language = "ro";

if(isset($_GET["language"]) && $_GET["language"]!=""){
    $_language=$_GET["language"];
}
}
```

```

if(isset($_GET["popup"]) && $_GET["popup"]!=""){
    $popup= 'no';
}else{
    $popup = 'yes';
}

?>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
<meta http-equiv="imagetoolbar" content="no" />
<title>Paradigma Solutions<?php echo $_TITLE != "" ? " " - ".dumps($_TITLE) : "" ;?></title>
<script type="text/javascript" src="<?php echo INCLUDE_PATH?>/js/funcs.js"></script>
<link rel="stylesheet" type="text/css" href="<?php echo INCLUDE_PATH?>/css/main.css" />
<?php echo dumps($_OTHER);?>
</head>

<body>
<div style="width:100%;background-image:url(<?php echo INCLUDE_PATH;?
>/pix/gradient.jpg);height:10px;"><div style="width:835px;;background-image:url(<?php echo
INCLUDE_PATH;?>/pix/gradient.jpg);height:10px;">&nbsp;   </div></div>
<table cellpadding="0" cellspacing="0" border="0" id="page" >
    <tr ><td style="height:25px;">
        <table cellpadding="0" cellspacing="0" border="0">
            <tr><td width="750"><ul class="menu">
                <li
style="display:inline;"><?php if($_CURR!='index'){ ?><a href="index.php?language=<?php echo
$_language;?>&popup=no" ><?php echo $text[$_language]['page_title']['index'];?></a><?php } else { ?
><span class="title" style="text-align:center;padding-top:10px;">&nbsp; <?php echo $text[$_language]['page_title']
[$_CURR];?>&nbsp; </span> <?
php } ?></li>
                <li
style="display:inline;"><?php if($_CURR!='about'){ ?><a href="about.php?language=<?php echo
$_language;?>" ><?php echo $text[$_language]['page_title']['about'];?></a><?php } else { ?><span
class="title" style="text-align:center;padding-top:10px;">&nbsp; <?php echo $text[$_language]['page_title'][$_CURR];?>&nbsp; </span> <?php } ?></li>
                <li
style="display:inline;"><?php if($_CURR!='portfolio'){ ?><a href="portfolio.php?language=<?php echo
$_language;?>" ><?php echo $text[$_language]['page_title']['portfolio'];?></a><?php } else { ?><span
class="title" style="text-align:center;padding-top:10px;">&nbsp; <?php echo $text[$_language]['page_title'][$_CURR];?>&nbsp; </span> <?php } ?></li>
                <li
style="display:inline;"><?php if($_CURR!='contact'){ ?><a href="contact.php?language=<?php echo
$_language;?>" ><?php echo $text[$_language]['page_title']['contact'];?></a><?php } else { ?><span
class="title" style="text-align:center;padding-top:10px;">&nbsp; <?php echo $text[$_language]['page_title'][$_CURR];?>&nbsp; </span> <?php } ?></li>
            </tr>
        </table>
    </td>
</tr>
</table>

```

```

        </ul>
    </td>
    <td width="85"><a href="<?php echo $_CURR;?>.php?
language=uk&popup=no"></a>
        <a href="<?php echo $_CURR;?>.php?
language=ro&popup=no"></a>
    </td>
</tr>
</table>
</td>
</tr>
<tr><td style="text-align:left;padding:2px 0px 0px 0px;">
    <div style="padding-left:10px;float:left;width:300px;"><a href="index.php?language=<?
php echo $_language;?>&popup=no" class="text_small"
onmouseover="this.style.textDecoration='underline'" onmouseout="this.style.textDecoration='none'"><?
php echo $text[$_language]['page_title']['website'];?></a><span class="text_small">&nbsp;&nbsp;&nbsp;<?php
echo $_CURR;?></span> </div>
    <div style="float:left;padding-left:373px;"><a href="index.php?language=<?php echo
$_language;?>&popup=no">" alt="Paradigma Solutions&trade;" border="0" /></a></div>
    </td>
</tr>
<tr><td style="vertical-align:top;padding-top:0px;">
    <a href="index.php?language=<?php echo $_language;?>&popup=no">" border="0" alt="Paradigma Solutions&trade;" /></a> </td>
</tr>
<tr><td id="content" valign="top" style="height:550px;">

<?php
}
function layBot(){ global
$_CURR,$_language,$text;
if(!isset($_CURR))
    $_CURR = "";
if(!isset($_language))
    $_language = "uk";
?>

    </td>
</tr>
</table>
<!--PUT HERE SO THAT IT RESIZES ACCORDING TO PAGE-->
<div >&nbsp;&nbsp;&nbsp;</div>
<!--/PUT HERE SO THAT IT RESIZES ACCORDING TO PAGE-->

<div style="background-image:url(<?php echo INCLUDE_PATH;?>/pix/gradient.jpg);width:100%;"
align="center">
    <div id="foot" style="background-image:url(<?php echo INCLUDE_PATH;?

```



```

>/pix/gradient.jpg);width:835px;padding-top:15px;"><?php echo $text[$_language]["foot"]["1"];?
>&nbsp;<a href="tc.php?language=<?php echo $_language;?>" target="_blank" style="font-weight:700;"
class="text_contact" onmouseover="this.style.color='#000000'"
onmouseout="this.style.color='#9A9A98'"><?php echo $text[$_language]["foot"]["2"];?></a>
        <ul class="sitemap">
            <li style="display:inline;border-right:1px solid #9A9A98;"><a
href="index.php?language=<?php echo $_language;?>&popup=no" ><?php echo $text[$_language]
['page_title']['index'];?></a></li>
            <li style="display:inline;border-right:1px solid #9A9A98;"><a
href="about.php?language=<?php echo $_language;?>" ><?php echo $text[$_language]['page_title']
['about'];?></a></li>
            <li style="display:inline;border-right:1px solid #9A9A98;"><a
href="portfolio.php?language=<?php echo $_language;?>" ><?php echo $text[$_language]['page_title']
['portfolio'];?></a></li>
            <li style="display:inline;"><a href="contact.php?language=<?php echo
$_language;?>" ><?php echo $text[$_language]['page_title']['contact'];?></a></li>
        </ul>
    </div>
</div>
<!-- GoStats JavaScript Based Code -->
<script type="text/javascript" src="http://gostats.ro/js/counter.js"></script>
<script type="text/javascript">_gos='c4.gostats.ro';_goa=343724;
_got=5;_goi=1;_goz=0;_gol='traffic web';_GoStatsRun();</script>
<noscript><a target="_blank" title="traffic web" href="http://gostats.ro"></a></noscript> <!-- End GoStats
JavaScript Based Code -->
</body>
</html>

<?php
}
?>
index.php

<?php

require_once ('./php/loader.php');

$_CURR = "index";
$_TITLE = "";
$_OTHER = "";

layTop();

?>
<script type="text/javascript" > var popup = '<?php echo $popup;?>';
if(typeof(enabled) !== 'undefined' && popup!='no') popImage();</script>

<div class="text" style="text-align:justify;padding:20px 10px 20px 10px;">
<div class="text"><?php echo str_replace(['*lang*'],$_language,$text[$_language]['home']['1']);?>

```

```

></div><br />
<div class="motto"><?php echo $text[$_language]['home']['2'];?></div><br />
<div class="text" style="text-align:justify;"><?php echo str_replace(['*lang*'],$_language,
$text[$_language]['home']['3']);?></div><br />
<div class="text"><?php echo $text[$_language]['home']['4'];?></div><br />
<div class="text"><?php echo $text[$_language]['home']['5'];?></div><br />
<div class="text"><?php echo $text[$_language]['home']['6'];?></div><br />
<div class="motto"><?php echo $text[$_language]['home']['7'];?></div><br />
<div style="text-align:center;"><a href="about.php?language=<?php echo $_language;?>" >" alt="Welcome" /></a></div> </div>
<?php layBot();

?>

```

Unitatea de învățare 5



Functii PHP de conectare la baza de date MySQL

V.1. Conectare la serverul MySQL și selectarea unei baze de date

V.2. Prelucrarea informației din baza de date

V.3. Închiderea conexiunii la serverul MySQL

V.1. Conectare la serverul MySQL și selectarea unei baze de date

mysql_connect (HOST, USER, PASSWORD) — deschide o conexiune la serverul MySQL aflat la adresa HOST pentru userul USER

mysql_select_db (DB, SERVER) – selectează o baza de date DB de pe serverul SERVER

Exemplu:

```
<?php
function connect2DB(){
    $DB_HOST = "localhost";
    $DB_USER = 'root';
    $DB_PASS = 'mysql';
    $DB_DBname = 'bogdan';

    $server = mysql_connect($DB_HOST,$DB_USER,$DB_PASS) or die(mysql_error());
    mysql_select_db($DB_DBname,$server) or die(mysql_error()); }
connect2DB();
?>
```

V.2. Prelucrarea informației din baza de date

mysql_query (QUERY) – execută query-ul curent (fie ca acesta este SELECT, INSERT, DELETE etc.)

Exemplu:

```
$insert = "INSERT INTO login VALUES('admin1','".sha1('parola1')."')";
mysql_query($insert) or die(mysql_error());
```

mysql_num_rows (EXECUTED QUERY) – returnează numărul de rânduri al rezultatului query-ului executat, dacă nu găsește nici un rezultat returnează 0

Exemplu:

```
$select = "SELECT password FROM login WHERE user='". $curr_user."'";
$sel = mysql_query($select) or die(mysql_error());

if(mysql_num_rows($sel)==1){
    .....
}
```

mysql_fetch_array (EXECUTED QUERY) – returneaza rezultatul unui query executat, sub forma unui array denumit **\$row** cu cheile numele campurilor tabelului

Exemplu:

```
$select = "SELECT password FROM login WHERE user='".mysql_escape_string($curr_user)."'";
$sel = mysql_query($select) or die(mysql_error());

while($row=mysql_fetch_array($sel)){
    $curr_pass_from_DB = $row['password'];
}
```

mysql_escape_string (VARIABILA) – escapeaza valoarea din VARIABILA curenta, aceasta functie este folosita impotriva SQL Injection

Exemplu:

```
$select = "SELECT password FROM login WHERE user='".mysql_escape_string($curr_user)."'";
$sel = mysql_query($select) or die(mysql_error());
```

mysql_error() – returneaza eroarea daca query-ul nu se executa corect

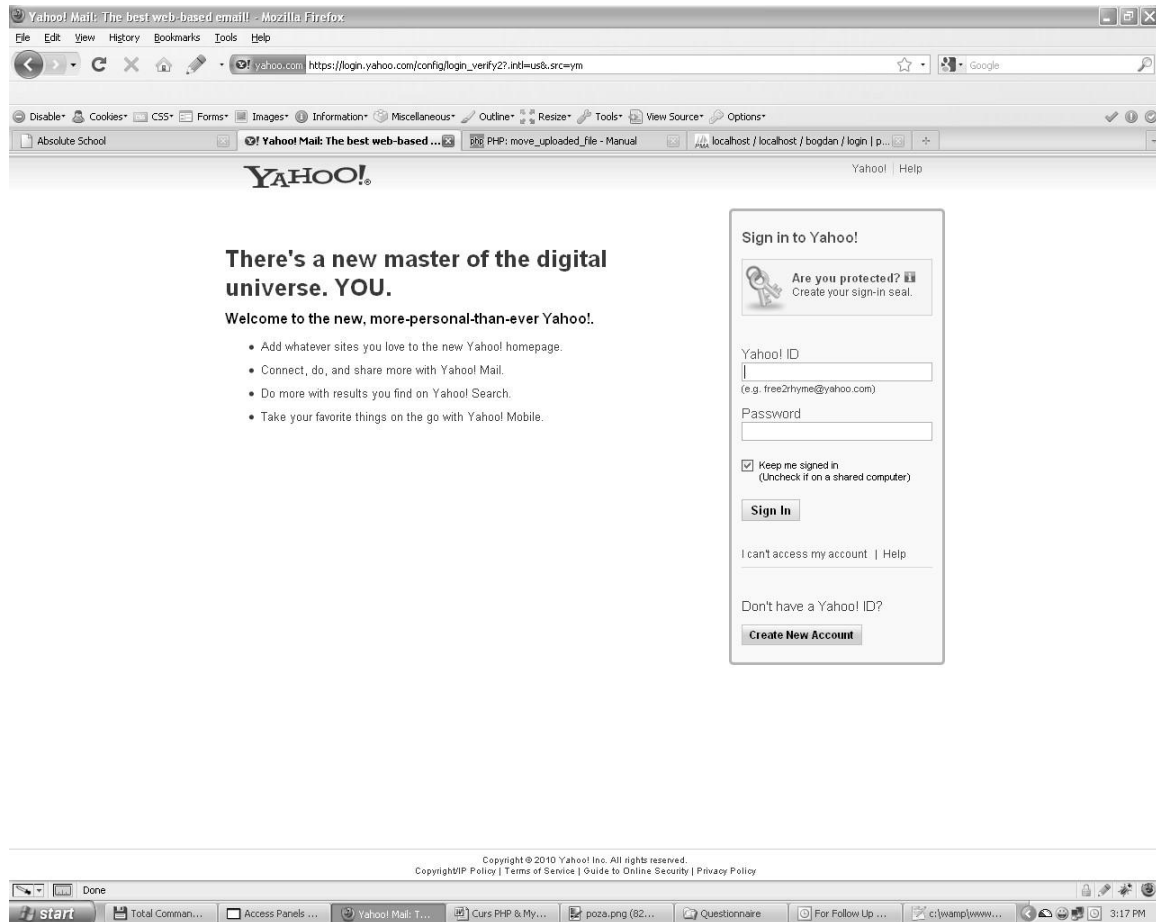
```
$select = "SELECT password FROM login WHERE user='". $curr_user."'";
$sel = mysql_query($select) or die(mysql_error());
```

Exemplu de comenzi combinate in cadrul unui script:

```
<?php
$select = "SELECT pass FROM login_RB WHERE user='".mysql_escape_string($user)."'";
$sel = mysql_query($select) or die(mysql_error());

if(mysql_num_rows($sel)==1){
    while($row=mysql_fetch_array($sel)){
        $password = $row['pass'];
    }
}
?>
```

☺ **Exemplul 10** – Să se creeze un modul de autentificare pentru un website care să conțină minim câmpurile user și parola. Parola sa fie criptata folosind algoritmul de criptare SHA1 si validata cu tuplul user/parola din tabela de login dintr-o baza de date. Acest tip de autentificare a utilizatorilor este folosit de majoritatea website-urilor de profil Yahoo!, Facebook etc. (ex: <https://login.yahoo.com>)



```
<?php
```

```
function connect2DB(){
    $DB_HOST = 'localhost';
    $DB_USER = 'root';
    $DB_PASS = '';
    $DB_DBname = 'bogdan';
```

```
$server = mysql_connect($DB_HOST,$DB_USER,$DB_PASS) or die(mysql_error());
mysql_select_db($DB_DBname,$server) or die(mysql_error());
}
connect2DB();
```

```
/*$drop_table = "DROP TABLE login";
mysql_query($drop_table) or die(mysql_error());
```


V.3. Închiderea conexiunii la serverul MySQL

mysql_close (CONEXIUNE) – închide conexiunea CONEXIUNE la serverul MySQL. Dacă nu se specifica nici un nume de conexiune atunci este închisa ultima conexiune deschisa.

Exemplu:

```
<?php
$DB_HOST = "localhost";
$DB_USER= 'root';
$DB_PASS='mysql';
$DB_DBname='bogdan';

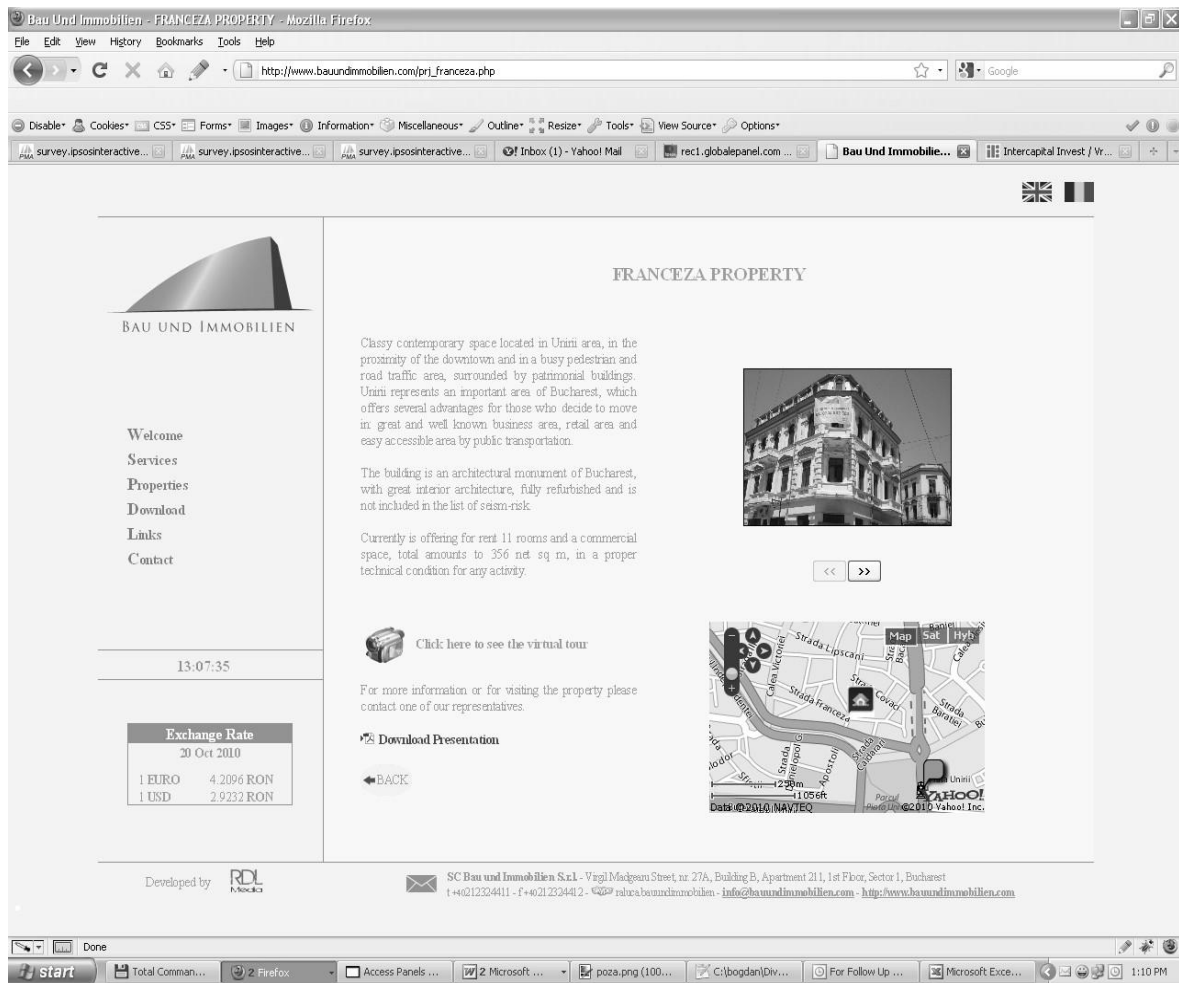
$server = mysql_connect($DB_HOST,$DB_USER,$DB_PASS)or die(mysql_error());
mysql_select_db( $DB_DBname, $server) or die(mysql_error);
.....
//operatii asupra tabelor din baza de date
.....
mysql_close();

?>
```



Lista completă de funcții PHP pentru lucrul cu o baza de date MySQL o puteți găsi accesând linkul de mai jos: <http://www.php.net/manual/en/ref.mysql.php>

☺ **Exemplul 11** – Website portal agenție imobiliară pentru firma Bau Und Immobilien <http://www.bauundimmobilien.com> ce are o pagină de proprietăți ce se modifică dinamic în funcție de conținutul din baza de date.



• create_DB.php

```
<?php
```

```
$db = mysql_connect("localhost","root","mysql") or die(mysql_error());
mysql_select_db("bogdan",$db) or die(mysql_error());
```

```
$drop = "DROP TABLE `bui_DB`";
mysql_query($drop) or die(mysql_error());
```

```
$create = " CREATE TABLE `bui_DB` ("
$create .= " `id` int(3) NOT NULL auto_increment,";
$create .= " `property_name` varchar(200) default NULL,";
$create .= " `propertyType_uk` varchar(200) default NULL,";
$create .= " `propertyType_ro` varchar(200) default NULL,";
$create .= " `propertyType_de` varchar(200) default NULL,";
$create .= " `short_description_uk` varchar(900) default NULL,";
$create .= " `short_description_ro` varchar(900) default NULL,";
$create .= " `short_description_de` varchar(900) default NULL,";
$create .= " `description_uk` varchar(2000) default NULL,";
$create .= " `description_ro` varchar(2000) default NULL,";
$create .= " `description_de` varchar(2000) default NULL,";
$create .= " `picture1` varchar(300) default NULL,";
```

```

$create := " `picture2` varchar(300) default NULL,";
$create := " `picture3` varchar(300) default NULL,";
$create := " `status_uk` varchar(300) default NULL,";
$create := " `status_ro` varchar(300) default NULL,";
$create := " `status_de` varchar(300) default NULL,";
$create := " `file_uk` varchar(300) default NULL,";
$create := " `file_ro` varchar(300) default NULL,";
$create := " `file_de` varchar(300) default NULL,";
$create := " `tur` int(1) default NULL,";
$create := " `tur_website` varchar(255) default NULL,";
$create := " `property_lat` varchar(25) default NULL,";
$create := " `property_long` varchar(25) default NULL,";
$create := " `property_title_uk` varchar(255) default NULL,";
$create := " `property_title_ro` varchar(255) default NULL,";
$create := " `property_title_de` varchar(255) default NULL,";
$create := " `site_lat` varchar(25) default NULL,";
$create := " `site_long` varchar(25) default NULL,";
$create := " `site_title_uk` varchar(255) default NULL,";
$create := " `site_title_ro` varchar(255) default NULL,";
$create := " `site_title_de` varchar(255) default NULL,";
$create := " `datain` date default NULL,";
$create := " `timein` time default NULL,";
$create := " PRIMARY KEY (`id`)";
$create := " );"; mysql_query($create) or

```

```
die(mysql_error());
```

```

$insert = "INSERT INTO `bui_DB` VALUES (1, 'preotescu', 'PREOTESCU PROJECT', 'PROIECT
PREOTESCU', 'PREOTESCU PROJECT',

```

'Under construction office units located in Unirii area, in the proximity of the downtown and in a busy pedestrian and road traffic area, surrounded by patrimonial buildings. The future office building will have 2 basements for 8 underground parking spaces and 6 floors with total office spaces of 656.55 sq m.',

'Proiect de imobil pentru birouri situat in zona Unirii, in imediata apropiere de centrul orasului, intr-un areal binecunoscut, atat pietonal cat si al arterelor importante, fiind inconjurat de cladiri de patrimoniu si spatii verzi. Viitorul imobil de birouri proiectat va avea 2 subsoluri, unde se pot parca 8 masini, plus 6 niveluri pentru spatii comerciale si de birouri (2S+P+4+1R), totalizand o suprafata utila de 656.55 mp',

'Under construction office units located in Unirii area, in the proximity of the downtown and in a busy pedestrian and road traffic area, surrounded by patrimonial buildings. The future office building will have 2 basements for 8 underground parking spaces and 6 floors with total office spaces of 656.55 sq m.',

'Under construction office units located in Unirii area, in the proximity of the downtown and in a busy pedestrian and road traffic area, surrounded by patrimonial buildings. Unirii represents an important area of Bucharest, which offers several advantages for those who decide to move in: great and well known business area, retail area and easy accessible area by public transportation.

 The existing property is a ground floor villa with 160 constructed sq m. The villa was erected in 1920 and is divided such as:

- 4 rooms (between 10-21 sq m)

- kitchen (10 sq m)

- bathroom (4 sq m)

- a spacious garage (30 sq m)

- storage (13 sq m)

- interior yard (78 sq m)

Needs total renovation work.'

'Proiect de imobil pentru birouri situat in zona Unirii, in imediata apropiere de centrul orasului, intr-un areal binecunoscut, atat pietonal cat si al arterelor importante, fiind inconjurat de cladiri de patrimoniu si spatii verzi. Zona Unirii reprezinta o zona extrem de importanta in Bucuresti, care ofera avantaje clare celor care decid sa se stabileasca aici: zona de afaceri, comerciala si usor accesibila, atat prin mijloacele de transport in comun cat si pentru cei care folosesc autoturismele proprii.

In prezent exista o constructie rezidentiala desfasurata pe parter, avand o suprafata de 160 mp. Imobilul, construit in 1920, este compus din:

- 4 camere (cu suprafete intre 10-21 mp)

- Bucatarie (10 mp)

- Baie (4 mp)

- Constructie anexa (13 mp)

- Garaj spatios (30 mp)

- Curte (cu suprafata de 78 mp)

Necesita renovare totala daca se doreste mentinerea acesteia.'

'Under construction office units located in Unirii area, in the proximity of the downtown and in a busy pedestrian and road traffic area, surrounded by patrimonial buildings. Unirii represents an important area of Bucharest, which offers several advantages for those who decide to move in: great and well known business area, retail area and easy accessible area by public transportation.

The existing property is a ground floor villa with 160 constructed sq m. The villa was erected in 1920 and is divided such as:

- 4 rooms (between 10-21 sq m)

- kitchen (10 sq m)

- bathroom (4 sq m)

- a spacious garage (30 sq m)

- storage (13 sq m)

- interior yard (78 sq m)

Needs total renovation work.'

'1_Preotescu_1.jpg', '1_Preotescu_2.jpg', '1_Preotescu_3.jpg', 'Work in progress', 'In curs de proiectare', 'Work in progress', '1_Corespondenta Preotescu_EN.pdf', '1_Corespondenta Preotescu_RO.pdf', '1_Corespondenta Preotescu_EN.pdf', 1, 'http://www.rdlmedia.ro/projects/bauundimmobilien-preotescu/', '44.42195001359743', '26.09604835510254', 'Office Building', 'Cladire Birouri', 'Office Building', '44.41876229178734', '26.095168590545654', 'Libertatii Square', 'Piata Libertatii', 'Libertatii Square', '2009-05-11', '07:48:57');

```
mysql_query($insert) or die(mysql_error());
```

```
$insert = "INSERT INTO `bui_DB` VALUES (2, 'smardan', 'SMARDAN RESIDENTIAL APARTMENT', 'APARTMENT SMARDAN', 'SMARDAN RESIDENTIAL APARTMENT', 'Located in the middle of the great historical area of Bucharest, in a representative patrimonial building, the premise offers spacious 3 rooms accommodation and featuring modern amenities. Those who love to live in the heart of the city, in a pedestrian, retail and easy accessible area by public transportation area can be sure that this is the perfect choice ';
```

'Situat in mijlocul zonei istorice a Bucurestiului, intr-un imobil reprezentativ al patrimoniului romanesc, apartamentul ofera 3 camere, baie, bucatarie cu dotari si finisaje de ultima ora. Cei care doresc sa locuiasca in inima orasului, in zona pietonala, comerciala si usor accesibila din punctul de vedere al mijloacelor de transport pot fi siguri ca aceasta este alegerea perfecta',

'Located in the middle of the great historical area of Bucharest, in a representative patrimonial building, the premise offers spacious 3 rooms accommodation and featuring modern amenities. Those who love to live in the heart of the city, in a pedestrian, retail and easy accessible area by public transportation area can be sure that this is the perfect choice',

'Located in the middle of the great historical area of Bucharest, in a representative patrimonial building, the premise offers spacious 3 rooms accommodation and featuring modern amenities.

The apartment offers furnished and equipped kitchen, an amazing bathroom of 13 sq m (with complete shower cabin and washing machine), Air Conditioning system (both for cold and hot air), surrounding audio system and double glass wooden windows.

 In the total of 76 sq m, fully refurbished in 2009, we can find also a living and dining area of 21 sq m; a studio of 11 sq m and a light bedroom of 18 sq m, with a stylish balcony.

Those who love to live in the heart of the city, in a pedestrian, retail and easy accessible area by public transportation area can be sure that this is the perfect choice.'

'Situat in mijlocul zonei istorice a Bucurestiului, intr-un imobil reprezentativ al patrimoniului romanesc, apartamentul ofera 3 camere, baie, bucatarie cu dotari si finsaje de ultima ora.

In interiorul apartamentului gasim o bucatarie mobilata si complet utilata, o baie ampla de 13 mp (cu o cabina de dus cu hidromasaj si aburi, dar si cu masina de spalat rufe), aer conditionat bitermic, sistem audio in retea, geamuri termopan cu cadre din lemn si usa metalica securizata.

In afara de cele doua incaperi mai sus mentionate, in cei 76 mp utili ai apartamentului gasim o sufragerie de 21 mp, o camera de studiu de 11 mp si un dormitor luminos de 18 mp, plus o logie.

 Cei care doresc sa locuiasca in inima orasului, in zona pietonala, comerciala si usor accesibila din punctul de vedere al mijloacelor de transport pot fi siguri ca aceasta este alegerea perfecta.'

'Located in the middle of the great historical area of Bucharest, in a representative patrimonial building, the premise offers spacious 3 rooms accommodation and featuring modern amenities.

The apartment offers furnished and equipped kitchen, an amazing bathroom of 13 sq m (with complete shower cabin and washing machine), Air Conditioning system (both for cold and hot air), surrounding audio system and double glass wooden windows.

 In the total of 76 sq m, fully refurbished in 2009, we can find also a living and dining area of 21 sq m; a studio of 11 sq m and a light bedroom of 18 sq m, with a stylish balcony.

Those who love to live in the heart of the city, in a pedestrian, retail and easy accessible area by public transportation area can be sure that this is the perfect choice.', '2_Smardan_1.jpg', '2_Smardan_2.jpg', '2_Smardan_3.jpg', 'For Rent', 'De Inchiriat', 'For Rent', '2_Corespondenta Smardan_EN.pdf', '2_Corespondenta Smardan_RO.pdf', '2_Corespondenta Smardan_EN.pdf', 0,',44.431573426835755',26.099846363067627',Residential Apartment For Rent',Apartament De Inchiriat',Residential Apartment For Rent',44.435434605825975',26.10274314880371',Universitate Square',Piata Universitatii',Universitate Square',2009-05-11', '07:49:04')";

```
mysql_query($insert) or die(mysql_error());
```

```
echo "Tables created";
```

```
?>
```

• template.php

```
<?php
```

```
require_once ('./php/loader.php');
```

```
//getting data from DB
```

```
connect2DB();
```

```
$id = substr(basename($_SERVER['PHP_SELF']),4,strlen(basename($_SERVER['PHP_SELF']))-8);
```

```
$property_name = array();
```

```
$propertyType = array();
```

```

$description = array();
$picture = array();

$select = "SELECT * FROM bui_DB WHERE property_name='".$Sid."'";
$sel = mysql_query($select) or die(mysql_error());
while($row=mysql_fetch_array($sel)){
    $property_name=stripslashes($row[1]);
    $propertyType['uk']= stripslashes($row[2]);
    $propertyType['ro'] = stripslashes($row[3]);
    $propertyType['de'] = stripslashes($row[4]);
    $description['uk'] = stripslashes($row[8]);
    $description['ro'] = stripslashes($row[9]);
    $description['de'] = stripslashes($row[10]);
    $picture[1] = stripslashes($row[11]);
    $picture[2] = stripslashes($row[12]);
    $picture[3] = stripslashes($row[13]);
    $status['uk'] = stripslashes($row[14]);
    $status['ro'] = stripslashes($row[15]);
    $status['de'] = stripslashes($row[16]);
    $file_uk = stripslashes($row[17]);
    $file_ro = stripslashes($row[18]);
    $file_de = stripslashes($row[19]);
    $tur = stripslashes($row[20]);
    $tur_website = stripslashes($row[21]);
    $property_lat = stripslashes($row[22]);
    $property_long = stripslashes($row[23]);
    $property_title['uk'] = stripslashes($row[24]);
    $property_title['ro'] = stripslashes($row[25]);
    $property_title['de'] = stripslashes($row[26]);
    $site_lat = stripslashes($row[27]);
    $site_long = stripslashes($row[28]);
    $site_title['uk'] = stripslashes($row[29]);
    $site_title['ro'] = stripslashes($row[30]);
    $site_title['de'] = stripslashes($row[31]);

}
$_CURR = "prj_".$property_name;
$_TITLE = $propertyType['uk'];

$text['uk']['page_title'][$_CURR] = $propertyType['uk'];
$text['ro']['page_title'][$_CURR] = $propertyType['ro'];
$text['de']['page_title'][$_CURR] = $propertyType['de'];

$_OTHER = "";

layTop();

$showPicture = 'no'; for($i=1;$i<=3;$i++){
if(substr($picture[$i],strlen($picture[$i])-10,10)!='_noPicture'){
    $showPicture =
        'yes'; break; }
}

```



```

[ 'properties' ][ 'tour' ];?></a></td></tr></table><?php } ?><br><?php echo $text[ $_language ][ 'properties' ]
[ 'info' ];?><br><br><?php if ( $showFile == "yes" ) { ?><a href = ". /upload /<?php echo $curFile ?>"
target = "_ blank" style = "text-decoration: none; color: #ff0000; font-weight: 700; "><img src = ". /pix /pdf. gif"
border = "0" ><?php echo $text[ $_language ][ 'template' ][ 'download' ];?></a><?php } else { echo "<br />"; } ?
><br /><br /><table cellpadding = "0" cellspacing = "0" border = "0">
<tr><td style = "background-image: url ('. /pix /BackButton. jpg'); background-
repeat: no repeat; width: 56px; height: 33px; padding-left: 18px; vertical-align: middle; "><span
onClick = "goTo ( 'properties. php' )" style = "text-decoration: none; color: #999999; "
onMouseOver = "this. style. color = '#000000'; document. body. style. cursor = 'pointer'; "
onMouseOut = "this. style. color = '#999999'; document. body. style. cursor = 'default'; "><?php echo
$text[ $_language ][ 'template' ][ 'back' ];?></span></td><td>&nbsp;</td></tr>
</table></td><td width = "80">&nbsp;</td>
<td width = "300" style = "text-align: center; ">
<div id = "map" width = "300" height = "200" onClick = "window. open ( 'map. php?property = <?php echo
$property_name; ?> & language = <?php echo $_language; ?
>', 'mywindow', 'width = 500, height = 500, resizable = yes, scrollbars = yes, toolbar = no' )" >&nbsp;</div><br>
<i style = "color: #999999; font-size: 12px; font-family: Arial; "><?php echo $text[ $_language ][ 'template' ]
[ 'map' ];?></i><br>
</td></tr></table> <br>

<?php layBot(); ?>

```

BIBLIOGRAFIE

- **Rețele de calculatoare**

- 1) **ANDREW TANENBAUM** *"Rețele de calculatoare"*

- **PHP**

- 2) <http://www.w3schools.com/php/default.asp>
- 3) <http://www.php.net/manual>
- 4) **SKLAR DAVID** **PHP Cookbook**
- 5) **SKLAR DAVID** **Learning PHP 5**

- **MYSQL**

- 6) <http://www.mysql.com/>
- 7) **DAVID LANE** **Web Database Application With PHP And MySQL**

ANEXĂ



☺ **Aplicație practică de tipul unui site dinamic de anunțuri, care cuprinde:**

- proiectarea structurii site-ului și a bazei de date;
- scripturile necesare procedurilor de adăugare și afișare a datelor;
- gruparea înregistrărilor pe categorii;
- contor de pagină;
- sondaj de opinie cu generare automată a rezultatelor;
- motor de căutare intern.

◦ **Structura website:**

I. **Structura home page / afișare anunturi**

| | | | |
|------|----------------|--------------|------------------|
| Home | Adaugare Anunt | Cutare Anunt | Sondaj de opinie |
|------|----------------|--------------|------------------|

| Anunt | Contact | Categorie  | Tip  | Opinie |
|---------|----------|---|---|--------|
| Anunt 1 | 072..... | Masini | Vanzare | ★★★★☆ |
| Anunt 2 | 074..... | IT&C | Cumparare | ★★★★★ |
| | | | | |
| | | | | |

| |
|----------------|
| Footer Section |
|----------------|

II. Structura pagină adăugare anunț

| | | | |
|------|----------------|---------------|------------------|
| Home | Adaugare Anunt | Cautare Anunt | Sondaj de opinie |
|------|----------------|---------------|------------------|

| | |
|---|--|
| Anunt | <input type="text"/> |
| Date de contact | <input type="text"/> |
| Categorie | <input type="text" value="Categorie"/> ▼ |
| Tip | <input type="text" value="Tip"/> ▼ |
| <input type="button" value="Adauga anunt"/> | |

Footer Section

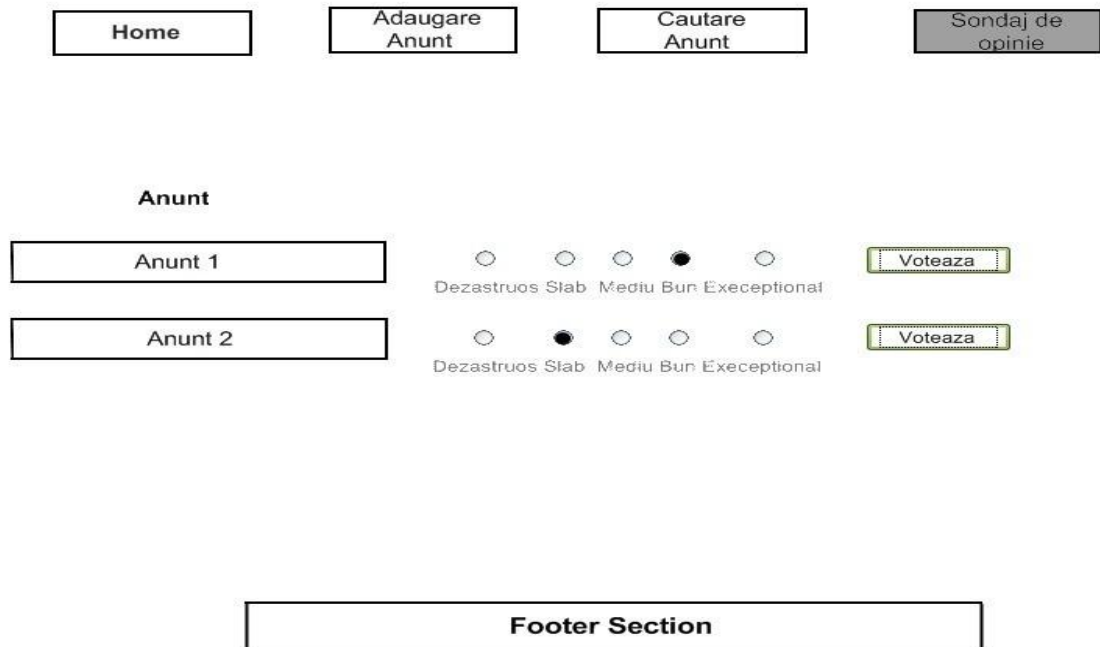
III. Structura pagină căutare anunț

| | | | |
|------|----------------|---------------|------------------|
| Home | Adaugare Anunt | Cautare Anunt | Sondaj de opinie |
|------|----------------|---------------|------------------|

| | | | |
|---------|----------------------|--|------------------------------------|
| Cautati | <input type="text"/> | <input type="text" value="Categorie"/> ▼ | <input type="text" value="Tip"/> ▼ |
|---------|----------------------|--|------------------------------------|

Footer Section

IV. Structura pagină sondaj opinie



◦ Structura sistem de fișiere

website

- **/css/** main.css
style.css
- **/js/** funcs.js
jquery-1.4.min.js
- **/php/** funcs.php
- **/images/**
image1.jpg
image2.gif
- index.php
- adaugare.php
- cautare.php
- sondaj.php

- **Structură bază de date**

TABELA_ANUNTURI

| Id | Anunt | Contact | Tip | Categorie | Data |
|-----------|--------------|----------------|-------------|------------------|-------------|
| md5 | Varchar(255) | INT(10) | Varchar(25) | Varchar(25) | Date |

TABELA_LOGIN

| User | Parola |
|--------------|---------------|
| Varchar(255) | SHA1 |

- **Contor vizitatori website:**

Încărcat pe home/landing page

<http://gostats.ro/>

<http://www.google.com/analytics/>