

Curs 9-10

2016/2017

# Tehnici moderne de proiectare a aplicatiilor web

# CURS

I.	HTML si XHTML (recapitulare)	1 oră
II	CSS	2 ore
III	Baze de date, punct de vedere practic	1 oră
IV	Limbajul de interogare SQL	4 ore
V	PHP - HyperText Preprocessor	8 ore
VI	XML - Extended Mark-up Language si aplicatii	4 ore
VII	Conlucrare intre PHP/MySql, PHP/XML, Javascript/HTML	2 ore
VIII	Exemple de aplicatii	6 ore
	Total	28 ore

MySql

# Laborator 7

# Rezultat (cumparator)

**Magazin Firma**

[Inceput](#) | [Inapoi](#)

## Magazin online Firma X SRL

Alegeti:

- [Cumparator](#)
- [Vanzator](#)

## Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	<a href="#">Papetarie</a>	3
2	<a href="#">Instrumente</a>	3
3	<a href="#">Audio-video</a>	3
4	<a href="#">Calculatoare</a>	3
5	<a href="#">Jucarii</a>	2

Total produse: 14

## Magazin online Firma X SRL

### Realizati comanda

Nr.	Produs	Pret	Cantitate
1	Carti	100	<input type="text" value="1"/>
2	Caiete	50	<input type="text" value="2"/>
3	Penare	150	<input type="text" value="1"/>
4	Stilouri	125	<input type="text" value="0"/>
5	Creioane	25	<input type="text" value="0"/>

## Magazin online Firma X SRL

### Rezultate comanda

Pret total (fara TVA): 350

Pret total (cu TVA): 416.5

Comanda receptionata la data: 17/03/2010 ora 08:24

 post  
 get



# Rezultat (vanzator)

**Magazin** Firma X

[Inceput](#) | [Inapoi](#)

## Magazin online Firma X SRL

Alegeti:

- [Cumparator](#)
- [Vanzator](#)

### Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	<a href="#">Papetarie</a>	3
2	<a href="#">Instrumente</a>	3
3	<a href="#">Audio-video</a>	3
4	<a href="#">Calculatoare</a>	3
5	<a href="#">Jucarii</a>	2

Total produse: 14

Categorie noua de produse:

### Lista produse in categoria Calculatoare

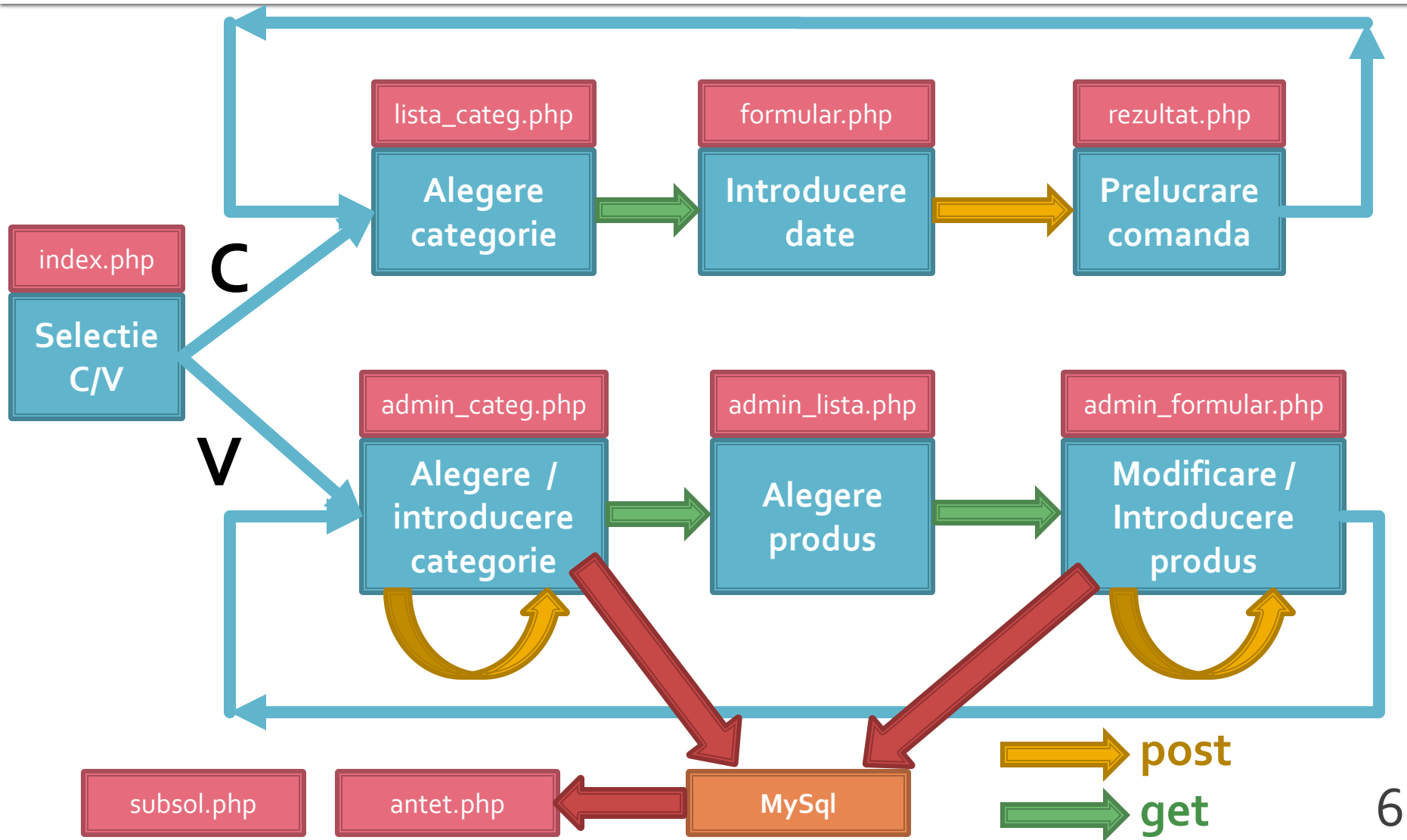
Nr.	Produs	Descriere	Pret	Cantitate	Actiuni
1	Laptop	calculator mic	2000	2	<a href="#">modifica</a>
2	Desktop	calculator mare	1000	5	<a href="#">modifica</a>
3	Imprimanta	prn	200	2	<a href="#">modifica</a>
-	Produs nou				<a href="#">adauga</a>

### Produs in categoria Calculatoare

Produs	<input type="text" value="laptop"/>
Descriere	<input type="text" value="calculator mic"/>
Pret	<input type="text" value="2000"/>
Cantitate	<input type="text" value="2"/>



# Plan aplicatie



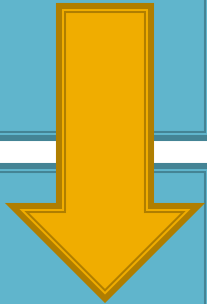
# Plan aplicatie – vanzator

- Deoarece citirea datelor se face in fisierul antet.php (modificat anterior) vor aparea modificari doar la nivelul scrierii datelor noi introduse
- Fisiere
  - admin\_lista.php – nemodificat
  - admin\_categ.php – scrie categorii noi in baza de date: se incuieste cod XML cu cod MySql
  - admin\_formular.php – scrie produse noi / corectii in baza de date: se incuieste cod XML cu cod MySql

# admin\_categ.php

```
if (isset($_POST["c_nou"]))
    //categorie noua introdusa
    $categ_nou=$xml->addChild("categorii");
    $categ_nou->addAttribute("nume", $_POST["nou"]);
    $xml->asXML("lista.xml"); // salvare fisier
    $produse[$_POST["nou"]]=array(); // update matrice produse
    echo "<p>Categoria ".$_POST["nou"]." adaugata!</p>";
}
```

```
if (isset($_POST["c_nou"]))
    //categorii noua introdusa
    $query = "INSERT INTO `categorii` (`nume`, `detalii`)VALUES ('
".$_POST["nou_nume"]."`, '".$_POST["nou_desc"]."`)";
    echo $query; //util in perioada de testare
    $result = mysql_query($query, $conex) or die(mysql_error());
    $record=mysql_insert_id(); //obtinerea id-ului nou
    $produse[$_POST["nou_nume"]]=array(); // update matrice produse
    echo "<p>Categoria ".$_POST["nou_nume"]." adaugata! Are id = ".$record."</p>";
}
```



# admin\_categ.php

Magazin Firma X SRL

[Inceput](#) | [Inapoi](#)

## Magazin online Firma X SRL

### Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	<a href="#">Papetarie</a>	3
2	<a href="#">Instrumente</a>	3
3	<a href="#">Audio-video</a>	3

Total produse: 9

Categorie noua de produse:

Nume:

Descriere:

Magazin Firma X SRL

[Inceput](#) | [Inapoi](#)

## Magazin online Firma X SRL

```
INSERT INTO `categorii` (`nume`, `detalii`) VALUES ('jucarii', 'pentru copii')
```

Categoria jucarii adaugata! Are id = 4

### Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	<a href="#">Papetarie</a>	3
2	<a href="#">Instrumente</a>	3
3	<a href="#">Audio-video</a>	3
4	<a href="#">Jucarii</a>	0

Total produse: 9

Categorie noua de produse:

Nume:

Descriere:

## Magazin online Firma X SRL

```
INSERT INTO `categorii` (`nume`, `detalii`) VALUES ('jucarii', 'pentru copii')
```

Categoria jucarii adaugata! Are id = 4

# admin\_formular.php

- Pentru inlocuire/adaugare produs apare o tratare diferita a celor doua situatii:
  - Adaugarea de produs face apel la interogarea SQL `INSERT INTO `produse` ...`
  - Modificarea unui produs existent va face apel la interogarea SQL `UPDATE `produse` SET ...`

# admin\_formular.php

```
if (isset($_POST["prod_ant"]))//exista deja acest produs anterior?
    //exista deja acest produs UPDATE
    unset($produse[$_POST['categ']][$_POST['prod_ant']]);//trebuie sters produsul anterior inlocuit
    $query = "UPDATE `produse` SET `nume`='".$_POST["prod"]."', `detalii`='".$_POST["descriere"]."',
`cant`='".$_POST["cantitate"]."', `pret`='".$_POST["pret"]."' WHERE `nume`='".$_POST["prod_ant"].'";
    echo $query;//util in perioada de testare
    $result = mysql_query($query, $conex) or die(mysql_error());
    echo "<p>Produsul '".$_POST["prod"]."' modificat in categoria '".$_POST['categ'].'";
}
else
    //NU exista acest produs INSERT
    $query = "INSERT INTO `produse` (`nume`, `detalii`, `pret`, `cant`, `id_categ`) VALUES
('".$_POST["prod"]."', '".$_POST["descriere"]."', '".$_POST["pret"]."', '".$_POST["cantitate"]."',
(SELECT `id_categ` FROM categorii WHERE `nume` = '".$_POST['categ'].')";
    echo $query;//util in perioada de testare
    $result = mysql_query($query, $conex) or die(mysql_error());
    $record=mysql_insert_id();//obtinerea id-ului nou
    echo "<p>Produsul '".$_POST["prod"]."' adaugat in categoria '".$_POST['categ'].'"; Are id =
".$_record."</p>";
}
$produse[$_POST['categ']][$_POST['prod']] = array("descr" => $_POST['descriere'], "pret" => $_POST['pret'], "cant" =>
$_POST['cantitate']);
```

# Final laborator

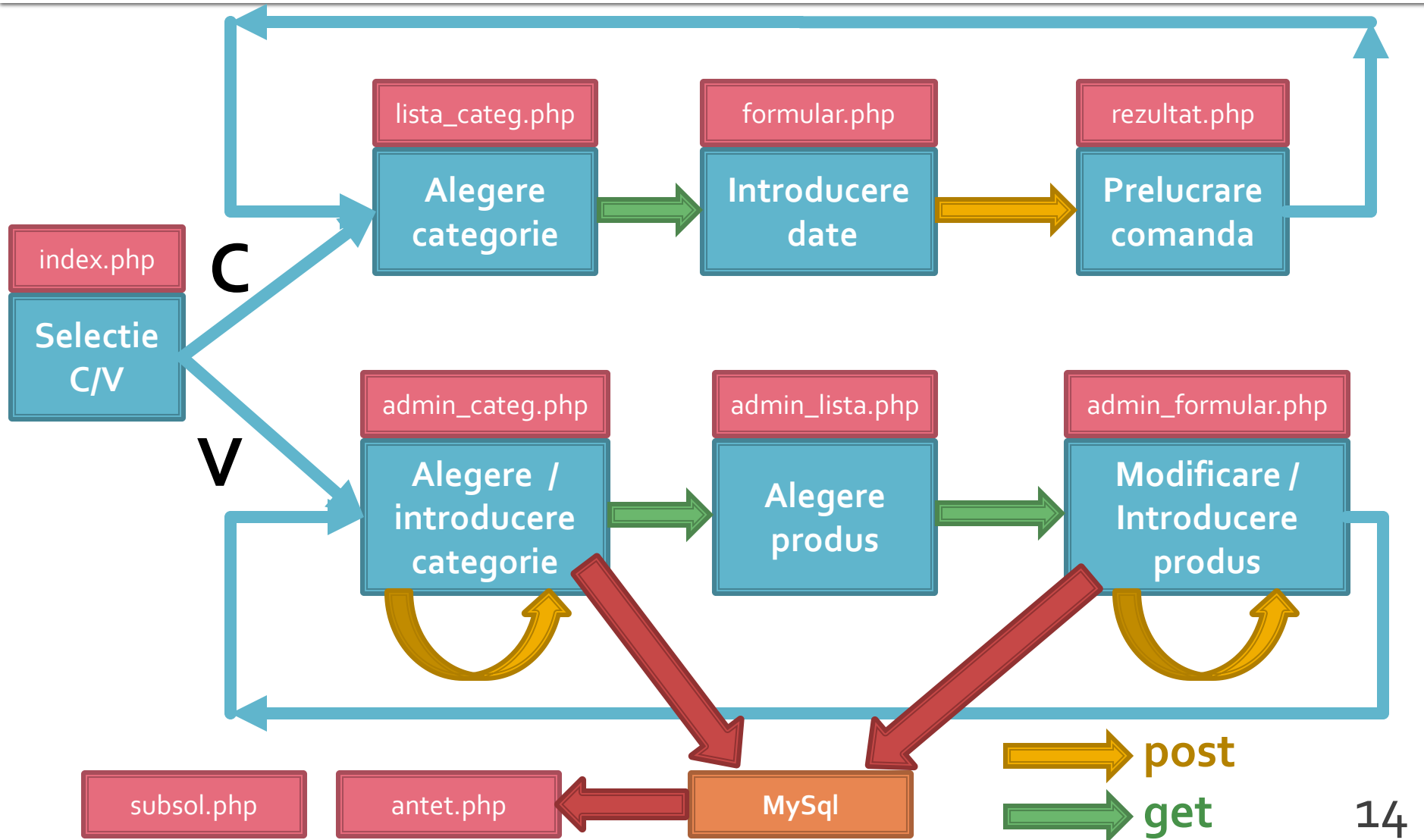
- Sursele complete ale aplicatiei pot fi obtinute de pe site-ul laboratorului
- Utilizarea MySql in aplicatii asa cum a fost facuta in acest exemplu **nu este optima**
  - Se incarca initial intreaga baza de date intr-o matrice de produse (antet.php)
  - Aceasta metoda **nu este** eficienta:
    - Server-ul MySql este o aplicatie compilata nativa sistemului de operare pe care ruleaza, in timp ce PHP este un limbaj interpretat
    - Se incarca inutil toate datele chiar si atunci cand nu este necesar (de exemplu cand afisez doar produsele dintr-o categorie sau cand afisez pentru a fi modificate doar detaliile unui produs)



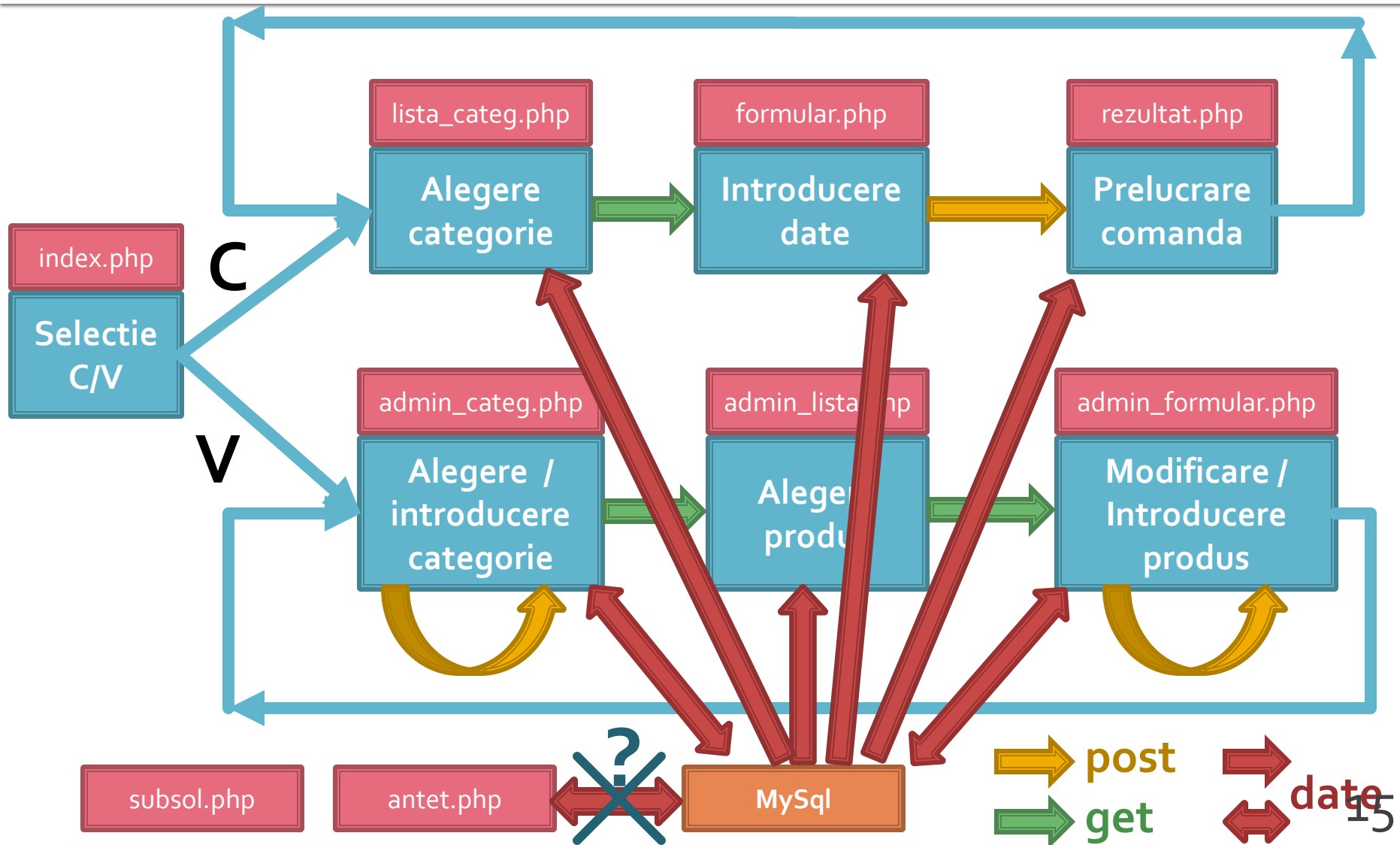
# Final laborator

- Varianta corecta presupune:
  - Citirea datelor in fiecare fisier in parte
  - Selectia datelor necesare pe server-ul MySql (mult mai eficient decat PHP)
  - De multe ori e mai eficienta utilizarea resursei rezultate din interogarea SQL decat crearea unei variabile matriciale suplimentare
    - `$result = mysql_query($query, $conex);`  
`$row_result = mysql_fetch_assoc($result);`  
`..... $row_result['nume'] .....`

# Plan aplicatie - laborator



# Plan aplicatie - optim



# Activitate suplimentara

# Activitate suplimentara

- Exemplul prezentat in sursele de pe site (laborator) este inefficient
- Suplimentar ascunde o **greseala de logica** care impiedica functionarea corecta a programului
  - programul nu este protejat, nu verifica faptul ca in casuta in care se asteapta numere nu se introduc siruri de text
  - **greseala de logica** presupune utilizatorul **cooperant si educat**, introduce ceea ce se asteapta de la el sa introduca, dar chiar in aceste conditii apare o abatere de la functionarea corecta

# Recompensa activitate suplimentara

- Raspunsul corect va fi recompensat cu:
  - **2p** in plus la nota de laborator (se pot compensa astfel eventuale absente)
  - **2p** in plus la nota de la testarea finala (examen)
- Nota de la proiect
  - Nu este influentata
- Nota finala se obtine prin medie ponderata **dupa** aplicarea suplimentelor amintite mai sus

**Nu se aplica din 2015/2016**

# Regulament recompensa

- Raspunsul si codul de corectie trebuie trimise individual prin email
- Codul trebuie sa fie functional
- Maxim **2** incercari pentru fiecare student
- Studentii pot discuta intre ei **dar**
- Oricare **doua raspunsuri identice se elimina reciproc**

**Nu se aplica din 2015/2016**

# Aspecte practice recomandate in realizarea aplicatiilor web



# Metode de lucru recomandate 1

- Daca nu aveti acces simplu la "log-urile" server-ului MySql puteti vedea cum ajung efectiv interogariile la el afisand temporar textul interogarii
  - `$query = "SELECT * FROM `produse` AS p WHERE `id_categ` = ".$row_result_c['id_categ'];`  
`echo $query;` //util in perioada de testare
    - Textul prelucrat de PHP al interogarii va fi afisat in clar pe pagina facand mai usoara depanarea programului
    - Aceste linii **trebuie** eliminate in forma finala a programului ca masura de securitate

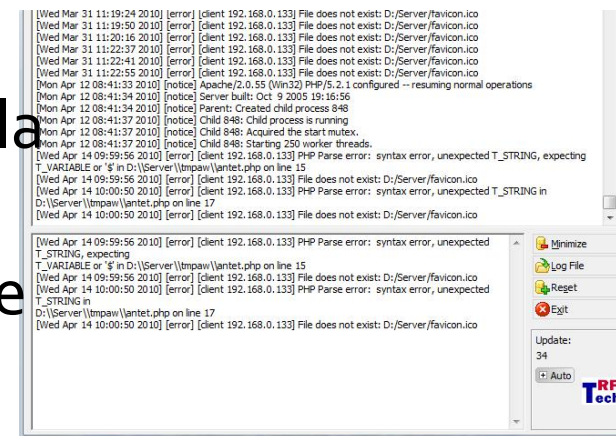
# Metode de lucru recomandate 2

- Verificarea “log-ului” de erori al server-ului Apache ramane principala metoda de depanare a codului PHP.

- W2000: Utilizarea aplicatiei prezentata la laborator este mai comoda datorita automatizarii dar orice alta varianta este utila

- Centos 7.1:

- putty → nano /var/log/httpd/error\_log
- <http://192.168.30.5/logfile.php> (nonstandard)

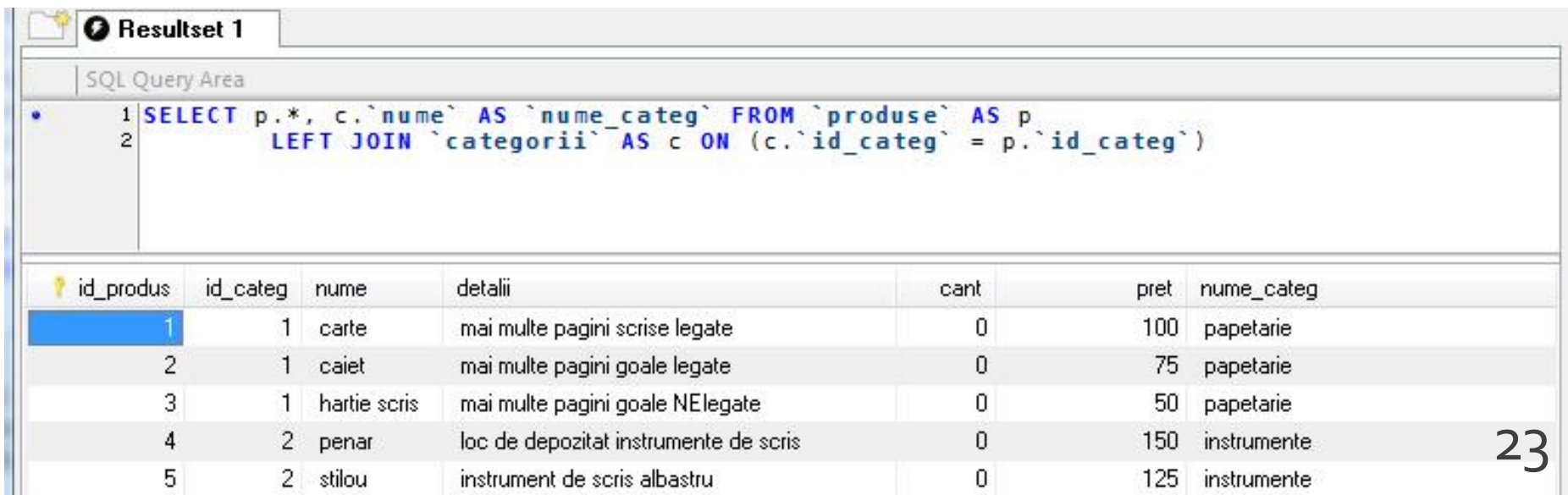


The screenshot shows a log viewer window with a list of error messages. The messages include:

- [Wed Mar 31 11:19:24 2010] [error] [client 192.168.0.133] File does not exist: D:/Server/favicon.ico
- [Wed Mar 31 11:20:16 2010] [error] [client 192.168.0.133] File does not exist: D:/Server/favicon.ico
- [Wed Mar 31 11:22:37 2010] [error] [client 192.168.0.133] File does not exist: D:/Server/favicon.ico
- [Wed Mar 31 11:22:41 2010] [error] [client 192.168.0.133] File does not exist: D:/Server/favicon.ico
- [Wed Mar 31 11:22:55 2010] [error] [client 192.168.0.133] File does not exist: D:/Server/favicon.ico
- [Mon Apr 12 08:41:33 2010] [notice] Apache/2.0.55 (Win32) PHP/5.2.1 configured -- resuming normal operations
- [Mon Apr 12 08:41:34 2010] [notice] Server built: Oct 9 2005 19:16:56
- [Mon Apr 12 08:41:34 2010] [notice] Parent: Created child process 848
- [Mon Apr 12 08:41:37 2010] [notice] Child 848: Child process is running
- [Mon Apr 12 08:41:37 2010] [notice] Child 848: Acquired the start mutex.
- [Mon Apr 12 08:41:37 2010] [notice] Child 848: Starting 250 worker threads.
- [Wed Apr 14 09:59:56 2010] [error] [client 192.168.0.133] PHP Parse error: syntax error, unexpected T\_STRING, expecting T\_VARIABLE or '\$' in D:\Server\lmpawi\lntet.php on line 15
- [Wed Apr 14 10:00:50 2010] [error] [client 192.168.0.133] File does not exist: D:/Server/favicon.ico
- [Wed Apr 14 10:00:50 2010] [error] [client 192.168.0.133] PHP Parse error: syntax error, unexpected T\_STRING in D:\Server\lmpawi\lntet.php on line 17
- [Wed Apr 14 10:00:50 2010] [error] [client 192.168.0.133] File does not exist: D:/Server/favicon.ico
- [Wed Apr 14 09:59:56 2010] [error] [client 192.168.0.133] PHP Parse error: syntax error, unexpected T\_STRING, expecting T\_VARIABLE, expecting
- [Wed Apr 14 09:59:56 2010] [error] [client 192.168.0.133] File does not exist: D:/Server/favicon.ico
- [Wed Apr 14 10:00:50 2010] [error] [client 192.168.0.133] PHP Parse error: syntax error, unexpected T\_STRING in
- [Wed Apr 14 10:00:50 2010] [error] [client 192.168.0.133] File does not exist: D:/Server/favicon.ico

# Metode de lucru recomandate 3

- In perioada de definitivare a formei interogarilor MySql este de multe ori benefic sa se utilizeze mai intai **MySql Query Browser/PhpMyAdmin** pentru incercarea interogarilor, urmand ca apoi, cand sunteti multumiti de rezultat, sa transferati interogarea SQL in codul PHP



The screenshot shows a MySQL Query Browser window with a tab labeled "Resultset 1". The "SQL Query Area" contains the following query:

```
1 SELECT p.*, c.`nume` AS `nume_categ` FROM `produse` AS p
2 LEFT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`)
```

Below the query area, a table displays the results of the query. The table has the following columns: id\_produș, id\_categ, nume, detalii, cant, pret, and nume\_categ. The first row is highlighted in blue.

id_produș	id_categ	nume	detalii	cant	pret	nume_categ
1	1	carte	mai multe pagini scrise legate	0	100	papetarie
2	1	caiet	mai multe pagini goale legate	0	75	papetarie
3	1	hartie scris	mai multe pagini goale NElegate	0	50	papetarie
4	2	penar	loc de depozitat instrumente de scris	0	150	instrumente
5	2	stilou	instrument de scris albastru	0	125	instrumente

# Metode de lucru recomandate 3

MySQL Query Browser - Connection: root@server / tmpaw

File Edit View Query Script Tools Window Help

Transaction Explain Compare

Resultset 1

SQL Query Area

```
1 SELECT p.*, c.`nume` AS `nume_categ` FROM `produse` AS p
2 LEFT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`)
```

id_produc	id_categ	nume	detalii	cant	pret	nume_categ
1	1	carte	mai multe pagini scrise legate	0	100	papetarie
2	1	caiet	mai multe pagini goale legate	0	75	papetarie
3	1	hartie scris	mai multe pagini goale NElegate	0	50	papetarie
4	2	penar	loc de depozitat instrumente de scris	0	150	instrumente
5	2	stilou	instrument de scris albastru	0	125	instrumente
6	2	creion	instrument de scris gri	0	25	instrumente
7	3	cd	canta	0	50	audio-video
8	3	dvd	vizual	0	100	audio-video
9	3	blue ray	vizual extrem	0	500	audio-video

9 rows fetched in 0.0035s (0.0016s)

Edit Apply Changes Discard Changes First Last Search

1: 1

# Metode de lucru recomandate 4

- eficienta unei aplicatii web
  - 100% - **toate prelucrarile "mutate" in RDBMS**
  - PHP **doar** afisarea datelor
- eficienta unei aplicatii MySql
  - 25% **alegerea corecta a tipurilor de date**
  - 25% **crearea indecsilor necesari in aplicatii**
  - 25% **normalizarea corecta a bazei de date**
  - 20% **cresterea complexitatii interogarilor pentru a "muta" prelucrarile pe server-ul de baze de date**
  - 5% **scrierea corecta a interogarilor**

# Metode de lucru recomandate 5

- La implementarea unei aplicatii noi (proiect)
  1. Imaginarea planului aplicatiei (ex: S14-S15)
    - "cum as vrea eu sa lucrez cu o astfel de aplicatie"
    - hartie/creion/timp – esentiale
  2. Identificarea datelor/transmisia de date intre pagini
    - get/post/fisier unic colectare-prelucrare
    - baza de date read/write
  3. Identificarea structurii logice a datelor utilizate
    - "clase" de obiecte/fenomene tratate identic
    - se are in vedere scalabilitatea (posibilitatea de crestere a numarului de elemente dintr-o clasa)

# Metode de lucru recomandate 5

- La implementarea unei aplicatii noi (proiect)
  4. Realizarea structurii bazei de date
    - In general un tabel pentru fiecare clasa logica distincta **DAR...**
    - se are in vedere scalabilitatea (daca aplicatia creste sa **NU** apara cresterea numarului de clase/tabele) **SI...**
    - normalizare
  5. Identificarea tipului de date necesar pentru coloane
    - de preferat numerele intregi in orice situatie care presupune ordonare
    - dimensiunea campurilor nu mai mare decat e necesar (poate fi fortata prin atributul "size" in eticheta HTML "input")
  6. Imaginarea formei fizice a paginilor
    - "am mai vazut asa si mi-a placut" (Don't make me think!)
    - investigarea posibilitatii de a introduce functionalitate template

# Metode de lucru recomandate 5

- La implementarea unei aplicatii noi (proiect)
  7. Popularea manuala a bazei de date cu date initiale
    - MySql Query Browser (sau PhpMyAdmin) / automat / imprumut
    - programarea individuala a paginilor are nevoie de prezenta unor date
  8. Programare individuala a paginilor
    - In general in ordinea din planul aplicatiei (de multe ori o pagina asigura datele necesare pentru urmatoarea din plan)
    - modul "verbose" activ pentru PHP (adica: `echo $a; print_r($matr)`)
  9. Pregatirea pentru distributie/mutare
    - testare detaliata (eventual un "cobai")
    - eliminarea adaosurilor "verbose"
    - backup
    - generarea unui eventual install/setup



# MySQL – eficienta

- eficienta unei aplicatii web
  - 100% - **toate prelucrarile "mutate" in RDBMS**
  - PHP **doar** afisarea datelor
- eficienta unei aplicatii MySQL
  - 25% **alegerea corecta a tipurilor de date**
  - 25% **crearea indecsilor necesari in aplicatii**
  - 25% **normalizarea corecta a bazei de date**
  - 20% **cresterea complexitatii interogarilor pentru a "muta" prelucrarile pe server-ul de baze de date**
  - 5% **scrierea corecta a interogarilor**

# Project

# Teme de proiect

- La toate temele **1p** din nota este obtinut de indeplinirea functionalitatii cerute.
- La toate temele forma paginii prezinta importanta (dependenta de dificultatea temei)

# PROIECT (final)

- Tema de nota **7 (>6)**
  - Tema unica pentru fiecare student
  - Baza de date cu care se lucreaza contine minim **15** de inregistrari in tabelul cel mai "voluminos«
- Tema de nota **8 (>6)**
  - Conditiiile de la tema de nota 7 **si in plus**
  - Necesitatea conlucrarii intre **2 studenti** cu doua teme "pereche"
  - Se accepta ca un student sa realizeze ambele puncte
  - Numar **minim** de pagini dinamice (php+mysql) in aplicatie **4 = 2 X 2**
  - Baza de date cu care se lucreaza contine minim **30** de inregistrari in tabelul cel mai "voluminos"

# PROIECT (final)

- Tema de nota **9 (>5)**
  - Condițiile de la tema de nota 8 **si in plus**
  - Necesitatea conlucrării între 2 studenti cu teme "pereche"
  - Tema se preda/trimite cu macar 1 zi înainte de susținerea ei
  - Numar **minim** de pagini dinamice (php+mysql) in aplicatie **6 = 3 X 2**
  - Baza de date cu care se lucreaza sa contina minim 60 de inregistrari in tabelul cel mai "voluminos".

# PROIECT (final)

- Tema de nota **10 (>5)**
  - Condițiile de la tema de nota 9 **si in plus**
  - Numar **minim** de pagini dinamice (php+mysql) in aplicatie **8 = 4 X 2**
  - Baza de date cu care se lucreaza contine minim **300** de inregistrari in tabelul cel mai "voluminos"
  - Necesitatea investigarii posibilitatilor de **imbunatatire** a aplicatiei si adaugarii de functionalitate
  - nota individuala la proiect va depinde intr-o mica masura (in limita a 1p) de nota minima a colegilor din echipa

# PROIECT (final)

- proiectul se sustine individual (oral si practic)
- grila de notare la proiect schimbata fata de anii precedenti
- fiecare membru al unei echipe (la temele de nota 9 si 10) trebuie sa sustina in aceeasi zi proiectul
- nota individuala la proiect va depinde intr-o mica masura (in limita a 1p) de nota medie a colegilor din echipa (numai la temele de 10 si 10+)
  - $N-\min(E)=1 \rightarrow -0.5 p$
  - $N-\min(E)=2 \rightarrow -0.5 p$
  - $N-\min(E)=3 \rightarrow -1 p$

# PROIECT (final)

- In caz de necesitate, pentru completarea echipei cadrul didactic poate fi membru al echipelor (9/10/10+). Conditii:
  - metoda de comunicare in echipa sa fie prin email sau direct
  - latentă de raspuns: ~ 1 zi
  - reactiv
  - nota implicita 10 ( 😊 )
  - nu lucreaza noaptea, si in special nu in noaptea dinaintea predarii ( 😊 )
- dezavantaj asumat: "spion" in echipa



# PROIECT (final)

- Tema de nota **10+** (>5, in general **offline**)
  - Conditiiile de la tema de nota 10 **si in plus**
  - Baza de date cu care se lucreaza contine minim **500** de inregistrari in tabelul cel mai "voluminos«
  - Numar **minim** de pagini dinamice (php+mysql) in aplicatie **15 = 5 X 3**
  - Tema care face apel la controlul **sesiunii** client/server
  - Necesitatea utilizarii **Javascript** in **aplicatie** (aplicatie libera dar cu efect tehnic nu estetic)
  - Forma paginii trebuie sa respecte cerintele "F shape pattern"
  - Facilitati in ceea ce priveste prezenta la laborator (**DACA** toate celelalte conditii sunt indeplinite – P = **66%**, L = **0%**, E = **33%**)

# Exemplu

- 1. Galerie de imagini in care imaginile sunt ordonate dupa categorii.

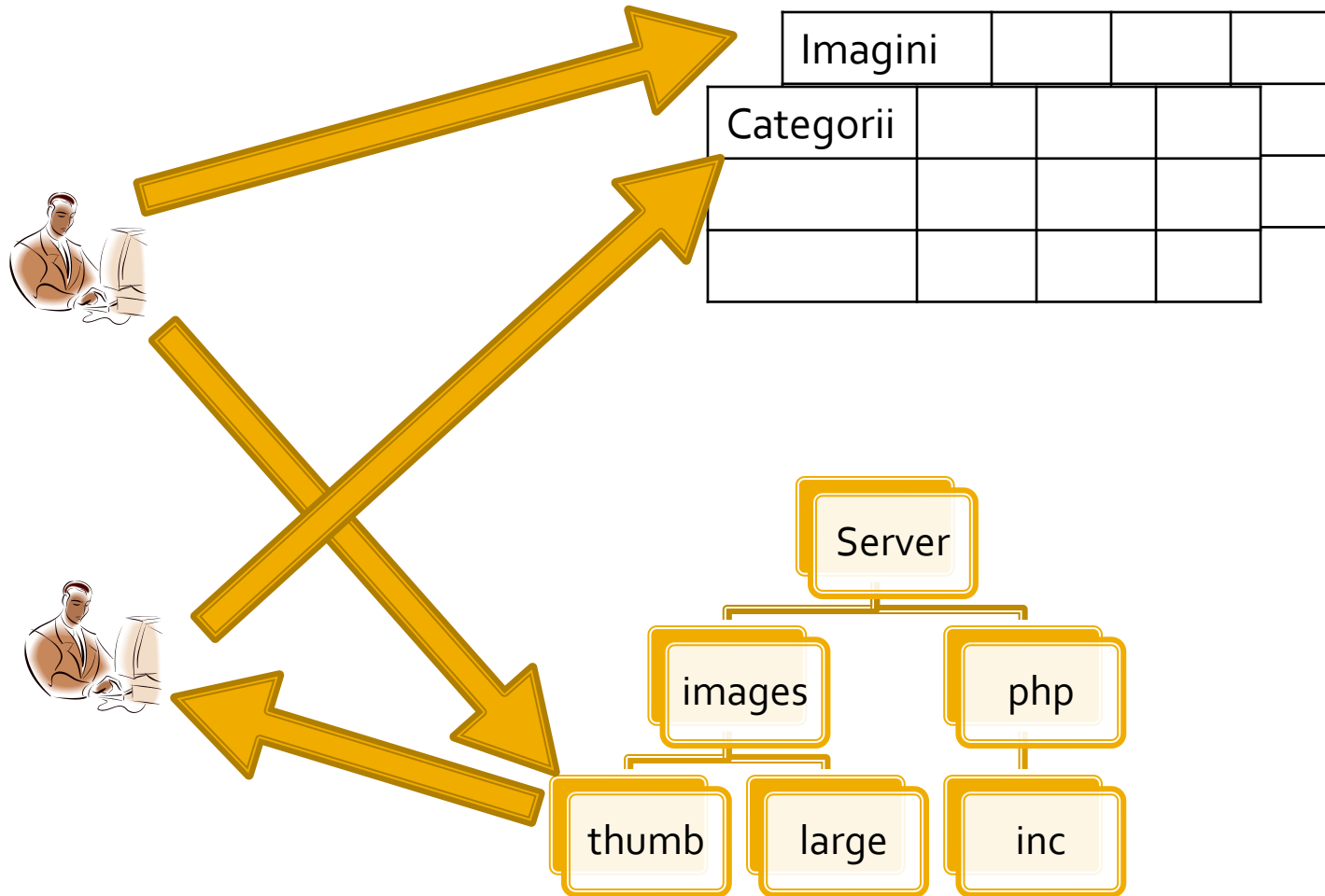


a. aplicatia pentru adaugarea de categorii si afisare a imaginilor (cu alegerea prealabila a categoriei si afisarea listei de imagini format mic)



b. aplicatia pentru adaugare de imaginilor (cu alegerea prealabila a categoriei si generarea prealabila a imaginii format mic)

# Exemplu



# Teme de proiect

- **Functionalitate**
  - La toate temele **1p** din nota este obtinut de indeplinirea functionalitatii cerute.
  - orice tehnologie, orice metoda, "sa faca ceea ce trebuie"
- **Forma paginii prezinta importanta**
  - dependenta de dificultatea temei
- **Initiativa**
  - **Necesitatea** investigarii posibilitatilor de imbunatatire
- **Cooperare**
  - Necesitatea conlucrarii intre 2/3 studenti cu teme "pereche"

# Notare

- 1p – functionalitate
  - cadrul didactic va incerca sa foloseasca aplicatia respectiva. Daca "pe dinafara e vopsit gardul" se obtine 1p
- 1p – mutarea site-ului (restaurare backup + setare server) pe un server de referinta
  - server-ul de referinta va fi masina virtuala utilizata la laborator (inclusiv aplicatiile cu pricina)
  - sa va pregatiti pentru situatia in care pe acel server exista si alte baze de date care nu trebuie distruse
  - fiecare student isi pune sursele in directorul propriu, in radacina server-ului. Daca tema depinde de anumite fisiere ale colegului, le cereti inainte
- 1p – cunoasterea codului
  - raspunsul la intrebari de genul: "unde ai facut aceasta"
- Teme "de nota 10"
  - 1p – initiativa. Investigarea posibilitatilor de imbunatatire
  - 1p – intrebari legate de cooperarea cu colegul de echipa
  - 1p – explicatii relativ la functionarea unei anumite secvente de cod

# Notare proiect 2016/2017

- grila de notare diferita
  - premiera activitatii individuale
  - mai greu de obtinut note mari
- 1p – functionalitate ✓
- numar de pagini dinamice ✓
- numar de inregistrari in baza de date ✓
- planul aplicatiei ✓

# Notare 2017

- numar de pagini dinamice ✓
- numar de inregistrari in baza de date ✓
  - se verifica indeplinirea conditiilor corespunzatoare si se realizeaza **de-clasificarea** temei pana cand **ambele** conditii sunt indeplinite

Tema de nota ...	Pagini	Inregistrari
10+	$15 = 5 \times 3$	500
10	$8 = 4 \times 2$	300
9	$6 = 3 \times 2$	60
8	$4 = 2 \times 2$	30
7	$1 = 1 \times 1$	15

# Notare 2017

- 1p – functionalitate
- 1p – mutarea **personala** a site-ului (restaurare backup + setare server) pe un server de referinta
  - server-ul de referinta va fi masina virtuala **Centos 7.1** utilizata la laborator (inclusiv aplicatiile cu pricina)
  - sa va pregatiti pentru situatia in care pe acel server exista si alte baze de date care **nu** trebuie distruse
  - fiecare student isi pune sursele in directorul propriu, in radacina server-ului. Daca tema depinde de anumite fisiere ale colegului, le cereti inainte
- 1p – cunoasterea codului
  - raspunsul la intrebari de genul: “unde ai facut aceasta”
- Teme “de nota 10,10+”
  - initiativa. Investigarea posibilitatilor de imbunatatire
  - intrebari legate de cooperarea cu colegul de echipa
  - explicatii relativ la functionarea unei anumite secvente de cod
  - utilizare sesiune, Javascript, F shape pattern



# Examen

- probleme
- fiecare student are subiect propriu
- toate materialele permise
- tehnica de calcul **nu** este necesara dar este permisa

# Examen

- Oricare din temele de proiect (sau asemenea) poate constitui una din problemele de examen
  - se va cere realizarea planului / structurii logice a aplicatiei (S5)
- Se poate cere scrierea unui cod pentru realizarea anumitor operatii, fara necesitatea corectitudinii tehnice absolute (";", nume corect al functiilor, parametri functie etc.)
- Se poate cere interpretarea unui cod php/MySql cu identificarea efectului

MySql

# Accesul la metode externe de stocare eficiente a datelor

# Normalizare

- Normalizarea asigura:
  - stocarea eficienta a datelor
  - prelucrarea eficienta a datelor
  - integritatea datelor
- Trei nivele de normalizare
- Eliminarea datelor redundante

OrderID	CustomerID	OrderDate	Items	OrderTotal
1	CACTU	1/1/1999	3 Zaanse koeken, 1 Tarte au sucre	\$89.70
2	BSBEV	1/5/1999	4 Mozzarella di Giovanni	\$139.20
3	SUPRD	5/2/1999	3 Ravioli Angelo, 6 Tofu	\$198.06

MySql – Recapitulare rapida

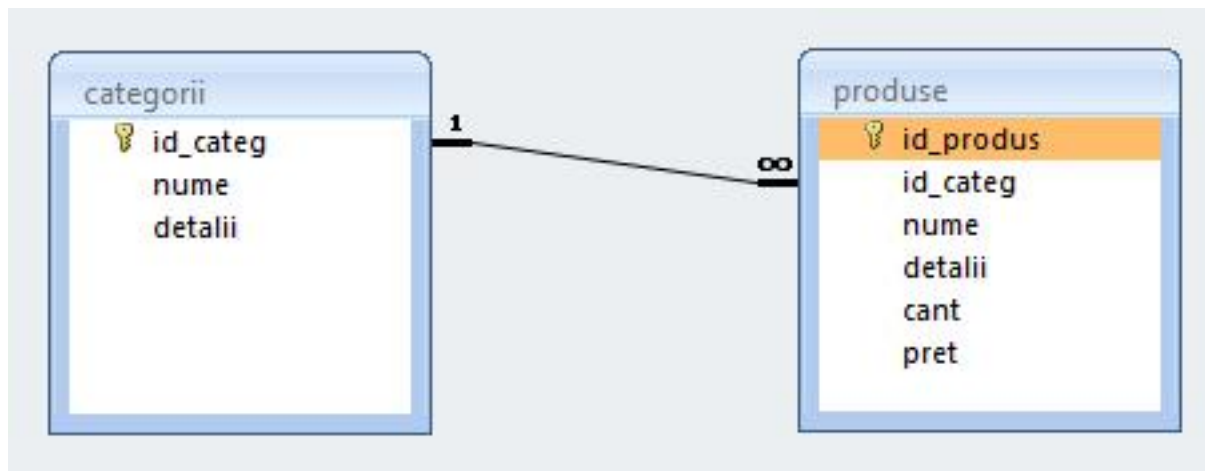
# Relatii in Bazele de date

# Relatii in Bazele de date

- In exemplul utilizat avem doua concepte diferite din punct de vedere logic
  - produs
  - categorie de produs
- Cele doua tabele nu sunt independente
- Intre ele exista o legatura data de functionalitatea dorita pentru aplicatie: **un produs va apartine unei anumite categorii de produse**

# Relatii in Bazele de date

- Legaturile implementata
  - One to Many
  - in tabelul "produse" apare cheia externa (foreign key): "id\_categ"

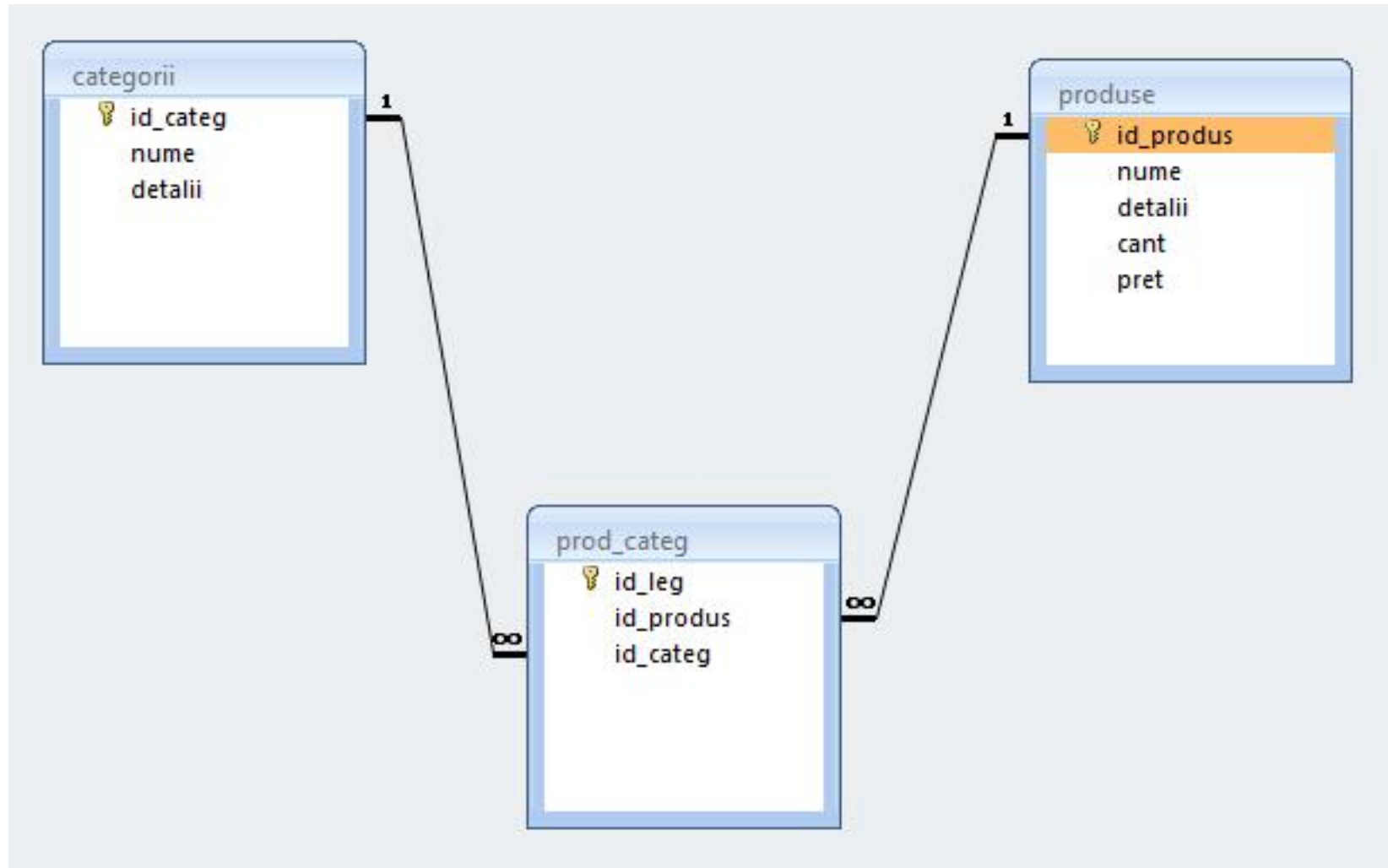


# Relatii in Bazele de date

- Daca se doreste o situatie cand un produs poate apartine **mai multor categorii** (o carte cu CD poate fi inclusa si in "papetarie" si in "audio-video")
  - relatia devine de tipul **Many to Many**
  - e necesara introducerea unui tabel de legatura cu coloanele "id\_leg" (cheie primara), "id\_categorie" si "id\_produs" (chei externe)



# Relatii in Bazele de date



# Relatii

- **Nu** trebuie evitate relatiile
  - Many to Many
  - One to Many
- Prelucrarea cade in sarcina server-ului de baze de date (**RDBMS**)
  - JOIN – **esential** in aplicatii cu baze de date

# MySQL – eficienta

- eficienta unei aplicatii web
  - 100% - **toate prelucrarile "mutate" in RDBMS**
  - PHP **doar** afisarea datelor
- eficienta unei aplicatii MySQL
  - 25% **alegerea corecta a tipurilor de date**
  - 25% **crearea indecsilor necesari in aplicatii**
  - 25% **normalizarea corecta a bazei de date**
  - 20% **cresterea complexitatii interogarilor pentru a "muta" prelucrarile pe server-ul de baze de date**
  - 5% **scrierea corecta a interogarilor**

# Acces la server-ul MySQL din PHP

# Acces la server-ul MySQL din PHP

- Bibliotecile corespunzatoare trebuie activate in php.ini – vezi laboratorul 1.
  - mysql
  - mysqli (improved accesul la functionalitati ulterioare MySQL 4.1)
- O baza de date existenta poate fi accesata daca exista un utilizator cunoscut in PHP cu drepturi de acces corespunzatoare – vezi laboratorul 1.
- O baza de date poate fi creata si din PHP dar nu e metoda recomandata daca nu e necesara
  - cod dificil de implementat pentru o **singura** utilizare
  - necesita existenta unui utilizatori cu drepturi mai mari pentru crearea bazei de date si alocarea de drepturi unui utilizator restrans

# Funcții PHP de acces MySQL

- `mysql_query`
  - trimiterea unei interogari SQL spre server
  - resource `mysql_query` ( string query [, resource link\_identifier] )
  - rezultatul
    - SELECT, SHOW, DESCRIBE sau EXPLAIN – resursa (tabel)
    - UPDATE, DELETE, DROP, etc – true/false
- `mysql_fetch_assoc`
  - returneaza o **matrice asociativa** corespunzatoare liniei de la indexul intern (indecsi de tip sir corespunzatori denumirii coloanelor – field – din tabelul de date) si incrementeaza indexul intern sau **false** daca nu mai sunt linii
  - array `mysql_fetch_assoc` ( resource result )

# Funcții PHP de acces MySQL

## Parcurgerea resurselor rezultat

- `mysql_fetch_assoc`
  - returnează o **matrice asociativă** corespunzătoare liniei de la indexul intern (indecsi de tip șir corespunzători denumirii coloanelor – field – din tabelul de date) și incrementează indexul intern sau **false** dacă nu mai sunt linii
  - array `mysql_fetch_assoc` ( resource result )
- `mysql_fetch_row`
  - returnează o matrice cu indecsi întregi
  - array `mysql_fetch_row` ( resource result )

# Funcții PHP de acces MySQL

## Parcurgerea resurselor rezultat

- `mysql_fetch_array`
  - grupează funcționalitatea `mysql_fetch_assoc` și `mysql_fetch_row`
  - array `mysql_fetch_array` ( resource result [, int result\_type] )
  - `MYSQL_ASSOC`, `MYSQL_NUM`, `MYSQL_BOTH` (implicit)
- `mysql_data_seek`
  - muta indexul intern la valoarea indicată
  - bool `mysql_data_seek` ( resource result, int row\_number )



# Resurse MySQL

- Resursele reprezinta o combinatie intre
  - date structurate (valori + structura) rezultate in urma unor interogari SQL
  - functii de acces la aceste date/structuri
- Analogie cu POO
  - o "clasa speciala" creata in urma interogarii cu functii predefinite de acces la datele respective

# Resurse MySQL

## Structura

Index intern	Col 1 (tip date)	Col 2 (tip date)	....
1			
2			
...			

## Date

Index intern	Col 1	Col 2	....
1	Val 11	Val 12	...
2	Val 21	Val 22	...
...	...	...	...

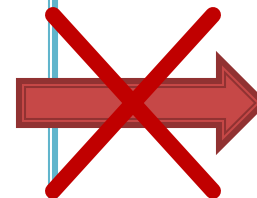
Functii de acces la structura



Functii de acces la date



Acces direct



# Resurse MySQL

- Functiile de acces la structura sunt rareori utilizate
  - majoritatea aplicatiilor sunt concepute pe structura fixa, si cunosc structura datelor primite
  - exceptie: aplicatii generale, ex.: PhpMyAdmin
- Majoritatea functiilor de acces la date sunt caracterizate de acces secvential
  - se citesc in intregime valorile stocate pe o linie
  - simultan se avanseaza indexul intern pe urmatoarea pozitie, pregatindu-se urmatoarea citire

# Resurse MySQL

- Functiile sunt optimizate pentru utilizarea lor intr-o structura de control **do {} while()**, sau **while() {}** de control
  - returneaza FALSE cand "s-a ajuns la capat"
- tipic se realizeaza o citire (mysql\_fetch\_assoc) urmata de o bucla **do {} while()**
  - pentru a se putea introduce cod de detectie probleme rulat o singura data

# Exemplu de utilizare

```
$hostname = "localhost";  
$database = "world";  
$username = "web";  
$password = "ceva";  
$conex= mysql_connect($hostname, $username, $password);  
mysql_select_db($database, $conex);
```

```
$query = "SELECT `Code`, `Name`, `Population` FROM `country` AS c ";  
$result = mysql_query($ query, $conex) or die(mysql_error());  
$row_result = mysql_fetch_assoc($ result );  
$totalRows_result = mysql_num_rows($ result );
```

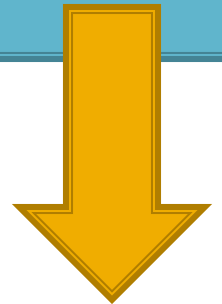
# Exemplu de utilizare

```
<?php
do {?>
<tr>
    <td><?php echo $index; ?>&nbsp;  </td>
    <td><?php echo $ row_result ['Code']; ?>&nbsp;  </td>
    <td><?php echo $ row_result ['Name']; ?>&nbsp;  </td>
    <td><?php echo $ row_result ['Population']; ?>&nbsp;  </td>
</tr>
<?php
    $index++;
}
while ($ row_result = mysql_fetch_assoc($ result )); ?>
```

# Modificari laborator cu date stocate text

- Codul aplicatiei ramane in mare parte acelasi
- Se modifica doar citirea valorilor pentru popularea matricii \$produse ("antet.php")

```
$matr=file("produse.txt");  
foreach ($matr as $linie)  
    {  
        $valori=explode("\t",$linie,5);  
        $produse[$valori[0]] [$valori[1]]=array ("descr" => $valori[2], "pret" => $valori[3], "cant" =>  
$valori[4]);  
    }
```



# Modificari laborator cu date stocate

## MySQL

```
$hostname = "localhost";
$database = "tmpaw";
$username = "web";
$password = "test";
$conex= mysql_connect($hostname, $username, $password);
mysql_select_db($database, $conex);
$query = "SELECT * FROM `categorii` AS c";
$result_c = mysql_query($query, $conex) or die(mysql_error());
$row_result_c = mysql_fetch_assoc($result_c);
$totalRows_result = mysql_num_rows($result_c);
do {
    $query = "SELECT * FROM `produse` AS p WHERE `id_categ` = ".$row_result_c['id_categ'];
    $result_p = mysql_query($query, $conex) or die(mysql_error());
    $row_result_p = mysql_fetch_assoc($result_p);
    $totalRows_result = mysql_num_rows($result_p);
    $produse[$row_result_c['nume']] = array();
    do {
        $produse[$row_result_c['nume']][$row_result_p['nume']] = array ("descr" =>
$row_result_p['detalii'], "pret" => $row_result_p['pret'], "cant" => $row_result_p['cant']);
    }
    while ($row_result_p = mysql_fetch_assoc($result_p));
}
while ($row_result_c = mysql_fetch_assoc($result_c));
```



# MySQL – eficienta

- eficienta unei aplicatii web
  - 100% - **toate prelucrarile "mutate" in RDBMS**
  - PHP **doar** afisarea datelor
- eficienta unei aplicatii MySQL
  - 25% **alegerea corecta a tipurilor de date**
  - 25% **crearea indecsilor necesari in aplicatii**
  - 25% **normalizarea corecta a bazei de date**
  - 20% **cresterea complexitatii interogarilor pentru a "muta" prelucrarile pe server-ul de baze de date**
  - 5% **scrierea corecta a interogarilor**

# Optimizare

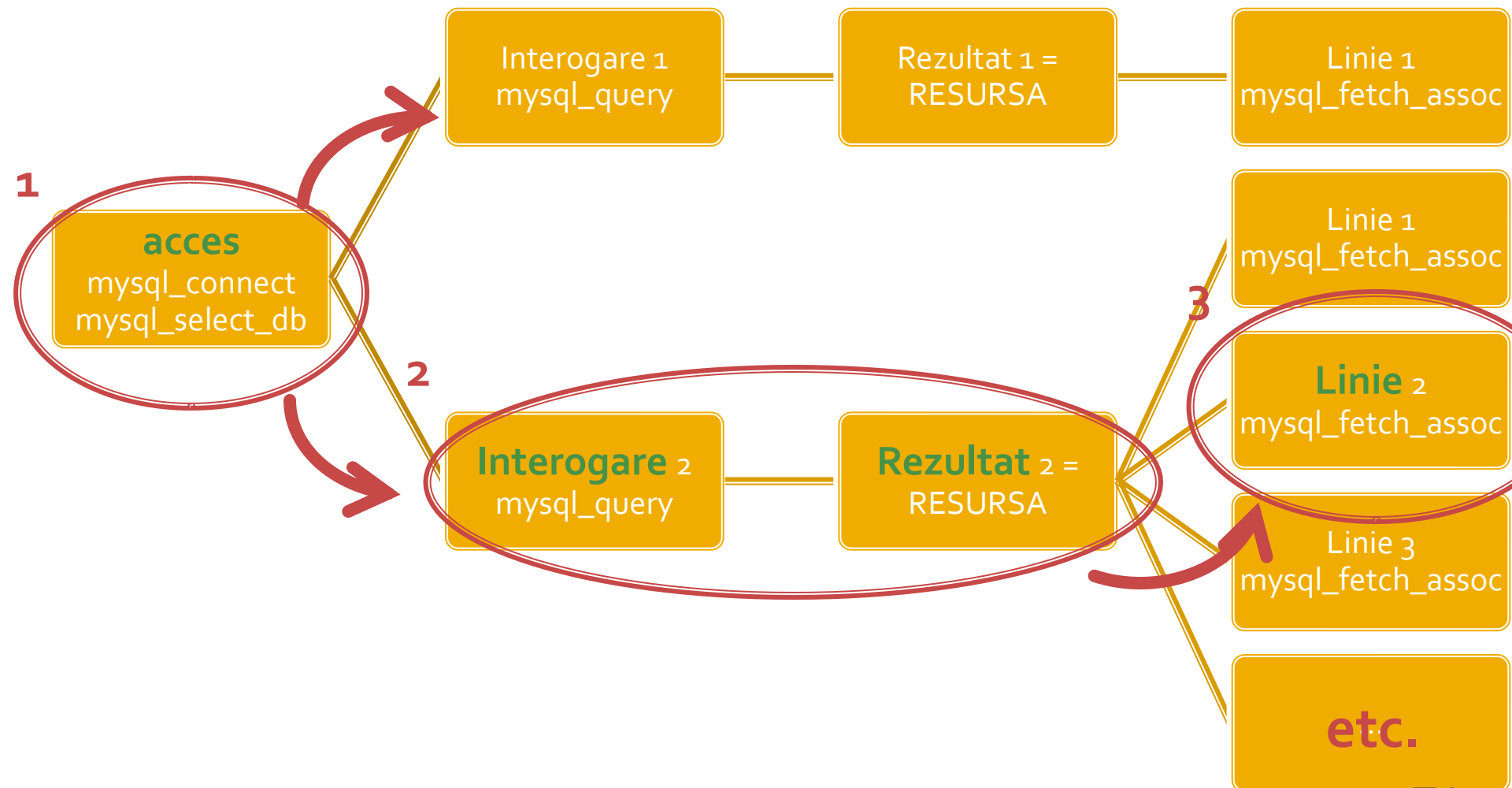
- o singura interogare SQL, unirea tabelelor lasata in baza server-ului MySQL

```
$hostname = "localhost";
$database = "tmpaw";
$username = "web";
$password = "test";
$conex= mysql_connect($hostname, $username, $password);
mysql_select_db($database, $conex);

$query = "SELECT p.*, c.`nume` AS `nume_categ` FROM `produse` AS p
        LEFT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`)";
$result = mysql_query($query, $conex) or die(mysql_error());
$row_result = mysql_fetch_assoc($result);
$totalRows_result = mysql_num_rows($result);

do{
    $produse[$row_result['nume_categ']][$row_result['nume']] = array ("descr" => $row_result['detalii'], "pret"
=> $row_result['pret'], "cant" => $row_result['cant']);
}
while ($row_result = mysql_fetch_assoc($result));
```

# Funcții de acces la server-ul MySQL



!! IMPORTANT

**PHP > 5.5**

# PHP 5.5

- Incapand cu versiunea 5.5 a PHP extensia mysql este declarata **depreciata**
  - orice utilizare a unei functii genereaza eroare de tip **E\_DEPRECATED**
  - se preconizeaza ca in PHP > 6 aceasta extensie va fi eliminata total
- Alternativele de utilizare sunt
  - extensia mysqli (MySQL Improved)
  - extensia PDO (PHP Data Objects)

# Extensia mysqli

- Inafara securitatii sporite ofera acces la facilitatile curente ale server-ului MySQL
  - accesul la interogari predefinite (Prepared Statements) (viteza, securitate)
    - server side
    - client side
  - proceduri stocate pe server (viteza, securitate)
  - interogari multiple
  - tranzactii (integritate)

# Extensia mysqli

- Doua modalitati de utilizare
  - procedurala (similar mysql)
  - POO (similar PDO)
- Utilizarea procedurala (aproape) similara cu utilizarea extensiei originale mysql
  - tranzitie facila
  - tranzitie cu mici diferente de parametri

# mysqli – Procedural

```
<?php
$mysqli = mysqli_connect("example.com", "user", "password", "database");
$res = mysqli_query($mysqli, "SELECT 'Please do not use the mysql extension ' AS _msg FROM DUAL");
$row = mysqli_fetch_assoc($res);
echo $row['_msg'];

$mysqli = mysqli_connect("example.com", "user", "password");
mysqli_select_db("test");
$res = mysqli_query("SELECT ' for new developments.' AS _msg FROM DUAL", $mysqli);
$row = mysqli_fetch_assoc($res);
echo $row['_msg'];
?>
```

- toate functiile mysql au un echivalent mysqli
- majoritatea functiilor au aceeasi parametri in aceeasi ordine
- sunt totusi functii cu mici diferente (Ex: **mysqli\_connect**, **mysqli\_query**)



# mysqli – Programare orientata obiect

```
<?php
$var = new mysqli("example.com", "user", "password", "database");
$res = $var->query ( "SELECT 'Please do not use the mysql extension ' AS _msg FROM DUAL");
$row = $res->fetch_assoc();
echo $row['_msg'];

$mysql = mysqli_connect("example.com", "user", "password");
mysqli_select_db("test");
$res = mysqli_query("SELECT ' for new developments.' AS _msg FROM DUAL", $mysql);
$row = mysqli_fetch_assoc($res);
echo $row['_msg'];
?>
```

# Resurse MySQL – mysqli

## Structura

Index intern	Col 1 (tip date)	Col 2 (tip date)	....
1			
2			
...			

## Date

Index intern	Col 1	Col 2	....
1	Val 11	Val 12	...
2	Val 21	Val 22	...
...	...	...	...

## Metode

Constructor	query	fetch_assoc	....
-------------	-------	-------------	------

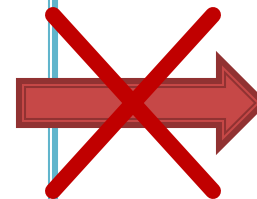
Functii de acces la structura



Functii de acces la date



Acces direct



Metode atasate resursei



# Conversia la mysql (obligatorie)

## ■ exemplul anterior

```
$hostname = "localhost";
$database = "tmpaw";
$username = "web";
$password = "test";
$conex= mysql_connect($hostname, $username, $password);
mysql_select_db($database, $conex);

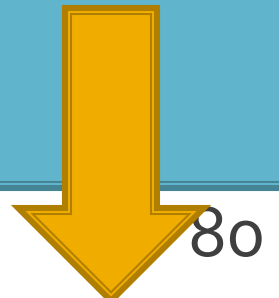
$query = "SELECT p.*, c.`nume` AS `nume_categ` FROM `produse` AS p
        LEFT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`)";
$result = mysql_query($query, $conex) or die(mysql_error());
$row_result = mysql_fetch_assoc($result);
$totalRows_result = mysql_num_rows($result);

do{
    $produse[$row_result['nume_categ']][$row_result['nume']]=array ("descr" => $row_result['detalii'], "pret"
=> $row_result['pret'], "cant" => $row_result['cant']);
}
while ($row_result = mysql_fetch_assoc($result));
```



# mysqli (Procedural)

```
//$conex= mysql_connect($hostname, $username, $password);  
//mysql_select_db($database, $conex);  
$conex = mysqli_connect($hostname, $username, $password, $database);  
  
$query = "SELECT p.*, c.`nume` AS `nume_categ` FROM `produse` AS p  
        LEFT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`)";  
//$result = mysql_query($query, $conex) or die(mysql_error());  
$result = mysqli_query($conex, $query);  
  
//$row_result = mysql_fetch_assoc($result);  
$row_result = mysqli_fetch_assoc($result);  
  
//$totalRows_result = mysql_num_rows($result);  
$totalRows_result = mysqli_num_rows($result);  
  
do {  
    $produse[$row_result['nume_categ']][$row_result['nume']]=array ("descr" => $row_result['detalii'], "pret"  
=> $row_result['pret'], "cant" => $row_result['cant']);  
    }  
//while ($row_result = mysql_fetch_assoc($result));  
while ($row_result = mysqli_fetch_assoc($result));
```



# mysqli (POO)

```
//$conex= mysql_connect($hostname, $username, $password);
//mysql_select_db($database, $conex);
//$conex = mysqli_connect($hostname, $username, $password, $database);
$conex = new mysqli($hostname, $username, $password, $database);

$query = "SELECT p.*, c.`nome` AS `nome_categ` FROM `produse` AS p
        LEFT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`)";
//$result = mysql_query($query, $conex) or die(mysql_error());
//$result = mysqli_query($conex, $query);
$result = $conex->query( $query );

//$row_result = mysql_fetch_assoc($result);
//$row_result = mysqli_fetch_assoc($result);
$row_result = $result->fetch_assoc();

//$totalRows_result = mysql_num_rows($result);
//$totalRows_result = mysqli_num_rows($result);
$totalRows_result = $result->num_rows;

do {
    $produse[$row_result['nome_categ']][$row_result['nome']]=array ("descr" => $row_result['detalii'], "pret"
=> $row_result['pret'], "cant" => $row_result['cant']);
}
//while ($row_result = mysql_fetch_assoc($result));
while ($row_result = $result->fetch_assoc(););
```

MySql

# Tipuri de date

# MySQL – tipuri de date

- numeric
  - intregi
    - BIT (implicit 1 bit)
    - TINYINT (implicit 8 biti)
    - SMALLINT (implicit 16 biti)
    - INTEGER (implicit 32biti)
    - BIGINT (implicit 64biti)
  - real
    - FLOAT
    - DOUBLE
    - DECIMAL – fixed point

# MySQL – tipuri de date

- data/timp
  - DATE ('YYYY-MM-DD')
    - '1000-01-01' pana la '9999-12-31'
  - DATETIME ('YYYY-MM-DD HH:MM:SS')
    - '1000-01-01 00:00:00' pana la '9999-12-31 23:59:59'
  - TIMESTAMP ('YYYY-MM-DD HH:MM:SS')
    - '1970-01-01 00:00:00' pana la partial 2037



# MySQL – tipuri de date

- sir
  - CHAR (M)
    - sir de lungime constanta M,  $M < 255$
  - VARCHAR (M)
    - sir de lungime variabila, maxim M,  $M < 255$  ( $M < 65535$ )
- cantitati mari de date
  - TEXT
    - au alocat un set de caractere, operatiile tin cont de acesta
  - BLOB
    - sir de octeti, operatiile tin cont de valoarea numerica
  - TINYBLOB/TINYTEXT, BLOB/TEXT, MEDIUMBLOB/MEDIUMTEXT, LARGEBLOB/LARGETEXT
    - date  $2^8-1$ ,  $2^{16}-1$ ,  $2^{24}-1$ ,  $2^{32}-1 = 4\text{GB}$

# MySQL – tipuri de date

- enumerare

- ENUM('val<sub>1</sub>', 'val<sub>2</sub>', ...)

- una singura din cele maxim 65535 valori distincte posibile

- SET('val<sub>1</sub>', 'val<sub>2</sub>', ...)

- niciuna sau mai multe din cele maxim 64 valori distincte
    - echivalent cu "setare de biti" într-un întreg pe 64 biti cu tabela asociată

# Limbas SQL

# MySQL – eficienta

- eficienta unei aplicatii web
  - 100% - **toate prelucrarile "mutate" in RDBMS**
  - PHP **doar** afisarea datelor
- eficienta unei aplicatii MySQL
  - 25% **alegerea corecta a tipurilor de date**
  - 25% **crearea indecsilor necesari in aplicatii**
  - 25% **normalizarea corecta a bazei de date**
  - 20% **cresterea complexitatii interogarilor pentru a "muta" prelucrarile pe server-ul de baze de date**
  - 5% **scrierea corecta a interogarilor**

# Referinta relativa

- Referinta la elementele unei baze de date se face prin utilizarea numelui elementului respectiv daca nu exista dubii (referinta relativa)
  - daca baza de date este selectata se poate utiliza numele tabelului pentru a identifica un tabel
    - `USE db_name;`  
`SELECT * FROM tbl_name;`
  - daca tabelul este identificat in instructiune se poate utiliza numele coloanei pentru a identifica coloana implicata
    - `SELECT col_name FROM tbl_name;`

# Referinta absoluta

- In cazul in care apare ambiguitate in identificarea unui element se poate indica descendenta sa pâna la disparitia ambiguitatii
- Astfel, o anumita coloana, `col_name`, care apartine tabelului `tbl_name` din baza de date (schema) `db_name` poate fi identificata in functie de necesitati ca:
  - `col_name`
  - `tbl_name.col_name`
  - `db_name.tbl_name.col_name`

# Nume de identificatori permise

- Numele de identificatori pot avea o lungime de reprezentare de maxim 64 octeti cu exceptia Alias care poate avea o lungime de 255 octeti
- Nu sunt permise:
  - caracterul NULL (ASCII 0x00) sau 255 (0xFF)
  - caracterul "/"
  - caracterul "\"
  - caracterul "."
- Numele nu se pot termina cu caracterul spatiu

# Nume de identificatori permise

- Numele de baze de date nu pot contine decat caractere permise in numele de directoare
- Numele de tabele nu pot contine decat caractere permise in numele de fisiere
- Anumite caractere utilizate vor impune necesitatea trecerii intre apostroafe a numelui
- Apostroful utilizat pentru nume de identificatori e apostroful invers (**backtick**) “`”
  - pentru a nu aparea confuzie cu variabilele sir
  - nu necesita aparitia apostrofului caracterele alfanumerice normale, “\_”, “\$”
- numele rezervate trebuie de asemenea cuprinse intre apostroafe pentru a fi utilizate



# Alias

- Orice identificator poate primi un nume asociat
  - **Alias**
    - pentru a elimina ambiguitati
    - pentru a usura scrierea
    - pentru a modifica numele coloanelor in rezultate
- Definirea unui alias se face in interiorul unei interogari SQL si are efect in aceeasi interogare
  - `SELECT `t`.* FROM `tbl_name` AS t;`
  - `SELECT `t`.* FROM `tbl_name` t;`

# Alias

- Desi utilizarea cuvintului cheie AS nu este obligatorie, obisnuinta utilizarii lui este recomandata, pentru a evita/identifica alocari eronate
  - `SELECT id, nume FROM produse;` ← doua coloane
  - `SELECT id nume FROM produse;` ← Alias "nume" creat pentru coloana "id"

# Alias

- Usurinta scrierii
  - `SELECT * FROM un_tabel_cu_nume_lung AS t WHERE t.col1 = 5 AND t.col2 = 'ceva'`
- Modificarea numelui de coloana, sau crearea unui nume pentru o coloana calculata in rezultate
  - `SELECT CONCAT(ume, " ", prenume) AS nume_intreg FROM studenti AS s;`
  - `SELECT `n1` AS `Nume`, `n2` AS `Nota`, `n3` AS `Numar matricol` FROM elevi AS e;`

# Alias

- Eliminarea ambiguitatilor
  - intalnita frecvent la relatii "many to many"
  - `SELECT p.*, c.`nume` AS `nume_categ` FROM `produse` AS p LEFT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`);`
  - tabelele c si p contin ambele coloanele "nume" si "id\_categ"
    - modificarea denumirii coloanei "nume" din categorii pentru evitarea confuziei cu coloana "nume" din produse
    - eventual se pot da nume diferite coloanelor "id\_categ" pentru a evita ambiguitatea in interiorul clauzei ON (desi si referinta absoluta rezolva aceasta problema)

# Metode de stocare

- Metoda de stocare a datelor nu e o caracteristica a server-ului ci a fiecarui tabel in parte
- Exemplu ulterior CREATE: "ENGINE = InnoDB"
- MySql suporta diferite metode de stocare, fiecare cu avantajele/dezavantajele sale
- Implicit se foloseste metoda MyISAM, dar la instalarea server-ului (laborator 1) o anumita selectie poate schimba valoarea implicita in InnoDB
- **Alegerea metodei de stocare potrivita are implicatii majore asupra performantei aplicatiei**

# Metode de stocare

- MyISAM
- InnoDB
- Memory
- Merge
- Archive
- Federated
- NDBCLUSTER
- CSV
- Blackhole
- Example

# Metode de stocare

## ■ MyISAM

- metoda de stocare implicita in MySql
- performanta ridicata (resurse ocupate si viteza)
- posibilitatea cautarii in intregul text (index FULLTEXT)
- blocare acces la nivel de tabel
- **nu** accepta tranzactii
- **nu** accepta FOREIGN KEY
  - probleme relative la integritatea datelor

## ■ InnoDB

## ■ Memory

# Metode de stocare

- **MyISAM**
- **InnoDB**
  - devine metoda de stocare implicita in MySql daca la instalare se alege model tranzactional
  - performanta medie (resurse ocupate si viteza)
  - blocare acces la nivel de linie
  - **nu** accepta index FULLTEXT
    - incepand cu MySql 5.6.4 este introdus index FULLTEXT
  - **accepta** tranzactii
  - **accepta** FOREIGN KEY
    - probleme mai putine la integritatea datelor prin constrangeri intre tabele
- **Memory**



# Metode de stocare

- MyISAM
- InnoDB
- **Memory**
  - metoda de stocare recomandata pentru tabele temporare
  - performanta maxima (viteza – datele sunt stocate in RAM)
    - **la oprirea server-ului datele se pierde**, tabelul este pastrat dar va fi fara nici o linie
  - **nu** accepta tipuri de date mari (BLOB, TEXT) – maxim 255 octeti
  - **nu** accepta index FULLTEXT
  - **nu** accepta tranzactii
  - **nu** accepta FOREIGN KEY
    - probleme relative la integritatea datelor

# Interrogari SQL

# Interogari

- Interogariile SQL pot fi
  - Pentru definirea datelor, crearea programatica de baze de date, tabele, coloane etc.
    - mai putin utilizate in majoritatea aplicatiilor
    - ALTER, CREATE, DROP, RENAME
  - Pentru manipularea datelor
    - SELECT, INSERT, UPDATE, REPLACE etc.
  - Pentru control/administrare tranzactii/server
- De cele mai multe ori aplicatiile doar manipuleaza datele. Structura este definita in avans de asemenea si administrarea este mai facila cu programe specializate
- Urmatoarele definitii sunt cele valabile pentru **MySql 5.0/MariaDB 5.1**

# MariaDB

- MariaDB 5.1 = MySQL 5.1
- ...
- MariaDB 5.5 = MySQL 5.5
- MariaDB 10 ~ (≠) MySQL 5.6
- Current MariaDB 10.1 (MySQL 5.7)

# ALTER DATABASE

- ALTER {DATABASE | SCHEMA} [db\_name] alter\_specification ...
  - alter\_specification:
    - [DEFAULT] CHARACTER SET [=] charset\_name
    - [DEFAULT] COLLATE [=] collation\_name
- Modifica caracteristicile generale ale unei baze de date
- E necesar dreptul de acces (privilegiu) ALTER asupra respectivei baze de date

# ALTER TABLE

- ALTER TABLE {table\_option [, table\_option] ... | partitioning\_specification}
  - table\_option:
    - ADD [COLUMN] col\_name column\_definition [FIRST | AFTER col\_name ]
    - ADD {INDEX|KEY} [index\_name] [index\_type] (index\_col\_name,...) [index\_option] ...
    - ADD [CONSTRAINT [symbol]] PRIMARY KEY [index\_type] (index\_col\_name,...) [index\_option]
    - ...
    - CHANGE [COLUMN] old\_col\_name new\_col\_name column\_definition [FIRST|AFTER col\_name]
    - MODIFY [COLUMN] col\_name column\_definition [FIRST | AFTER col\_name]
    - DROP [COLUMN] col\_name
    - DROP PRIMARY KEY
    - DROP {INDEX|KEY} index\_name
    - DISABLE KEYS
    - ENABLE KEYS
    - RENAME [TO] new\_tbl\_name
- permite modificarea unui tabel existent

# CREATE DATABASE

- CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] db\_name [create\_specification...]
  - create\_specification:
    - [DEFAULT] CHARACTER SET charset\_name
    - [DEFAULT] COLLATE collation\_name
- Crearea unei noi baze de date
- Necesara la instalarea unei aplicatii
- Fisierile SQL "backup" contin succesiunea DROP..., CREATE... pentru a inlocui datele in intregime

# CREATE INDEX

- `CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX index_name [USING index_type] ON tbl_name (index_col_name,...)`
  - `index_col_name:`
    - `col_name [(length)] [ASC | DESC]`
- Crearea unui index se face de obicei la crearea tabelului
- Interogarea `CREATE INDEX ...` se transpune in interogare `ALTER TABLE ...`



# CREATE TABLE

- CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl\_name [(create\_definition,...)] [table\_options] [select\_statement]
- CREATE [TEMPORARY] TABLE [IF NOT EXISTS] tbl\_name [( ) LIKE old\_tbl\_name ( )]
- Interogarea de creare a tabelului este memorata intern de server-ul MySql pentru utilizari ulterioare (in general in ALTER TABLE sa fie cunoscute specificatiile initiale)

# CREATE TABLE

- `create_definition` – coloana impreuna cu eventualele caracteristici (in special chei - indecsi):
  - `column_definition`
    - | [CONSTRAINT [symbol]] PRIMARY KEY [index\_type] (index\_col\_name,...)
    - | KEY [index\_name] [index\_type] (index\_col\_name,...)
    - | INDEX [index\_name] [index\_type] (index\_col\_name,...)
    - | [CONSTRAINT [symbol]] UNIQUE [INDEX] [index\_name] [index\_type] (index\_col\_name,...)
    - | [FULLTEXT|SPATIAL] [INDEX] [index\_name] (index\_col\_name,...)
    - | [CONSTRAINT [symbol]] FOREIGN KEY [index\_name] (index\_col\_name,...) [reference\_definition]
    - | CHECK (expr)
- `column_definition` – nume si tipul de date (curs 8):
  - `col_name type [NOT NULL | NULL] [DEFAULT default_value] [AUTO_INCREMENT] [UNIQUE [KEY] | [PRIMARY] KEY] [COMMENT 'string'] [reference_definition]`

# CREATE TABLE

- Exemple
  - CREATE TABLE test (a INT NOT NULL AUTO\_INCREMENT, PRIMARY KEY (a), KEY(b)) SELECT b,c FROM test2;
  - CREATE TABLE IF NOT EXISTS `schema`.`Employee` (  
`idEmployee` VARCHAR(45) NOT NULL,  
`Name` VARCHAR(255) NULL,  
`idAddresses` VARCHAR(45) NULL,  
PRIMARY KEY (`idEmployee`),  
CONSTRAINT `fkEmployee\_Addresses`  
FOREIGN KEY `fkEmployee\_Addresses` (`idAddresses`)  
REFERENCES `schema`.`Addresses` (`idAddresses`)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION)  
ENGINE = InnoDB  
DEFAULT CHARACTER SET = utf8  
COLLATE = utf8\_bin

# CREATE TABLE

- `CREATE ... LIKE ...` creaza un tabel fara date pe baza modelului unui tabel existent. Se pastreaza definitiile coloanelor si eventualele chei (index) definite in tabelul anterior
- `CREATE ... SELECT ...` creaza un tabel cu date pe baza modelului si datelor obtinute dintr-un alt tabel existent. Sunt obtinute anumite coloane (`SELECT`) cu tipul lor, dar fara crearea indecsilor
- `CREATE TEMPORARY TABLE` creaza un tabel temporar. Utilizat in cazul interogarilor complexe sau cu numar mare de rezultate

# DROP

- DROP {DATABASE | SCHEMA} [IF EXISTS]  
db\_name
- DROP INDEX index\_name ON tbl\_name
- DROP [TEMPORARY] TABLE [IF EXISTS]  
tbl\_name [, tbl\_name] ...
- Trebuie utilizate cu foarte mare atentie aceste interogari, stergerea datelor este ireversibila
- Fisierile SQL "backup" contin succesiunea DROP..., CREATE... pentru a inlocui datele in intregime

# Interrogari SQL

# Interogari

- Interogariile SQL pot fi
  - Pentru definirea datelor, crearea programatica de baze de date, tabele, coloane etc.
    - mai putin utilizate in majoritatea aplicatiilor
    - ALTER, CREATE, DROP, RENAME
  - **Pentru manipularea datelor**
    - SELECT, INSERT, UPDATE, REPLACE, DELETE etc.
  - Pentru control/administrare tranzactii/server
- De cele mai multe ori aplicatiile doar manipuleaza datele. Structura este definita in avans de asemenea si administrarea este mai facila cu programe specializate
- Urmatoarele definitii sunt cele valabile pentru **MySQL 5.0/MariaDB 5.1**

# DELETE

- `DELETE [LOW_PRIORITY] [QUICK] [IGNORE]  
FROM table_name [WHERE where_condition]  
[ORDER BY ...] [LIMIT row_count]`
- Sterge linii din tabelul mentionat si returneaza  
numarul de linii sterse
- `[LOW_PRIORITY] [QUICK] [IGNORE]` sunt  
optiuni care instruiesc server-ul sa reactioneze  
diferit de varianta standard
- Exemplu:
  - `DELETE FROM somelog WHERE user = 'jcole'  
ORDER BY timestamp_column LIMIT 1;`



# DELETE

- [WHERE where\_condition] – folosit pentru a selecta liniile care trebuie sterse
  - In absenta conditiei se sterg **toate liniile** din tabel
- [LIMIT row\_count] sterge numai *row\_count* linii dupa care se opreste
  - In general pentru a limita ocuparea server-ului (recrearea indecsilor se face “on the fly”)
  - Operatia se poate repeta pana valoarea returnata e mai mica decat row\_count
- [ORDER BY ...] precizeaza ordinea in care se sterg liniile identificate prin conditie

# INSERT

- INSERT [LOW\_PRIORITY | DELAYED | HIGH\_PRIORITY] [IGNORE] [INTO] tbl\_name [(col\_name,...)] **VALUES** ({expr | DEFAULT},...),(...),... [ON DUPLICATE KEY UPDATE col\_name=expr, ... ]
- INSERT [LOW\_PRIORITY | DELAYED | HIGH\_PRIORITY] [IGNORE] [INTO] tbl\_name **SET** col\_name={expr | DEFAULT}, ... [ON DUPLICATE KEY UPDATE col\_name=expr, ... ]
- INSERT [LOW\_PRIORITY | HIGH\_PRIORITY] [IGNORE] [INTO] tbl\_name [(col\_name,...)] **SELECT** ... [ ON DUPLICATE KEY UPDATE col\_name=expr, ... ]

# INSERT

- Introduce linii noi intr-un tabel
- Primele doua forme introduc valori exprimate explicit
  - INSERT ... VALUES ...
  - INSERT ... SET ...
- INSERT ... SELECT ... introduce valori rezultate obtinute printr-o interogare SQL
- DELAYED – interogarea primeste raspuns de la server imediat, dar inserarea datelor se face efectiv cand tabelul implicat nu este folosit
  - valabil pentru metodele de stocare MyISAM, Memory, Archive

# INSERT

- Exemple
  - `INSERT INTO tbl_name (a,b,c) VALUES (1,2,3), (4,5,6), (7,8,9);`
  - `INSERT INTO tbl_name (col1,col2) VALUES (15,col1*2);`
  - `INSERT INTO table1 (field1,field3,field9) SELECT field3,field1,field4 FROM table2;`

# INSERT

- INSERT ... ON DUPLICATE KEY UPDATE ...
- Daca inserarea unei noi linii ar conduce la duplicarea unei chei primare sau unice, in loc sa se introduca o noua linie se modifica linia anterioara
- Exemple
  - INSERT INTO table (a,b,c) VALUES (1,2,3) ON DUPLICATE KEY UPDATE c=c+1;
  - INSERT INTO table (a,b,c) VALUES (1,2,3),(4,5,6) ON DUPLICATE KEY UPDATE c=VALUES(a)+VALUES(b);

# REPLACE

- REPLACE [LOW\_PRIORITY | DELAYED] [INTO] tbl\_name [(col\_name,...)] **VALUES** ({expr | DEFAULT},...),(...),...
- REPLACE [LOW\_PRIORITY | DELAYED] [INTO] tbl\_name **SET** col\_name={expr | DEFAULT}, ...
- REPLACE [LOW\_PRIORITY | DELAYED] [INTO] tbl\_name [(col\_name,...)] **SELECT** ...
- REPLACE functioneaza similar cu INSERT
  - daca noua linie nu realizeaza duplicarea unei chei primare sau unice se realizeaza insertie
  - daca noua linie realizeaza duplicarea unei chei primare sau unice se sterge linia anterioara dupa care se insereaza noua linie
- REPLACE e extensie MySql a limbajului SQL standard

# UPDATE

- UPDATE [LOW\_PRIORITY] [IGNORE] tbl\_name SET col\_name1=expr1 [, col\_name2=expr2 ...] [WHERE where\_condition] [ORDER BY ...] [LIMIT row\_count]
- Modificarea valorilor stocate intr-o linie
- Exemple
  - UPDATE persondata SET age=15 WHERE id=6;
  - UPDATE persondata SET age=age+1;

# SELECT

- SELECT [ALL | DISTINCT | DISTINCTROW ]  
[HIGH\_PRIORITY] [STRAIGHT\_JOIN]  
select\_expr, ... [FROM table\_references
  - [WHERE where\_condition]
  - [GROUP BY {col\_name | expr | position} [ASC | DESC],  
... [WITH ROLLUP]]
  - [HAVING where\_condition]
  - [ORDER BY {col\_name | expr | position} [ASC | DESC],  
...]
  - [LIMIT {[offset,] row\_count | row\_count OFFSET  
offset}]
- ]



# SELECT

- SELECT este **cea mai importanta** interogare SQL.
- Intelegerea setarilor si utilizarea inteligenta a indecsilor stau la baza eficientei unei aplicatii
- E absolut necesara realizarea interogarii in asa fel incat datele returnate sa fie exact cele dorite (prelucrarea sa se realizeze pe server-ul MySql)

# SELECT

- `select_expr`: macar o expresie selectata trebuie sa apara
  - identifica ceea ce trebuie extras ca valori de iesire din baza de date
  - pot fi nume de coloana(e)
  - pot fi date de sinteza (rezultate din utilizarea unor functii MySql) – necesara atribuirea unui Alias
    - `SELECT CONCAT(last_name,', ',first_name) AS full_name FROM mytable ORDER BY full_name;`

# SELECT

- WHERE where\_condition, HAVING where\_condition sunt utilizate pentru a introduce criterii de selectie
  - in general au comportare similara si sunt interschimbabile
  - WHERE accepta orice operatori mai putin functii aggregate – de “sumare” (COUNT, MAX)
  - HAVING accepta functii aggregate, dar se aplica la sfarsit, exact inainte de a fi trimise datele clientului, **fara nici o optimizare** – utilizarea este recomandata doar cand nu exista echivalent WHERE

# SELECT

- ORDER BY {col\_name | expr | position} [ASC | DESC]
  - ordoneaza datele returnate dupa anumite criterii (valoarea unei anumite coloane sau functii).
    - Implicit ordonarea este crescatoare ASC, dar se poate specifica ordine descrescatoare DESC
- GROUP BY {col\_name | expr | position}
  - realizeaza gruparea liniilor returnate dupa anumite criterii
  - permite utilizarea functiilor agregate (de sumare)

# SELECT

- GROUP BY – functii aggregate
  - AVG(expresie) – mediere valorilor
    - SELECT student\_name, AVG(test\_score) FROM student GROUP BY student\_name;
  - COUNT(expresie), COUNT(\*)
    - SELECT COUNT(\*) FROM student;
    - SELECT COUNT(DISTINCT results) FROM student;
    - SELECT student.student\_name, COUNT(\*) FROM student.course WHERE student.student\_id=course.student\_id GROUP BY student\_name;
    - SELECT columnname, COUNT(columnname) FROM tablename GROUP BY columnname HAVING COUNT(columnname)>1
- Cuvantul cheie DISTINCT este utilizat pentru a procesa doar liniile cu valori diferite
  - exemplu: 100 de note (rezultate) la examen
    - COUNT(results) va oferi raspunsul 100
    - COUNT(DISTINCT results) va oferi raspunsul 7 (notele diferite 4,5,6,7,8,9,10)

# SELECT

- GROUP BY – functii aggregate
  - MIN(expresie), MAX(expresie) – minim si maxim
    - SELECT student\_name, MIN(test\_score), MAX(test\_score) FROM student GROUP BY student\_name;
  - SUM(expresie) – sumarea valorilor
    - SELECT year, SUM(profit) FROM sales GROUP BY year;
- WITH ROLLUP – operatii de sumare super-aggregate (un nivel suplimentar de agregare)

# SELECT ... WITH ROLLUP

- `SELECT year, SUM(profit) FROM sales GROUP BY year;`
- `SELECT year, SUM(profit) FROM sales GROUP BY year WITH ROLLUP;`
  - se obtine un total general, linia "super-aggregate" este identificata dupa valoarea NULL a coloanei dupa care se face sumarea

year	SUM(profit)
2000	4525
2001	3010

year	SUM(profit)
2000	4525
2001	3010
NULL	7535

# SELECT

- LIMIT [offset,] row\_count | row\_count
  - se limiteaza numarul de linii returnate
  - utilizat frecvent in aplicatiile web
  - LIMIT 15 – returneaza doar primele 15 linii (1÷15)
  - LIMIT 10,15 – returneaza 15 linii dupa primele 10 linii (11÷25)



# JOIN

- Normalizarea si existenta relatiilor intre diversele tabele ale unei baze de date implica faptul ca pentru aflarea unor informatii utilizabile (complete), acestea trebuie extrase **simultan** din mai multe tabele
  - informatie inutilizabila: studentul cu id-ul 253 a luat nota 8 la examenul cu id-ul 35
- Uneori asamblarea informatiilor din mai multe tabele e necesara pentru obtinerea unor rapoarte complexe
  - Exemplu: tabel cu clienti, tabel cu comenzi, tabel cu produse; legatura produse-comenzi e implementata printr-un tabel suplimentar. Raspunsul la intrebarea cate produse x a cumparat clientul y cere tratarea unitara a celor 4 tabele implicate

# JOIN

- In general in SQL se poate descrie o astfel de unificare de date intre doua tabele:
  - left\_table JOIN\_type right\_table criteriu\_unificare
- JOIN\_type
  - JOIN – selecteaza toate liniile compuse in care criteriul este indeplinit pentru ambele tabele
  - LEFT JOIN – compune si selecteaza toate liniile din left\_table chiar daca nu este gasit un corespondent in right\_table
  - RIGHT JOIN – compune si selecteaza toate liniile din right table (similar)
  - FULL JOIN – compune si selecteaza toate liniile din left\_table si right\_table fie ca este indeplinit criteriul fie ca nu (nu este implementat in MySql, poate fi simulat)

# JOIN

- Clauza JOIN e utilizata pentru a realiza o unificare temporara, dupa anumite criterii, din punct de vedere logic, a doua tabele in vederea extragerii informatiei "suma" dorite
  - left\_table [INNER | CROSS] JOIN right\_table [join\_condition]
  - left\_table STRAIGHT\_JOIN right\_table
  - left\_table STRAIGHT\_JOIN right\_table ON condition
  - left\_table LEFT [OUTER] JOIN right\_table join\_condition
  - left\_table NATURAL [LEFT [OUTER]] JOIN right\_table
  - left\_table RIGHT [OUTER] JOIN right\_table join\_condition
  - left\_table NATURAL [RIGHT [OUTER]] JOIN right\_table
  - join\_condition: ON conditional\_expr | USING (column\_list)

# JOIN – Exemplu

- Tabel clienti
  - 4 clienti
- Tabel comenzi
  - client 1 – 2 comenzi
  - client 2 – 0 comenzi
  - client 3,4 – 1 comanda

```
CREATE TABLE `clienti` (  
  `id_client` int(10) unsigned NOT NULL auto_increment,  
  `nume` varchar(100) NOT NULL,  
  PRIMARY KEY (`id_client`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `clienti` (`id_client`,`nume`) VALUES  
(1,'Ionescu'),  
(2,'Popescu'),  
(3,'Vasilescu'),  
(4,'Georgescu');
```

```
CREATE TABLE `comenzi` (  
  `id_comanda` int(10) unsigned NOT NULL auto_increment,  
  `id_client` int(10) unsigned NOT NULL,  
  `suma` double NOT NULL,  
  PRIMARY KEY (`id_comanda`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `comenzi` (`id_comanda`,`id_client`,`suma`) VALUES  
(1,1,19.99),  
(2,1,35.15),  
(3,3,17.56),  
(4,4,12.34);
```

# INNER JOIN

- INNER JOIN sunt unificarile implicite, in care criteriul (join\_condition) trebuie indeplinit in ambele tabele (extensie a cuvintului cheie JOIN pentru evitarea ambiguitatii)
  - OUTER JOIN = {LEFT JOIN | RIGHT JOIN | FULL JOIN} – nu e obligatoriu sa fie indeplinit criteriul in ambele tabele
  - FULL JOIN nu e implementat in MySql, poate fi simulat ca UNION intre LEFT JOIN si RIGHT JOIN
- INNER JOIN sunt echivalente cu realizarea produsului cartezian intre cele doua tabele implicate urmata de verificarea criteriului, daca acesta exista

# CROSS JOIN

- In MySql INNER JOIN si CROSS JOIN sunt echivalente in totalitate
  - In SQL standard INNER este folosit in prezenta unui criteriu, CROSS in absenta sa
- INNER (CROSS) JOIN si “,” sunt echivalente cu produsul cartezian intre cele doua tabele implicate in conditile lipsei criteriului de selectie: fiecare linie a unui tabel este alaturata fiecarei linii din al doilea tabel
  - (un tabel cu M linii si A coloane) CROSS JOIN (un tabel cu N linii si B coloane) → (un tabel cu MxN linii si A+B coloane)

# CROSS JOIN

SQL Query Area

```
1 SELECT * FROM clienti JOIN comenzi;  
2 SELECT * FROM clienti, comenzi;  
3 SELECT * FROM clienti INNER JOIN comenzi;  
4 SELECT * FROM clienti CROSS JOIN comenzi;
```

id_client	nume	id_comanda	id_client	suma
1	Ionescu	1	1	19.99
2	Popescu	1	1	19.99
3	Vasilescu	1	1	19.99
4	Georgescu	1	1	19.99
1	Ionescu	2	1	35.15
2	Popescu	2	1	35.15
3	Vasilescu	2	1	35.15
4	Georgescu	2	1	35.15
1	Ionescu	3	3	17.56
2	Popescu	3	3	17.56
3	Vasilescu	3	3	17.56
4	Georgescu	3	3	17.56
1	Ionescu	4	4	12.34
2	Popescu	4	4	12.34
3	Vasilescu	4	4	12.34
4	Georgescu	4	4	12.34

# INNER JOIN – criterii

- USING – trebuie sa aiba o coloana cu nume identic in cele doua tabele
  - coloana comuna este afisata o singura data
- ON – accepta orice conditie conditionala
  - chiar daca numele coloanelor din conditie sunt identice, sunt tratate ca entitati diferite (id\_client apare de doua ori provenind din cele doua tabele)

SQL Query Area				
1	<code>SELECT * FROM clienti INNER JOIN comenzi USING (id_client);</code>			
id_client	nume	id_comanda	suma	
1	Ionescu	1	19.99	
1	Ionescu	2	35.15	
3	Vasilescu	3	17.56	
4	Georgescu	4	12.34	
1	<code>SELECT * FROM clienti INNER JOIN comenzi ON (clienti.id_client=comenzi.id_client);</code>			
id_client	nume	id_comanda	id_client	suma
1	Ionescu	1	1	19.99
1	Ionescu	2	1	35.15
3	Vasilescu	3	3	17.56
4	Georgescu	4	4	12.34



# NATURAL JOIN

- NATURAL JOIN e echivalent cu o unificare INNER JOIN cu o clauza USING(...) care utilizeaza toate coloanele cu nume comun intre cele doua tabele

SQL Query Area			
1	<code>SELECT * FROM clienti NATURAL JOIN comenzi;</code>		
id_client	nume	id_comanda	suma
1	Ionescu	1	19.99
1	Ionescu	2	35.15
3	Vasilescu	3	17.56
4	Georgescu	4	12.34

# LEFT JOIN

- Unificare de tip OUTER JOIN
- Se returneaza linia din left\_table chiar daca nu exista corespondent in right\_table (se introduc valori NULL)
- Cuvantul cheie OUTER este optional

SQL Query Area

```
1 | SELECT * FROM clienti LEFT OUTER JOIN comenzi USING(id_client);
```

id_client	nume	id_comanda	suma
1	Ionescu	1	19.99
1	Ionescu	2	35.15
2	Popescu	NULL	NULL
3	Vasilescu	3	17.56
4	Georgescu	4	12.34

# RIGHT JOIN

- Unificare de tip OUTER JOIN
- Se returneaza linia din right\_table chiar daca nu exista corespondent in left\_table
- Echivalent cu LEFT JOIN cu tabelele scrise in ordine inversa

SQL Query Area				
1 SELECT * FROM clienti RIGHT OUTER JOIN comenzi USING(id_client);				
id_client	id_comanda	suma	nume	
1	1	19.99	Ionescu	
1	2	35.15	Ionescu	
3	3	17.56	Vasilescu	
4	4	12.34	Georgescu	

SQL Query Area				
1 SELECT * FROM comenzi RIGHT OUTER JOIN clienti USING(id_client);				
id_client	nume	id_comanda	suma	
1	Ionescu	1	19.99	
1	Ionescu	2	35.15	
2	Popescu	NULL	NULL	
3	Vasilescu	3	17.56	
4	Georgescu	4	12.34	

# JOIN

- STRAIGHT\_JOIN – forteaza citirea mai intai a valorilor din left\_table si apoi a celor din right\_table (in anumite cazuri citirea se realizeaza invers)
- USE\_INDEX, IGNORE\_INDEX, FORCE\_INDEX controlul index-ului utilizat pentru gasirea si selectia liniilor, poate aduce spor de viteza

# UNION

- Combina rezultatele mai multor interogari SELECT intr-un singur rezultat general
- SELECT ... UNION [ALL | DISTINCT] SELECT ... [UNION [ALL | DISTINCT] SELECT ...]
- Poate fi folosit pentru a realiza FULL JOIN

```
SQL Query Area
1 SELECT * FROM comenzi LEFT JOIN clienti ON (comenzi.id_client=clienti.id_client)
2 UNION
3 SELECT * FROM comenzi RIGHT JOIN clienti ON (comenzi.id_client=clienti.id_client)
4 WHERE comenzi.id_client IS NULL
```

id_comanda	id_client	suma	id_client	nume
1	1	19.99	1	Ionescu
2	1	35.15	1	Ionescu
3	3	17.56	3	Vasilescu
4	4	12.34	4	Georgescu
NULL	NULL	NULL	2	Popescu

# Subquery

- O “subinterogare” este o interogare de tip SELECT utilizata ca operand intr-o alta interogare
- O “subinterogare” poate fi privit ca un tabel temporar si tratat ca atare (inclusiv cu JOIN) eventual cu atribuire de nume (Alias) daca este nevoie
- Exemple
  - `SELECT * FROM t1 WHERE column1 = (SELECT column1 FROM t2);`

# Subquery

- Subquery – un instrument foarte puternic
- permite selectii in doua sau mai multe etape
  - o prima selectie **dupa un criteriu**
  - urmata de o doua selectie **dupa un alt criteriu** in **rezultatele primei selectii**
  - ... samd
- Exista restrictii asupra tabelelor implicate pentru evitarea prelucrarilor recursive (bucle potential infinite)
  - ex: UPDATE tabel<sub>1</sub> SET ... SELECT ... FROM tabel<sub>1</sub> nu este permis

# Subquery

- Subquery – un instrument foarte puternic
- Permite evitarea multor prelucrari PHP si trimiterea lor spre server-ul MySql
  - `INSERT INTO tabel1 ... SELECT ... FROM tabel2` permite inserarea printr-o singura interogare a mai multor linii in tabel1 (in functie de numarul de linii rezultate din tabel2)



# Laborator 2 / 2011-2012

- Se recomanda aplicarea exercitiilor din laboratorul 2 / 2011-2012, pentru exemple de interogari, JOIN, subquery, JOIN cu subquery

Activitate suplimentara

# Solutie

# Activitate suplimentara

- Aplicatia folosita la curs / laborator nu este optima din motivele anuntate in cursul 9
- De asemenea este incompleta
  - o cerinta obligatorie intr-o aplicatie reala dar neacoperita in exemplu este **verificarea** datelor introduse
    - pe browser – Javascript
    - pe server dupa primirea datelor – PHP
  - se pot gasi usor combinatii de date introduse care sa duca la incompatibilitati browser-PHP-MySQL

# Greseala de logica

- Provine de la citirea initiala a intregii baze de date intr-o matrice in "antet.php" care apoi este folosita de celelalte fisiere.

```
$query = "SELECT p.*, c.`nume` AS `nume_categ` FROM `produse` AS p  
LEFT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`)";  
$result = mysql_query($query, $conex) or die(mysql_error());  
$row_result = mysql_fetch_assoc($result);  
$totalRows_result = mysql_num_rows($result);  
  
do {  
    $produse[$row_result['nume_categ']][$row_result['nume']] = array ("descr" =>  
$row_result['detalii'], "pret" => $row_result['pret'], "cant" => $row_result['cant']);  
}  
while ($row_result = mysql_fetch_assoc($result));
```

# Greseala de logica

- se manifesta la introducerea unei noi categorii

SQL Query Area

```
1 SELECT * FROM categorii c;
```

id_categ	nume	detalii
1	papetarie	NULL
2	instrumente	NULL
3	audio-video	NULL
4	jucarii	pentru copii

/ Area

```
SELECT * FROM produse p;
```

id_produc	id_categ	nume	detalii	cant	pret
1	1	carte	mai multe pagini scrise legate	0	100
2	1	caiet	mai multe pagini goale legate	0	75
3	1	hartie scris a	mai multe pagini goale NElegate	5	55
4	2	penar	loc de depozitat instrumente de scris	0	150
5	2	stilou	instrument de scris albastru	0	125
6	2	creion	instrument de scris gri	0	25
7	3	cd	canta	0	50
8	3	dvd	vizual	0	100
9	3	blue ray	vizual extrem	0	500
10	3	video cd	cd cu material video	10	75

# Greseala de logica

- Introducerea unei noi categorii se traduce prin aparitia unei linii noi in tabelul categorii ("jucarii"), fara produse asociate in tabelul de produse
- Realizarea unei selectii si uniri de tip "LEFT JOIN" are ca efect ignorarea categoriei vide in rezultat, ca urmare categoria "jucarii" nu va aparea in matricea **\$produse** pe care se bazeaza aplicatia mai departe

# LEFT JOIN

SQL Query Area

```
1 SELECT p.*, c.`nume` AS `nume_categ` FROM `produse` AS p  
2 LEFT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`)
```

id_produș	id_categ	nume	detalii	cant	pret	nume_categ
1	1	carte	mai multe pagini scrise legate	0	100	papetarie
2	1	caiet	mai multe pagini goale legate	0	75	papetarie
3	1	hartie scris a	mai multe pagini goale NElegate	5	55	papetarie
4	2	penar	loc de depozitat instrumente de scris	0	50	instrumente
5	2	stilou	instrument de scris albastru	0	125	instrumente
6	2	creion	instrument de scris gri	0	25	instrumente
7	3	cd	canta	0	50	audio-video
8	3	dvd	vizual	0	100	audio-video
9	3	blue ray	vizual extrem	0	500	audio-video
10	3	video cd	cd cu material video	10	75	audio-video

# Greseala de logica

- Urmarea ar fi ca o categorie vida nu va mai putea fi populata cu produse si nici macar afisata pentru ca nu se regaseste in rezultat
- In codul utilizat acest lucru este partial corectat prin modificarea matricii `$produse` la introducerea unei noi categorii
  - `$produse[$_POST["nou_nume"]]=array();`
- Aceasta corectie are doar efect temporar
  - in noua categorie se pot introduce produse doar la pasul imediat urmator
  - daca se introduce macar un singur produs la pasul urmator aplicatia **pare** sa functioneze corect
  - acest lucru mascheaza functionarea gresita deoarece utilizarea tipica este:
    - categorie noua → produs in acea categorie → **pare** ca functioneaza corect



# Greseala de logica

- corectie "temporara" in "admin\_categ.php"
- are efect doar in functionarea in continuare a scriptului, **imediat** dupa introducerea categoriei
  - se afiseaza noua categorie
  - se pot introduce produse

```
if (isset($_POST["c_nou"]))
    //categorie noua introdusa
    $query = "INSERT INTO `categorii` (`nume`, `detalii`) VALUES
('".$_POST["nou_nume"]."', '".$_POST["nou_desc"]."')";
    echo $query;//util in perioada de testare
    $result = mysql_query($query, $conex) or die(mysql_error());
    $record=mysql_insert_id();//obtinerea id-ului nou
    $produse[$_POST["nou_nume"]]=array(); // update matrice produse
    echo "<p>Categoria '".$_POST["nou_nume"]."' adaugata! Are id = ".$record."</p>";
}
```

# Corectii

- Corect: asa cum e anuntat in cursul 9
  - Citirea datelor in fiecare fisier in parte
  - Citirea numai a datelor necesare
- In acest mod "admin\_categ.php" va citi date doar din tabelul "categorii" pentru afisarea listei, cu identificarea tuturor categoriilor, inclusiv a celor vide
- Numararea produselor din fiecare categorie se poate face (si mult mai eficient) prin utilizarea functiei "aggregate" COUNT in MySql

# Minimal - RIGHT JOIN

- Alternativa minimala, pe scheletul existent, compatibil in urma (se repeta - **ineficient**)
- Utilizarea unei selectii RIGHT JOIN care permite evidentierea liniilor din tabelul categorii fara corespondent in tabelul produse

SQL Query Area

```
1 SELECT p.*, c.`nume` AS `nume_categ` FROM `produse` AS p
2 RIGHT JOIN `categorii` AS c ON (c.`id_categ` = p.`id_categ`)
```

id_produș	id_categ	nume	detalii	cant	pret	nume_categ
1	1	carte	mai multe pagini scrise legate	0	100	papetarie
2	1	caiet	mai multe pagini goale legate	0	75	papetarie
3	1	hartie scris a	mai multe pagini goale NElegate	5	55	papetarie
4	2	penar	loc de depozitat instrumente de scris	0	150	instrumente
5	2	stilou	instrument de scris albastru	0	125	instrumente
6	2	creion	instrument de scris gri	0	25	instrumente
7	3	cd	canta	0	50	audio-video
8	3	dvd	vizual	0	100	audio-video
9	3	blue ray	vizual extrem	0	500	audio-video
10	3	video cu	cd cu material video	10	75	audio-video
NULL	NULL	NULL	NULL	NULL	NULL	jucarii

# Minimal - RIGHT JOIN

- Linia cu valori NULL in dreptul produsului poate avea efecte neplacute in aplicatie
  - apare un produs fictiv in categoria "jucarii" cu valori nule
  - acest produs nu poate fi modificat dar se rezolva dupa introducerea unui alt produs in categoria "jucarii"

## Magazin online Firma X SRL

### Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	<a href="#">Papetarie</a>	3
2	<a href="#">Instrumente</a>	3
3	<a href="#">Audio-video</a>	4
4	<a href="#">Jucarii</a>	1

Total produse: 11

## Magazin online Firma X SRL

### Lista produse in categoria Jucarii

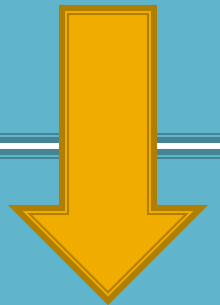
Nr.	Produs	Descriere	Pret	Cantitate	Actiuni
1					<a href="#">modifica</a>
-	Produs nou				<a href="#">adauga</a>

# Minimal - RIGHT JOIN

- Rezolvarea minimala (**ineficienta**) e completa cu introducerea unui test pentru identificarea liniilor nule

```
do {  
    $produce[$row_result['nume_categ']][$row_result['nume']]=array ("descr" =>  
$row_result['detalii'], "pret" => $row_result['pret'], "cant" => $row_result['cant']);  
}  
while ($row_result = mysql_fetch_assoc($result));
```

```
do {  
    if (empty($row_result['nume']))  
        $produce[$row_result['nume_categ']]=array();  
    else  
        $produce[$row_result['nume_categ']][$row_result['nume']]=array ("descr" =>  
$row_result['detalii'], "pret" => $row_result['pret'], "cant" => $row_result['cant']);  
}  
while ($row_result = mysql_fetch_assoc($result));
```



# Minimal - RIGHT JOIN

## Magazin online Firma X SRL

### Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	<a href="#">Papetarie</a>	3
2	<a href="#">Instrumente</a>	3
3	<a href="#">Audio-video</a>	4
4	<a href="#">Jucarii</a>	1
5	<a href="#">Test</a>	0

Total produse: 11

## Magazin online Firma X SRL

### Lista produse in categoria Test

Nr.	Produs	Descriere	Pret	Cantitate	Actiuni
		Produs nou			<a href="#">adauga</a>

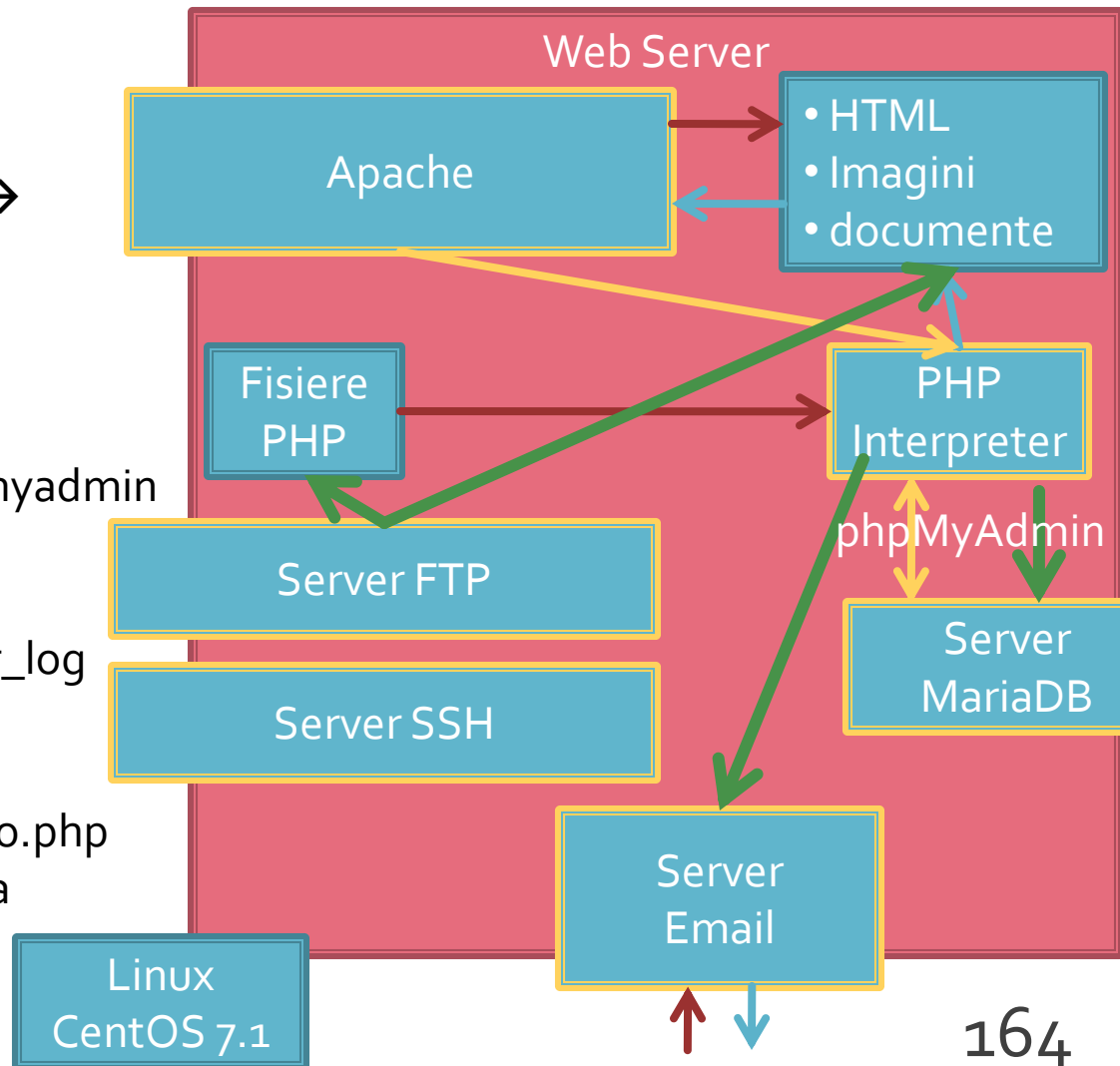
MySql – Server Centos 7.1

# Mini – Indrumar practic

## Lucru cu bazele de date

# Utilizare LAMP

1. login → root:masterrc
2. ifconfig → 192.168.30.5
3. putty.exe → 192.168.30.5 → SSH → root:masterrc (remote login)
4. [alte comenzi linux dorite]
5. FTP → Winscp → SFTP → student:masterrc@192.168.30.5
6. MySql → http://192.168.30.5/phpmyadmin → root:masterrc
7. Apache Error Log →
  - 7a. putty → nano /var/log/httpd/error\_log
  - 7b. http://192.168.30.5/logfile.php (nonstandard)
8. PHP info → http://192.168.30.5/info.php
9. daca serviciul DHCP duce la oprirea Apache: `service httpd restart`





# PhpMyAdmin

- <http://192.168.30.5/phpmyadmin>
  - root
  - parola administrator **MySql/MariaDB** (masterrc)



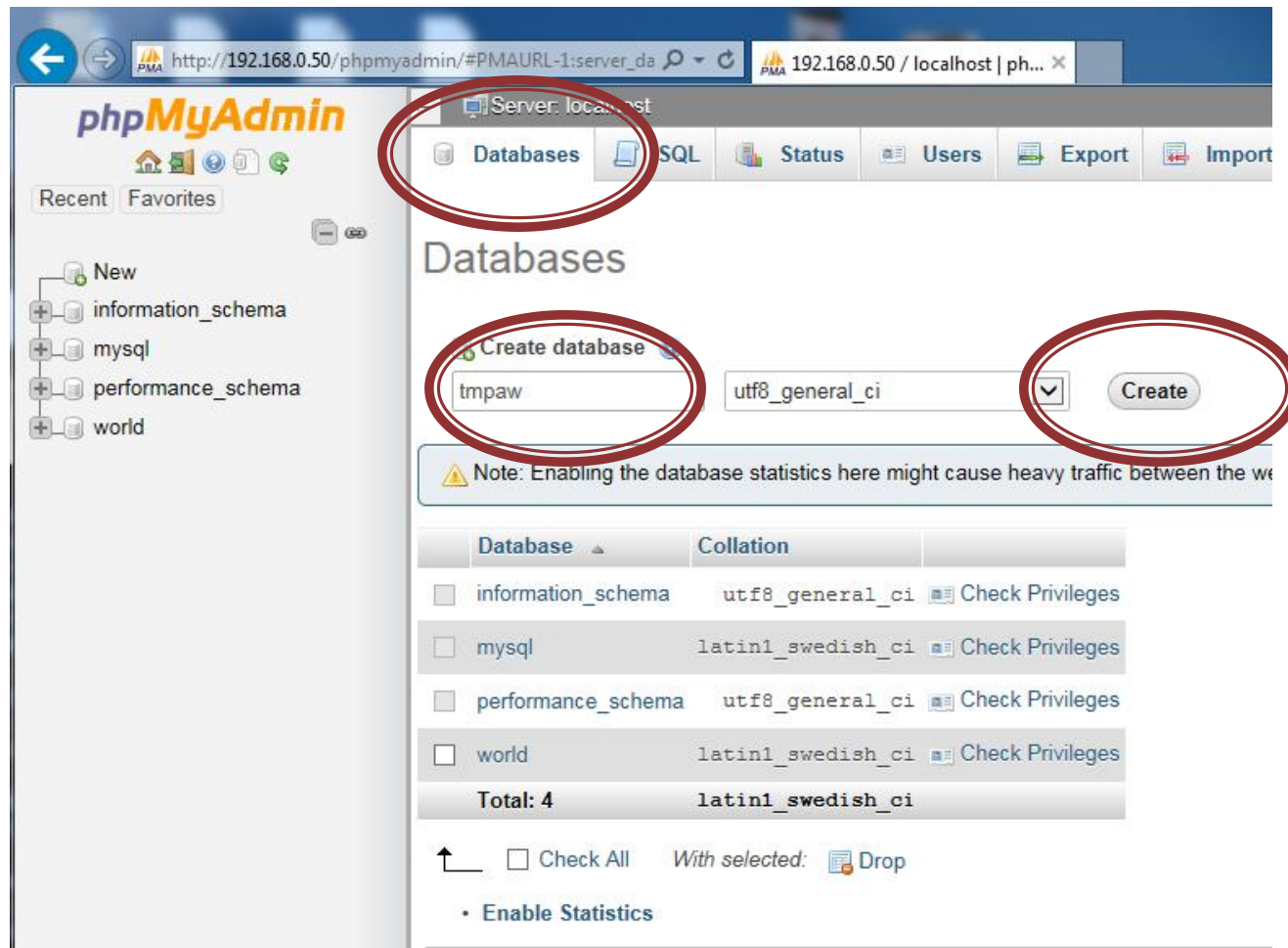
# PhpMyAdmin

The screenshot displays the PhpMyAdmin web interface in a browser window. The address bar shows the URL `http://192.168.0.50/phpmyadmin/#PMAURL-0:index.php` and the server name `192.168.0.50 / localhost | ph...`. The interface includes a navigation menu on the left with options like 'Recent' and 'Favorites', and a main content area with several panels:

- General Settings:** Includes a 'Change password' link and a 'Server connection collation' dropdown menu set to `utf8mb4_unicode_ci`.
- Appearance Settings:** Includes a 'Language' dropdown set to 'English', a 'Theme' dropdown set to 'pmahomme', and a 'Font size' dropdown set to '82%'. A 'More settings' link is also present.
- Database server:** Lists server details:
  - Server: Localhost via UNIX socket
  - Server type: MariaDB
  - Server version: 5.5.44-MariaDB - MariaDB Server
  - Protocol version: 10
  - User: root@localhost
  - Server charset: UTF-8 Unicode (utf8)
- Web server:** Lists web server details:
  - Apache/2.4.6 (CentOS) OpenSSL/1.0.1e-fips mod\_fcgid/2.3.9 PHP/5.4.16 mod\_python/3.5.0- Python/2.7.5
  - Database client version: libmysql - 5.5.44-MariaDB
  - PHP extension: mysqli
  - PHP version: 5.4.16
- phpMyAdmin:** Lists version and resource information:
  - Version information: 4.4.15.1
  - Documentation
  - Wiki
  - Official Homepage
  - Contribute
  - Get support
  - List of changes

# Creare Baza de Date

- Databases → "nume" → Create



The screenshot shows the phpMyAdmin interface. The 'Databases' tab is selected and circled in red. Below it, the 'Create database' form is visible, with the database name 'tmpaw' and the collation 'utf8\_general\_ci' entered. The 'Create' button is also circled in red. A table below the form lists existing databases and their collations.

Database	Collation	
<input type="checkbox"/> information_schema	utf8_general_ci	<a href="#">Check Privileges</a>
<input type="checkbox"/> mysql	latin1_swedish_ci	<a href="#">Check Privileges</a>
<input type="checkbox"/> performance_schema	utf8_general_ci	<a href="#">Check Privileges</a>
<input type="checkbox"/> world	latin1_swedish_ci	<a href="#">Check Privileges</a>
<b>Total: 4</b>	<b>latin1_swedish_ci</b>	

↑  Check All With selected: [Drop](#)

• [Enable Statistics](#)

# Creare tabelle in baza de date

- Baza de date (in lista) → Structure → div Create Table → nume/coloane → Go

The screenshot shows the phpMyAdmin web interface. The browser address bar indicates the URL `http://192.168.0.50/php`. The interface is for a server at `localhost` with the database `tmpaw`. The left sidebar shows a tree view of databases, with `tmpaw` selected. The main content area displays a message: "No tables found in database." Below this, the "Create table" option is selected. The "Name" field contains the text `categorii` and the "Number of columns" field contains the number `3`. A "Go" button is located at the bottom right of the form. Red circles highlight the "Structure" button in the top navigation bar, the "Create table" button, the "Name" field, the "Number of columns" field, and the "Go" button.

# Introducere coloane, tabel categorii

- (eventual) Adaugare coloane / Stabilire nume
- Name / Type / Length / Default

The screenshot shows the phpMyAdmin interface for creating a table named 'categorii' in the 'tmpaw' database. The table name 'categorii' is circled in red. The 'Add 1 column(s) Go' button is also circled in red. The table structure is displayed below, with columns 'id\_categ', 'nume', and 'detalii'. The 'id\_categ' column is of type 'INT' and has a 'None' default value, all circled in red. The 'nume' column is of type 'VARCHAR' with a length of '45' and a 'None' default value, with the length '45' circled in red. The 'detalii' column is of type 'VARCHAR' with a length of '150' and a 'None' default value. The interface includes a sidebar with 'Recent' and 'Favorites' sections, and a top navigation bar with 'Browse', 'Structure', 'SQL', 'Search', 'Import', and 'Privileges' buttons.

Name	Type	Length/Values	Default	Collation
id_categ	INT		None	
nume	VARCHAR	45	None	
detalii	VARCHAR	150	None	

# Introducere coloane

- (eventual) NOT NULL / Index / Auto Increment
  - in functie de “necesitatile” coloanei respective

Table name:  Add  column(s)

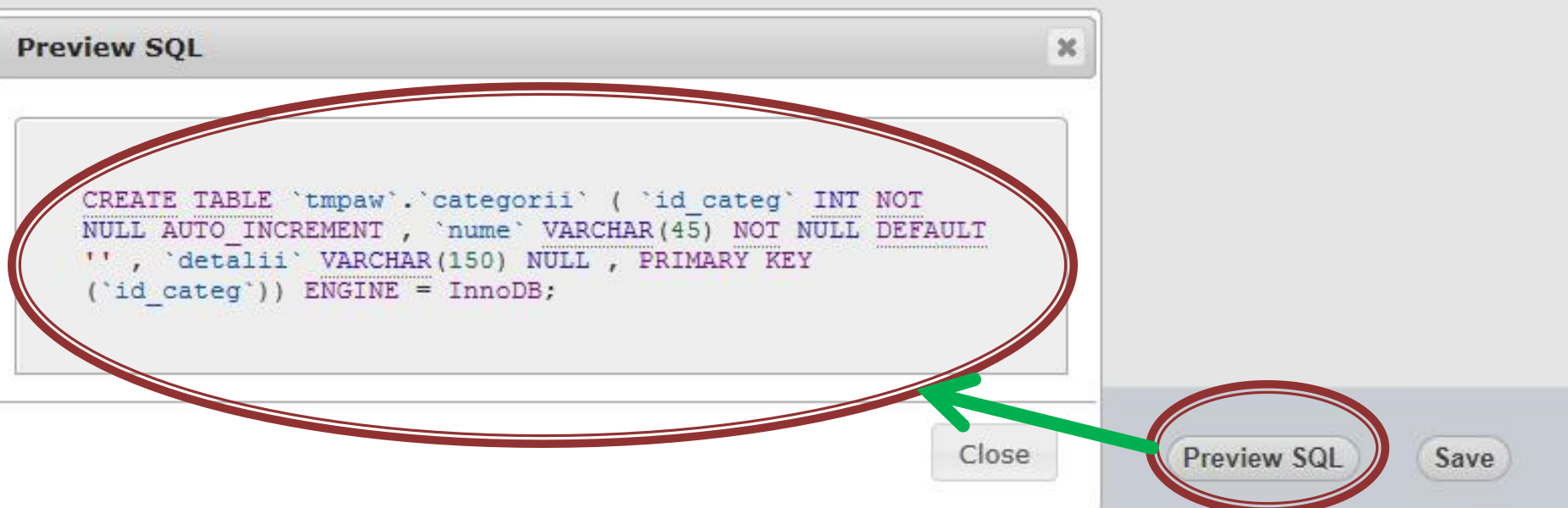
Structure

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	Comments
id_categ	INT		None			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>	
nume	VARCHAR	45	As defined:			<input type="checkbox"/>	---	<input type="checkbox"/>	
detalii	VARCHAR	150	None			<input checked="" type="checkbox"/>	---	<input type="checkbox"/>	



# Preview SQL

- in aproape toate etapele in PhpMyAdmin
  - exemplu de cod SQL/schelet utilizabil (copy/paste) in aplicatia PHP
  - modificari de finete absente din interfata
    - copy → Sectiune "SQL" in interfata → paste → modificare



# Introducere coloane, tabel produse

- New → Nume → Add Columns → ...

The screenshot shows the phpMyAdmin interface for a database named 'tmpaw'. The 'Table name' field is set to 'produse'. The 'Add' button is highlighted with a red circle, and the 'Go' button is also highlighted with a red circle. The 'Structure' tab is active, showing a table with columns: id\_produc, id\_categ, nume, detalii, cant, and pret. The 'Type' dropdown for the 'pret' column is set to 'FLOAT'. The 'New' button in the left sidebar is also highlighted with a red circle.

Name	Type	Length/Values	Default	Collation	Attributes	Null	Index	A_I	C
id_produc	INT		None			<input type="checkbox"/>	PRIMARY	<input checked="" type="checkbox"/>	
id_categ	INT		None			<input type="checkbox"/>	---	<input type="checkbox"/>	
nume	VARCHAR	45	As defined:			<input type="checkbox"/>	---	<input type="checkbox"/>	
detalii	VARCHAR	150	None			<input checked="" type="checkbox"/>	---	<input type="checkbox"/>	
cant	INT		None			<input checked="" type="checkbox"/>	---	<input type="checkbox"/>	
pret	FLOAT		None			<input checked="" type="checkbox"/>	---	<input type="checkbox"/>	



# Introducere date initiale (interfata)

- Tabel → Insert → Completare → Go

The screenshot displays the phpMyAdmin interface for a table named 'categorii' in the 'tmpaw' database. The 'Insert' tab is selected, and the 'id\_categ' column is highlighted. The 'nume' column contains the value 'papetarie'. The 'Go' button is visible at the bottom right. The 'insert as new row' dropdown is set to 'insert as new row', and the 'and then' dropdown is set to 'Go back to previous page'. The 'Continue insertion with' dropdown is set to '1' row.

Column	Type	Function	Null	Value
id_categ	int(11)			
nume	varchar(45)			papetarie
detalii	varchar(150)		☑	

Continue insertion with  row

# Vizualizare date existente

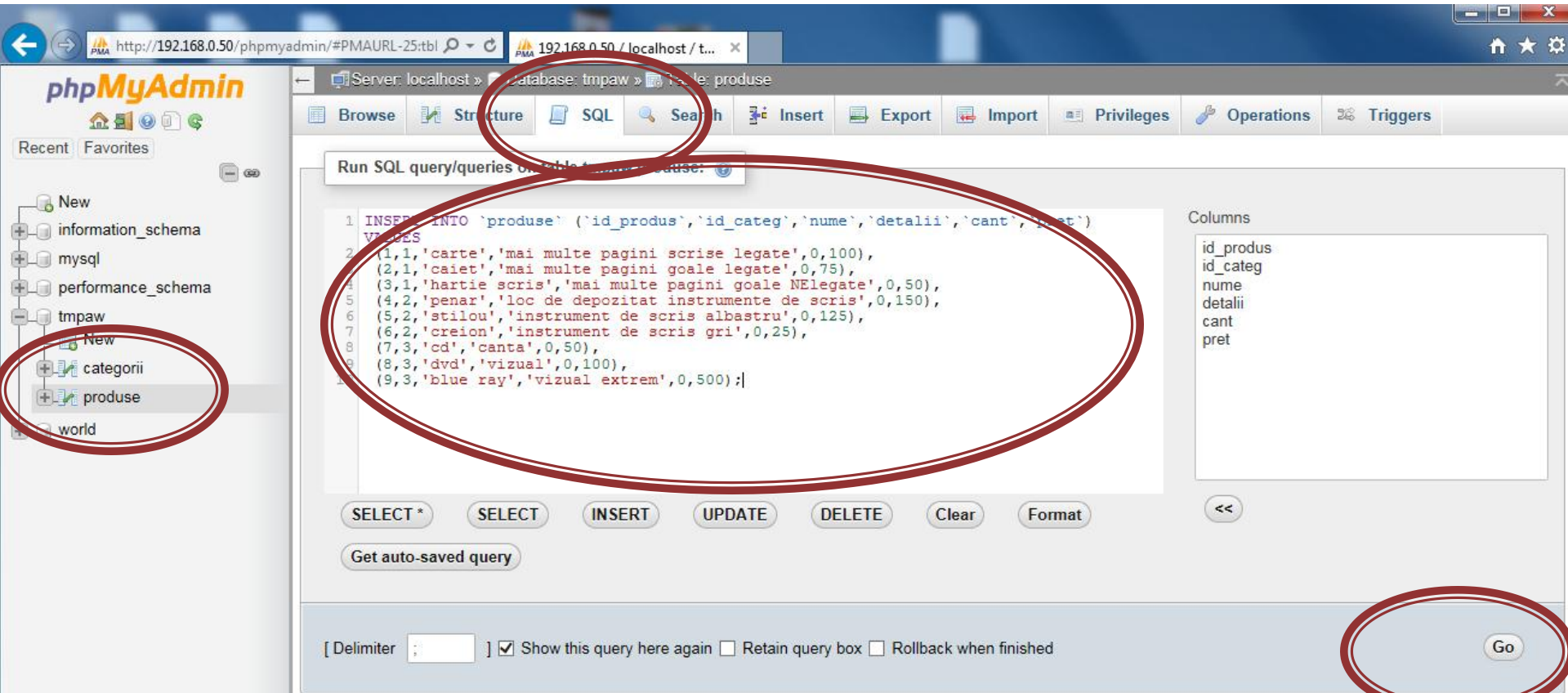
- Tabel → Browse → salt la pagina (numar de linii pe pagina)

The screenshot shows the phpMyAdmin interface for a database named 'tmpaw'. The 'categoriasii' table is selected in the sidebar and is highlighted in the main view. The table contains three rows of data. The 'Browse' button in the top navigation bar is circled in red. The table name 'categoriasii' in the sidebar is also circled in red. The table data is circled in red.

id_categ	nume	detalii
1	papetarie	NULL
2	instrumente	NULL
3	audio-video	NULL

# Introducere date initiale (SQL)

- Tabel → SQL → completare → Go



The screenshot shows the phpMyAdmin interface with the following elements:

- Navigation Panel (Left):** Shows a tree view of databases. The 'tmpaw' database is selected, and the 'produse' table is highlighted.
- SQL Tab:** The 'SQL' tab is active, showing an SQL query for inserting data into the 'produse' table. The query is: 

```
1 INSERT INTO `produse` (`id_produ`  
2 `id_categ`, `nume`, `detalii`, `cant`, `pret`)  
VALUES  
3 (1,1,'carte','mai multe pagini scrise legate',0,100),  
4 (2,1,'caiet','mai multe pagini goale legate',0,75),  
5 (3,1,'hartie scris','mai multe pagini goale NElegate',0,50),  
6 (4,2,'penar','loc de depozitat instrumente de scris',0,150),  
7 (5,2,'stilou','instrument de scris albastru',0,125),  
8 (6,2,'creion','instrument de scris gri',0,25),  
9 (7,3,'cd','canta',0,50),  
10 (8,3,'dvd','vizual',0,100),  
11 (9,3,'blue ray','vizual extrem',0,500);
```
- Buttons:** Below the query editor are buttons for 'SELECT \*', 'SELECT', 'INSERT', 'UPDATE', 'DELETE', 'Clear', and 'Format'. The 'Go' button is located at the bottom right and is circled in red.
- Options:** At the bottom, there are options for 'Delimiter' (set to ';'), 'Show this query here again' (checked), 'Retain query box' (unchecked), and 'Rollback when finished' (unchecked).

# Tabel produse

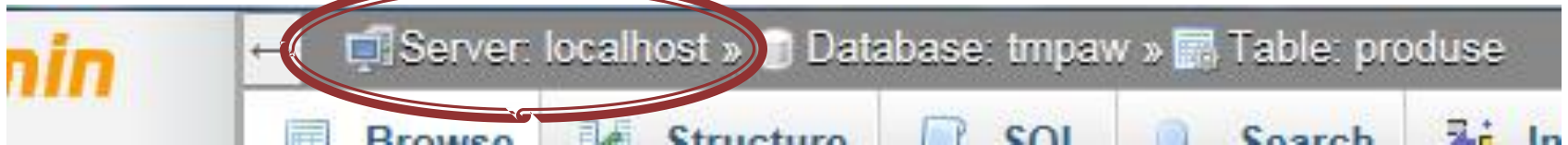
The screenshot shows the phpMyAdmin interface for a MySQL database named 'tmpaw'. The current view is the 'Structure' tab for the 'produse' table. The table has 9 rows and 7 columns: id\_produs, id\_categ, nume, detalii, cant, and pret. The 'Structure' tab is circled in red, and the 'produse' table is selected in the left sidebar, also circled in red. The SQL query shown is 'SELECT \* FROM `produse`'. The table data is as follows:

	id_produs	id_categ	nume	detalii	cant	pret
<input type="checkbox"/>	1	1	carte	mai multe pagini scrise legate	0	100
<input type="checkbox"/>	2	1	caiet	mai multe pagini goale legate	0	75
<input type="checkbox"/>	3	1	hartie scris	mai multe pagini goale NElegate	0	50
<input type="checkbox"/>	4	2	penar	loc de depozitat instrumente de scris	0	150
<input type="checkbox"/>	5	2	stilou	instrument de scris albastru	0	125
<input type="checkbox"/>	6	2	creion	instrument de scris gri	0	25
<input type="checkbox"/>	7	3	cd	canta	0	50
<input type="checkbox"/>	8	3	dvd	vizual	0	100
<input type="checkbox"/>	9	3	blue ray	vizual extrem	0	500



# Adaugare utilizator

- Server → Users → Add user



A screenshot of the phpMyAdmin 'Users overview' page. The navigation bar at the top shows 'Server: localhost' circled in red, and the 'Users' button is also circled in red. The main content area displays a table of users with columns for 'User name', 'Host', 'Password', 'Global privileges', 'Grant', and 'Action'. Below the table, there is a 'New' button circled in red, and an 'Add user' link with a user icon.

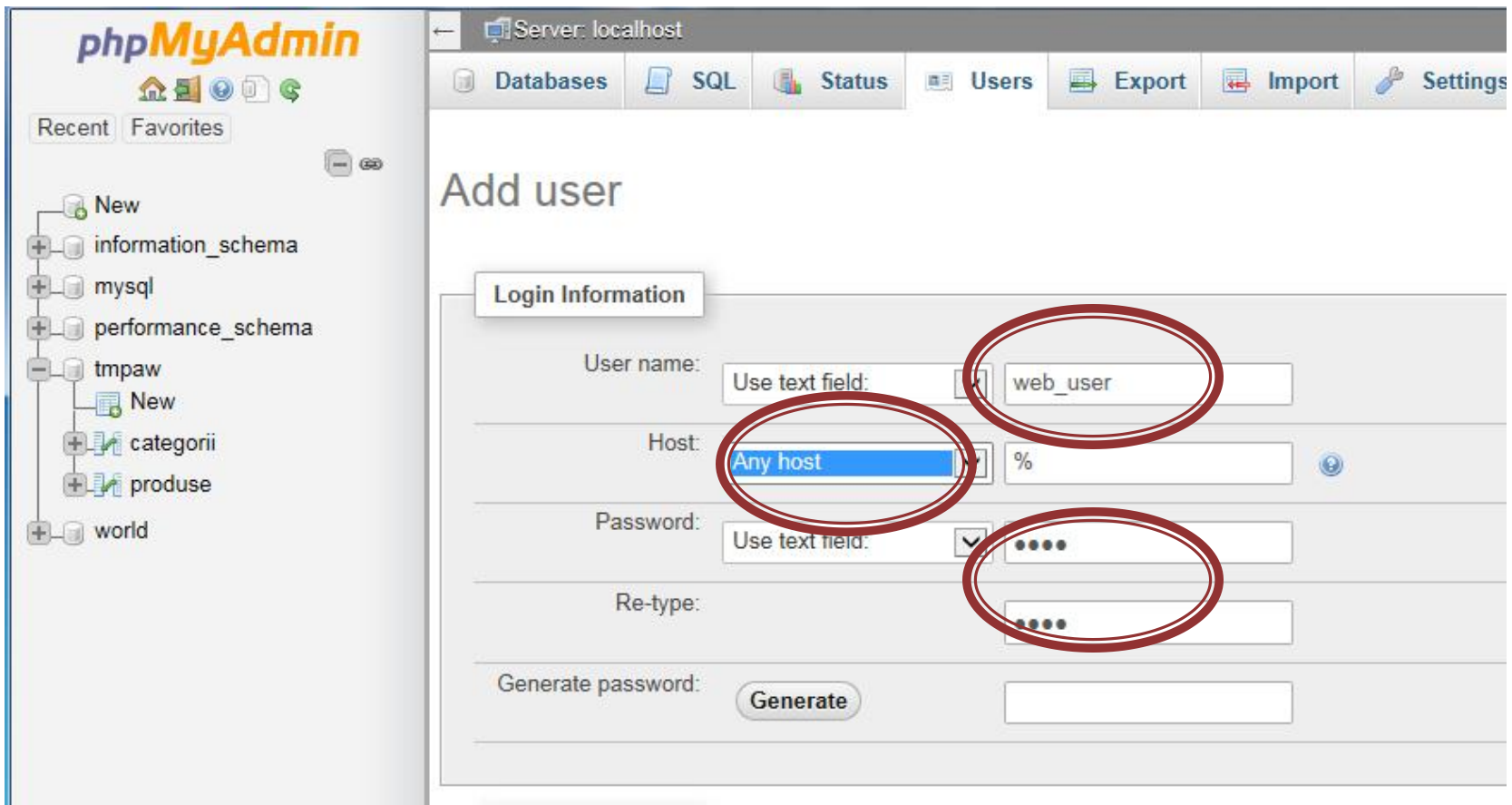
	User name	Host	Password	Global privileges	Grant	Action
<input type="checkbox"/>	root	127.0.0.1	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
<input type="checkbox"/>	root	:::1	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
<input type="checkbox"/>	root	localhost	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
<input type="checkbox"/>	root	tmpaw.etti	Yes	ALL PRIVILEGES	Yes	Edit Privileges Export
<input type="checkbox"/>	web	%	Yes	USAGE	No	Edit Privileges Export

↑  Check All With selected: Export

New  
Add user

# Adaugare utilizator

- Nu e recomandabil/**posibil** sa se utilizeze user-ul MySql "root" pentru aplicatii



The screenshot shows the phpMyAdmin interface for adding a new user. The 'Login Information' section contains the following fields:

- User name:** A text input field containing 'web\_user'.
- Host:** A dropdown menu with 'Any host' selected.
- Password:** A text input field with masked characters (dots).
- Re-type:** A text input field with masked characters (dots).

Four red circles are drawn around the 'User name', 'Host', 'Password', and 'Re-type' fields, highlighting them.

# Drepturi de acces

- Server → Users → Edit Privileges

The screenshot shows the phpMyAdmin interface. The breadcrumb 'Server: localhost' is circled in red. The 'Users' menu item in the top navigation bar is also circled in red. The 'Users overview' table is shown below, with the 'Edit Privileges' link for the 'web\_user' row circled in red.

	User name	Host	Password	Global privileges	Grant	Action
<input type="checkbox"/>	root	127.0.0.1	Yes	ALL PRIVILEGES	Yes	<a href="#">Edit Privileges</a> <a href="#">Export</a>
<input type="checkbox"/>	root	:::1	Yes	ALL PRIVILEGES	Yes	<a href="#">Edit Privileges</a> <a href="#">Export</a>
<input type="checkbox"/>	root	localhost	Yes	ALL PRIVILEGES	Yes	<a href="#">Edit Privileges</a> <a href="#">Export</a>
<input type="checkbox"/>	root	tmpaw.etti	Yes	ALL PRIVILEGES	Yes	<a href="#">Edit Privileges</a> <a href="#">Export</a>
<input type="checkbox"/>	web	%	Yes	USAGE	No	<a href="#">Edit Privileges</a> <a href="#">Export</a>
<input type="checkbox"/>	web_user	%	Yes	USAGE	No	<a href="#">Edit Privileges</a> <a href="#">Export</a>

# Drepturi de acces

- Database → nume → Go

The screenshot shows the phpMyAdmin interface for a MySQL server on localhost. The 'Database' tab is selected and circled in red. The main content area is titled 'Edit Privileges: User 'web\_user'@'%' and shows a section for 'Database-specific privileges'. A table below this section lists the databases 'mysql', 'tmpaw', and 'world', which are also circled in red. The table has columns for 'Database', 'Privileges', 'Grant', 'Table-specific privileges', and 'Action'. The 'Privileges' column is currently empty for all entries, and the 'Action' column contains the word 'None'. Below the table, there is a text input field for 'Add privileges on the following database(s)' and a 'Go' button.

Server: localhost

Databases SQL Status Users Export Import Settings

Global Database Change password Login Information

Edit Privileges: User 'web\_user'@'%'

Database-specific privileges

Database	Privileges	Grant	Table-specific privileges	Action
mysql				None
tmpaw				None
world				None

Add privileges on the following database(s):



# Drepturi de acces

- Se aloca drepturile SELECT + INSERT + UPDATE + DELETE asupra bazei de date create

The screenshot shows the phpMyAdmin interface for editing privileges. The user 'web\_user'@'%' is selected for the database 'tmpaw'. The 'Data' section is checked, indicating that SELECT, INSERT, UPDATE, and DELETE privileges are granted. The 'Structure' and 'Administration' sections are unchecked.

Server: localhost

Databases SQL Status Users Export Import Settings Replicati

Database Table

Edit Privileges: User `'web_user'@'%'` - Database `tmpaw`

Database-specific privileges  Check All

Note: MySQL privilege names are expressed in English.

Data	Structure	Administration
<input checked="" type="checkbox"/> SELECT	<input type="checkbox"/> CREATE	<input type="checkbox"/> GRANT
<input checked="" type="checkbox"/> INSERT	<input type="checkbox"/> ALTER	<input type="checkbox"/> LOCK TABLES
<input checked="" type="checkbox"/> UPDATE	<input type="checkbox"/> INDEX	<input type="checkbox"/> REFERENCES
<input checked="" type="checkbox"/> DELETE	<input type="checkbox"/> DROP	
	<input type="checkbox"/> CREATE TEMPORARY TABLES	
	<input type="checkbox"/> SHOW VIEW	

# Drepturi de acces, verificare

- Nume → Privileges
- Marea majoritate a aplicatiilor **nu** au nevoie de drepturi de acces la structura/administrare

Server: localhost » Database: tmpaw

Structure SQL Search Query Export Import Operations **Privileges** Routing

Users having access to "tmpaw"

User	Host	Type	Privileges	Grant	Action	
<input type="checkbox"/>	root	127.0.0.1	global	ALL PRIVILEGES	Yes	Edit Privileges
<input type="checkbox"/>	root	:::1	global	ALL PRIVILEGES	Yes	Edit Privileges
<input type="checkbox"/>	root	localhost	global	ALL PRIVILEGES	Yes	Edit Privileges
<input type="checkbox"/>	root	tmpaw.etti	global	ALL PRIVILEGES	Yes	Edit Privileges
<input type="checkbox"/>	web_user	%	database-specific	SELECT, INSERT, UPDATE, DELETE	No	Edit Privileges

Check All With selected: Export

# Index

- Adaugare index e esentiala pentru viteza
  - exemplu, produse grupate pe categorii, selectia produselor dintr-o categorie se face cu :
    - `SELECT * FROM `produse` WHERE `id_categ` = 1`
- Tabel → Structure → Index / Selectare + Index

The screenshot shows the phpMyAdmin interface for a database named 'tmpaw'. The 'Table: produse' structure is displayed in 'Table structure' view. The table has the following columns:

#	Name	Type	Collation	Attributes	Null	Default	Extra	Action
1	id_produs	int(11)			No	None	AUTO_INCREMENT	Change Drop Primary Unique Index Spatial Fulltext Distinct values
2	id_categ	int(11)			No	None		Change Drop Primary Unique Index Spatial Fulltext Distinct values
3	nume	varchar(45)	utf8_general_ci		No			Change Drop Primary Unique Index Spatial Fulltext Distinct values
4	detalii	varchar(150)	utf8_general_ci		Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values
5	cant	int(11)			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values
6	pret	float			Yes	NULL		Change Drop Primary Unique Index Spatial Fulltext Distinct values

Annotations in the image include:





- A red circle around the 'Structure' tab in the top navigation bar.
- A red circle around the 'id\_categ' row in the table structure.
- A red circle around the 'Unique' and 'Index' icons in the 'Action' column for the 'id\_categ' row.
- A green circle around the 'id\_categ' column name in the table structure.
- A red circle around the 'produse' table in the left sidebar.
- A green circle around the 'Unique' and 'Index' icons in the bottom toolbar.

# Verificare/Stergere index

- Apasare +Indexes, se deschide lista de indecsi
- Apasare -Indexes, se inchide lista de indecsi

- Indexes

Indexes ⓘ

Action	Keyname	Type	Unique	Packed	Column	Cardinality	Collation	Null	Comment
 Edit  Drop PRIMARY		BTREE	Yes	No	id_produ	9	A	No	
 Edit  Drop id_categ		BTREE	No	No	id_categ	9	A	No	

Create an index on  columns

# Backup, Restore

- Ca și în cazul Windows 2000 facilitatea de Backup realizează un script SQL care conține structura și datele exprimate sub forma de interogări SQL
- O deosebire între PhpMyAdmin și aplicațiile specifice MySQL (aceleși de pe Windows 2000 sau MySQL Workbench) este absența liniilor de creare a bazei de date
  - CREATE DATABASE IF NOT EXISTS tmpaw;
  - USE tmpaw;
- La utilizarea PhpMyAdmin trebuie să se creeze manual înaintea restaurării baza de date

# Script SQL Backup - utilitate

- Poate fi folosit ca un model extrem de bun pentru comenzile necesare pentru crearea programatica (din PHP de exemplu) a bazei de date

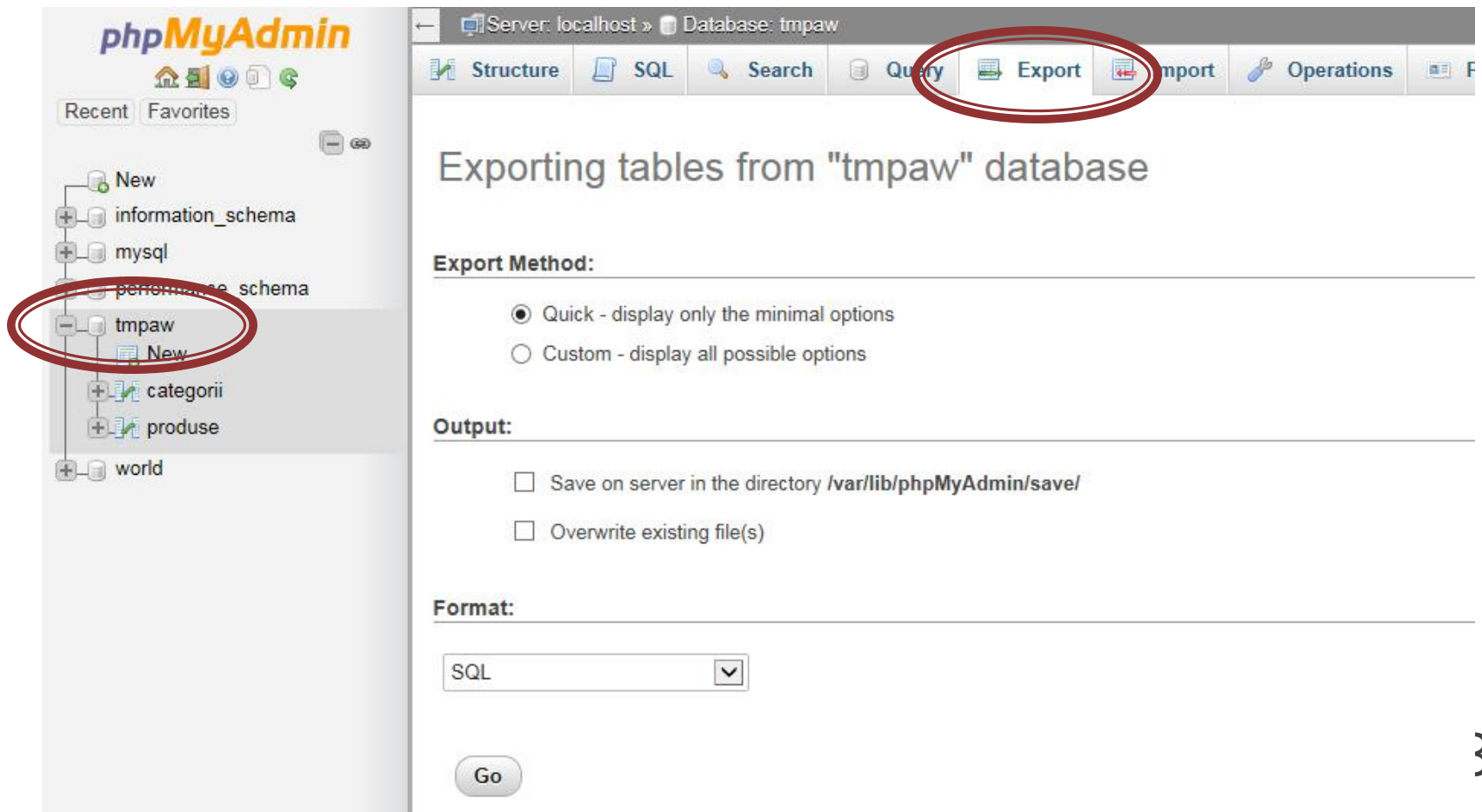
```
CREATE DATABASE IF NOT EXISTS tmpaw;  
USE tmpaw;
```

```
DROP TABLE IF EXISTS `categorii`;  
CREATE TABLE `categorii` (  
  `id_categ` int(10) unsigned NOT NULL auto_increment,  
  `nume` varchar(45) NOT NULL,  
  `detalii` varchar(150) default NULL,  
  PRIMARY KEY (`id_categ`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
INSERT INTO `categorii` (`id_categ`,`nume`,`detalii`) VALUES  
(1,'papetarie',NULL),  
(2,'instrumente',NULL),  
(3,'audio-video',NULL);
```

# Backup

- Nume (tabel sau baza de date) → Export



The screenshot displays the phpMyAdmin interface. On the left sidebar, the database 'tmpaw' is selected and circled in red. The main panel shows the 'Export' menu item, also circled in red. The export options are as follows:

Server: localhost » Database: tmpaw

Structure SQL Search Query **Export** Import Operations F

### Exporting tables from "tmpaw" database

**Export Method:**

- Quick - display only the minimal options
- Custom - display all possible options

**Output:**

- Save on server in the directory `/var/lib/phpMyAdmin/save/`
- Overwrite existing file(s)

**Format:**

SQL

Go



# Restore

- Se creaza in avans baza de date
- Nume → Import → Browse (alegere fisier backup)
- fisierele SQL pot fi compresate gzip, bzip2, zip

The screenshot shows the phpMyAdmin interface. On the left sidebar, the database 'tmpaw' is selected and circled in red. The main content area displays the 'Import' page for the database 'tmpaw'. The 'Import' button in the top navigation bar is also circled in red. The 'File to Import:' section has a 'Browse...' button circled in red. The 'Partial Import:' section has a checked checkbox for 'Allow the interruption of an import in case the script detects it is close to the PHP timeout limit.' and a text input field for 'Skip this number of queries (for SQL) or lines (for other formats), starting from the first one:' with the value '0'.

phpMyAdmin

Server: localhost » Database: tmpaw

Structure SQL Search Query Export Import Operations Privileges Routines

## Importing into the database "tmpaw"

**File to Import:**

File may be compressed (gzip, bzip2, zip) or uncompressed.  
A compressed file's name must end in `.[format].[compression]`. Example: `file.sql.gz`

Browse your computer:  **Browse...** (Max: 2048KiB)

You may also drag and drop a file on any page.

Select from the web server upload directory `/var/lib/phpMyAdmin/upload/`: There are no files to upload!

Character set of the file:

**Partial Import:**

Allow the interruption of an import in case the script detects it is close to the PHP timeout limit. (This might be a good way to import large file

Skip this number of queries (for SQL) or lines (for other formats), starting from the first one:



MySql – Server Windows 2000

# Mini – Indrumar practic

## Lucru cu bazele de date

# Realizarea bazei de date

- Se recomanda utilizarea utilitarului **MySQL Query Browser** sau un altul echivalent pentru crearea scheletului de baza de date (detalii – laborator 1)
- Se initializeaza aplicatia cu drepturi depline (“root” si parola)
  - se creaza o noua baza de date:
    - in lista “Schemata” – Right click – Create New Schema
  - se activeaza ca baza de date curenta noua “schema” – Dublu click pe numele ales

# Introducere tabele

- Introducere tabel – Click dreapta pe numele bazei de date aleasa – Create New Table
- se defineste structura tabelului
  - nume coloane
  - tip de date
  - NOT NULL – daca se accepta ca acea coloana sa ramana fara date (NULL) sau nu
  - AUTOINC – daca acea coloana va fi de tip intreg si va fi incrementata automat de server (util pentru crearea cheilor primare)
  - Default value – valoarea implicita care va fi inserata daca la introducerea unei linii noi nu se mentioneaza valoare pentru acea coloana (legat de optiunea NOT NULL)

# Tabel Categorii

The screenshot shows the MySQL Table Editor interface for a table named 'categorii' in the 'tmpaw' database. The table is currently empty. The 'Columns and Indices' tab is active, showing the following columns:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id_categ	INT(10)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
nume	VARCHAR(45)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
detalii	VARCHAR(150)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY	NULL	

The 'Indices' tab is also active, showing a primary index named 'PRIMARY' on the 'id\_categ' column. The index settings are:

- Index Name: PRIMARY
- Index Kind: PRIMARY
- Index Type: BTREE
- Index Columns: id\_categ

The 'Apply Changes' button is highlighted in blue. The background shows a MySQL Query Browser window with a query area containing 'SELECT \* FROM' and a result set table with columns 'id\_produ' and 'id\_c'.

# Tabel Prognose

The screenshot shows the MySQL Table Editor interface for a table named 'produse' in the 'tmpaw' database. The table has six columns: 'id\_produkt' (INT(10), UNSIGNED, ZEROFILL, NULL), 'id\_kategorija' (INT(10), UNSIGNED, ZEROFILL), 'naziv' (VARCHAR(45), BINARY), 'opis' (VARCHAR(150), BINARY, NULL), 'cena' (INT(10), UNSIGNED, ZEROFILL, NULL), and 'postotak' (FLOAT, UNSIGNED, ZEROFILL, NULL). A primary index is defined on the 'id\_produkt' column with index settings: Index Name: PRIMARY, Index Kind: PRIMARY, Index Type: BTREE. The background shows a query result set with columns 'id\_produkt' and 'id\_kategorija' and rows 1 through 9.

MySQL Query Browser - Connection: root@server / tmpaw

File Edit View Tools Window Help

Transaction Explain Compare

MySQL Table Editor

Table Name: produse Database: tmpaw Comment: InnoDB free: 11264 kB

Columns and Indices Table Options Advanced Options

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id_produkt	INT(10)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
id_kategorija	INT(10)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL		
naziv	VARCHAR(45)	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/> BINARY		
opis	VARCHAR(150)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY	NULL	
cena	INT(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
postotak	FLOAT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

Indices Foreign Keys Column Details

PRIMARY

Index Settings

Index Name: PRIMARY

Index Kind: PRIMARY

Index Type: BTREE

Index Columns (Use Drag'n'Drop)

id\_produkt


Apply Changes Discard Changes Close

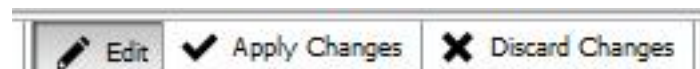
6: 8

Edit Apply Changes Discard Changes First Last Search

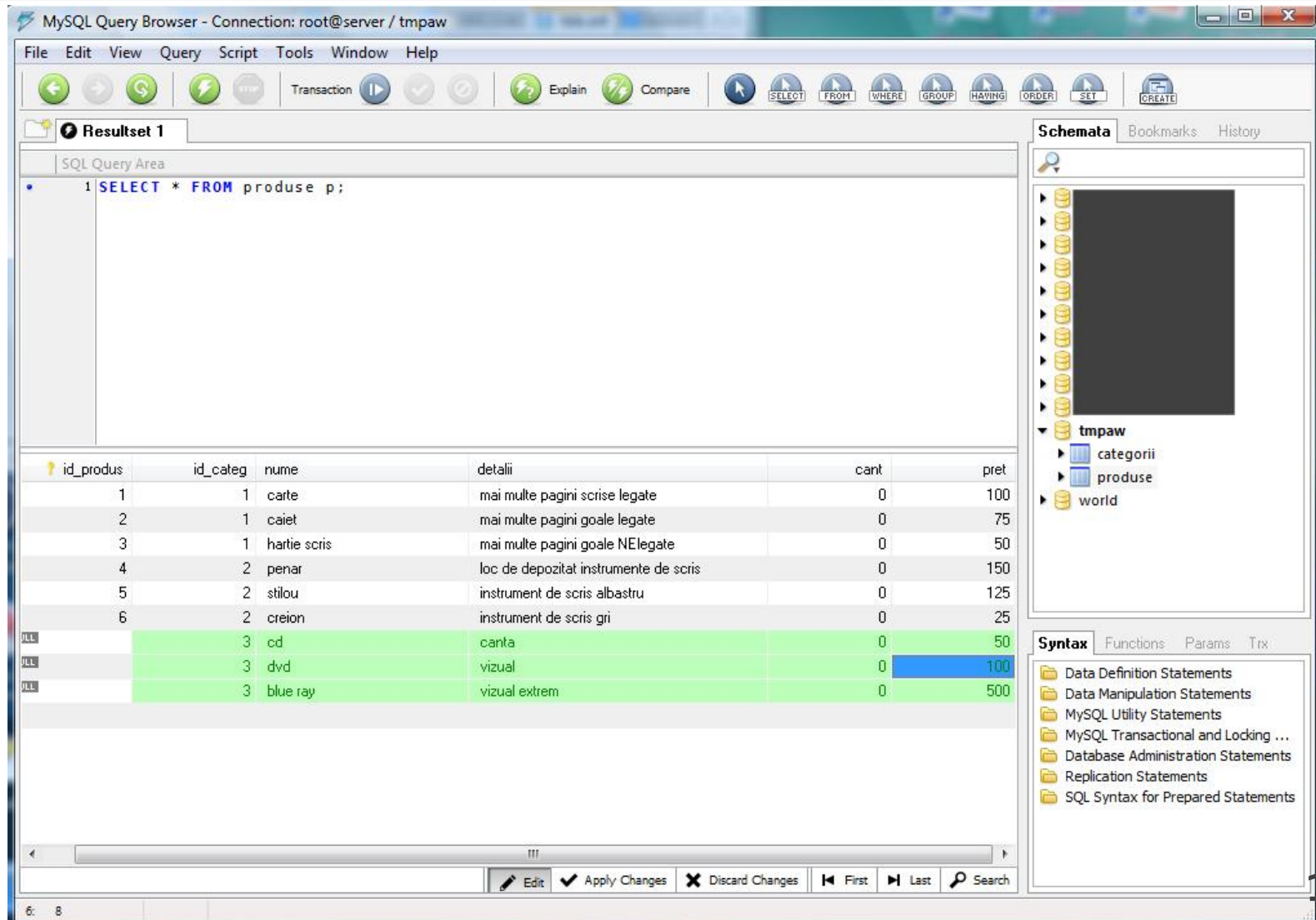
193

# Introducere date initiale

- Dublu click pe tabel → In zona “SQL Query Area” se completeaza interogarea de selectie totala
  - SELECT \* FROM produse p;
- Executia interogarii SQL
  - Meniu → Query → Execute
  - Bara de butoane 
- Lista rezultata
  - initial vida
  - poate fi editata – butoanele “Edit”, “Apply Changes”, “Discard Changes” din partea de jos a listei



# Introducere date initiale



The screenshot displays the MySQL Query Browser interface. The main window shows the SQL Query Area with the query: `1 SELECT * FROM produse p;`. Below the query area, a table of results is displayed. The table has columns: `id_produs`, `id_categ`, `nume`, `detalii`, `cant`, and `pret`. The data rows are as follows:

id_produs	id_categ	nume	detalii	cant	pret
1	1	carte	mai multe pagini scrise legate	0	100
2	1	caiet	mai multe pagini goale legate	0	75
3	1	hartie scris	mai multe pagini goale NElegate	0	50
4	2	penar	loc de depozitat instrumente de scris	0	150
5	2	stilou	instrument de scris albastru	0	125
6	2	creion	instrument de scris gri	0	25
ALL	3	cd	canta	0	50
ALL	3	dvd	vizual	0	100
ALL	3	blue ray	vizual extrem	0	500

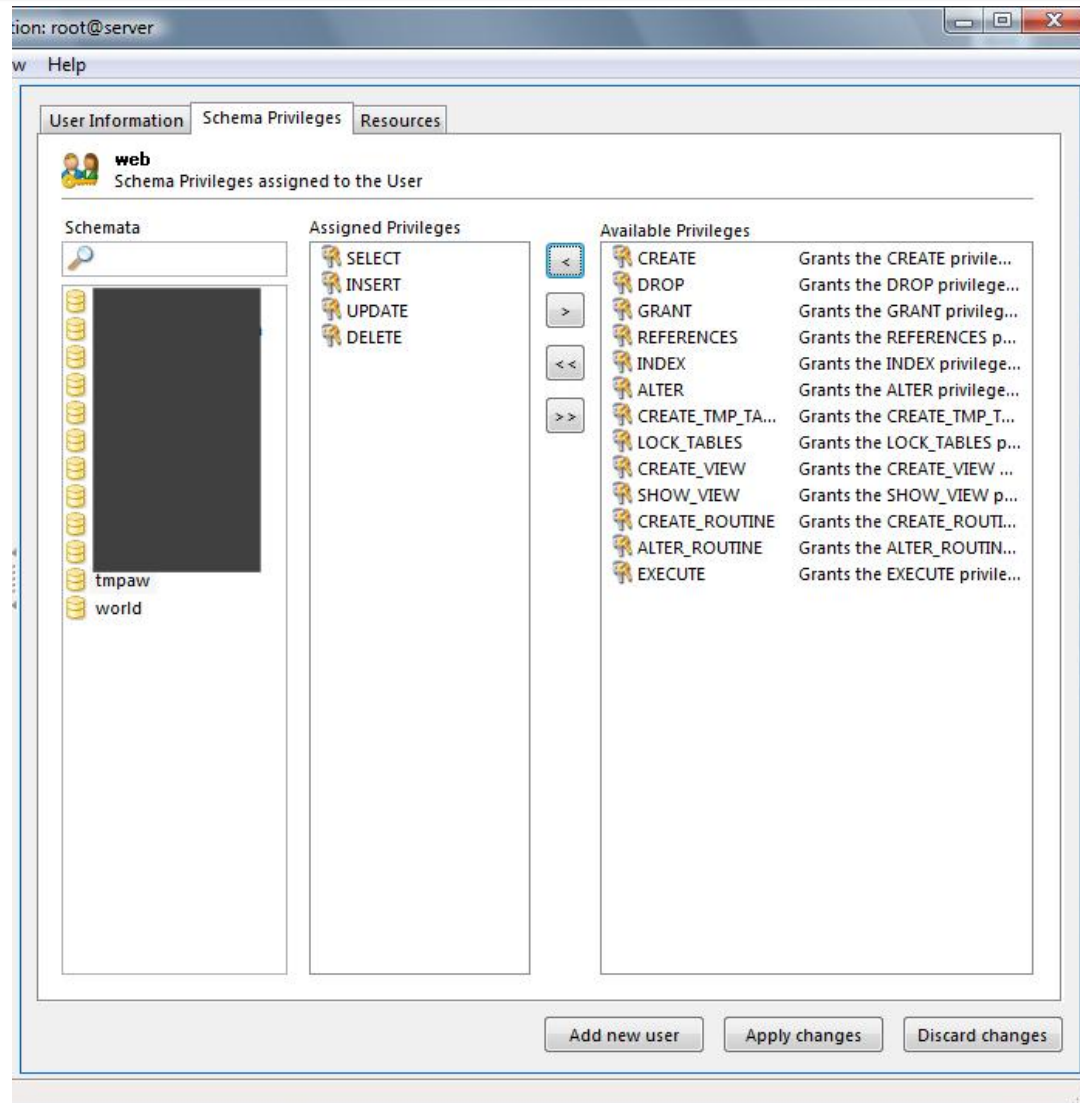
The interface also includes a menu bar (File, Edit, View, Query, Script, Tools, Window, Help), a toolbar with various icons, and a right-hand sidebar with 'Schemata', 'Bookmarks', and 'History' tabs. The 'Schemata' tab shows a tree view of the database structure, including the 'tmpaw' database with 'categorii' and 'produse' tables. The 'Syntax' tab is also visible at the bottom right.

# Backup, Restore, drepturi de acces

- Se recomanda utilizarea utilitarului **MySQL Administrator** sau un altul echivalent (detalii – laborator 1)
- Se initializeaza aplicatia cu drepturi depline (“root” si parola)
- Se creaza un utilizator limitat (detalii – laborator 1)
- Se aloca drepturile “SELECT” + “INSERT” + “UPDATE” asupra bazei de date create (sau mai multe daca aplicatia o cere)



# Drepturi de acces



# Backup

The screenshot shows the MySQL Administrator interface for configuring a backup project. The window title is "MySQL Administrator - Connection: root@server". The main area is titled "Backup Project" and has three tabs: "Backup Project", "Advanced Options", and "Schedule".

**General**

Project Name:  Name for this backup project.

**Schemata**

The Schemata list on the left includes: school, tmpaw, and world. The tmpaw schema is selected and highlighted in blue.


**Backup Content**

Data directory	Obj...	Rows	Data ...	Last update
<input checked="" type="checkbox"/> tmpaw				
<input checked="" type="checkbox"/> categorii	Inno...	3	16384	
<input checked="" type="checkbox"/> produse	Inno...	9	16384	

At the bottom of the window, there are three buttons: "New Project", "Save Project", and "Execute Backup Now".

Yellow arrows indicate the workflow: from the "Backup" icon in the left sidebar to the "tmpaw" schema in the Schemata list, then to the "Backup Content" table, and finally to the "Execute Backup Now" button.

# Restaurarea bazei de date

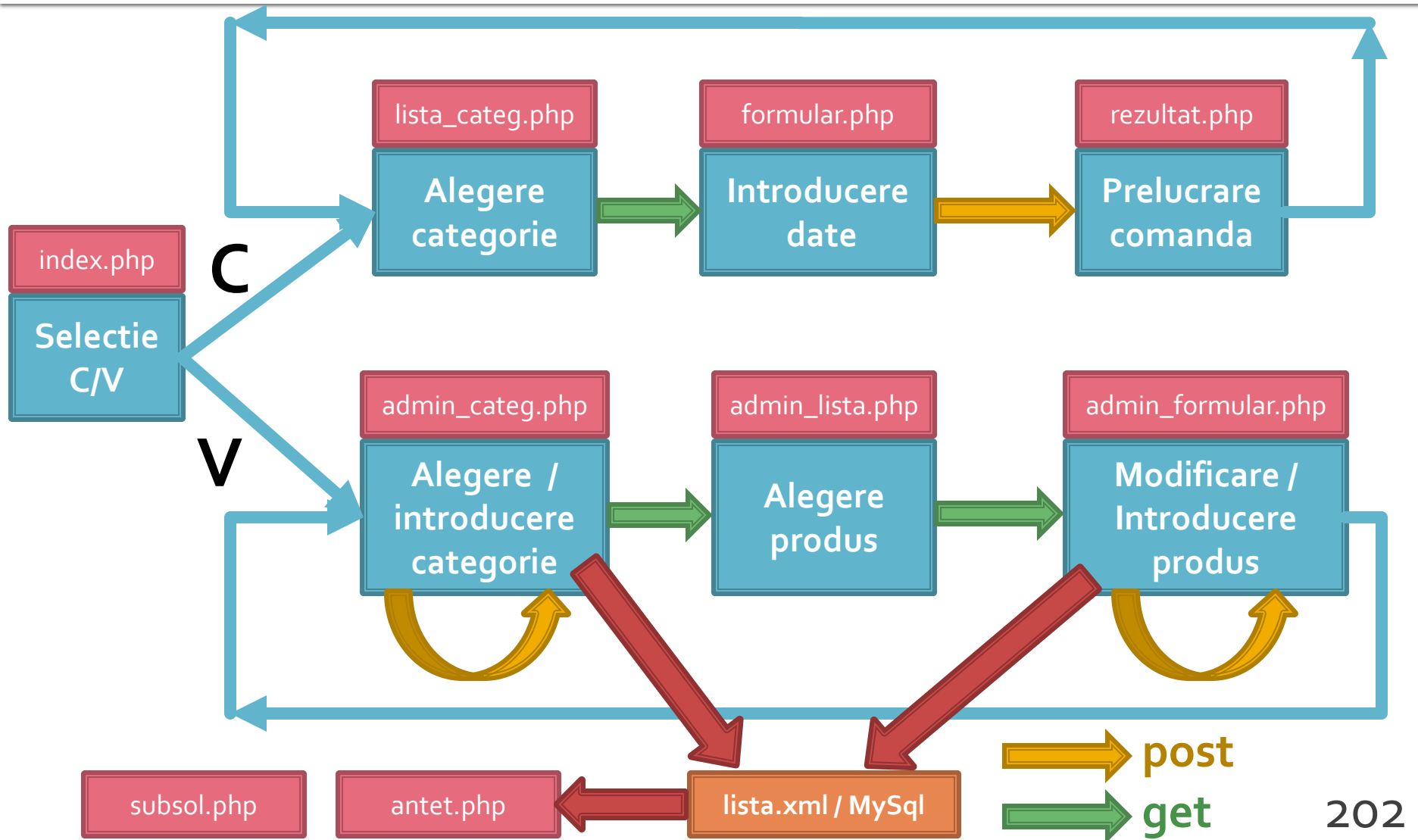
- Din **MySql Administrator**
  - Sectiunea Restore → "Open Backup File"
- Din **MySql Query Browser**
  - Meniu → File → Open Script
  - Executie script SQL
    - Meniu → Script → Execute
    - Bara de butoane 
- Scriptul SQL rezultat contine comenzile/interogariile SQL necesare pentru crearea bazei de date si popularea ei cu date

# Laborator 6

# Laborator 6+7

- Sa se continue magazinul virtual cu:
  - produsele sunt grupate pe categorii de produse
  - sa prezinte utilizatorului o lista de grupe de produse pentru a alege
  - sa prezinte utilizatorului o lista de produse si preturi in grupa aleasa
  - lista de produse si preturi se citeste dintr-o baza de date **MySQL**
  - se preia comanda si se calculeaza suma totala
  - **se creaza o pagina prin care vanzatorul poate modifica preturile si produsele**

# Plan aplicatie



# Rezultat (comparator)

### Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	<a href="#">Papetarie</a>	3
2	<a href="#">Instrumente</a>	3
3	<a href="#">Audio-video</a>	3
4	<a href="#">Calculatoare</a>	3
5	<a href="#">Jucarii</a>	2

Total produse: 14

### Magazin online Firma X SRL

#### Finalizati comanda

Nr.	Produs	Pret	Cantitate
1	Carti	100	<input type="text" value="1"/>
2	Caiete	50	<input type="text" value="2"/>
3	Penare	150	<input type="text" value="1"/>
4	Stilouri	125	<input type="text" value="0"/>
5	Creioane	25	<input type="text" value="0"/>

### Magazin online Firma X SRL

#### Rezultate comanda

Pret total (fara TVA): 350

Pret total (cu TVA): 416.5

Comanda receptionata la data: 17/03/2010 ora 08:24

 post  
 get

# Rezultat (vanzator)

## Magazin Firma X

[Inceput](#) | [Inapoi](#)

### Magazin online Firma X SRL

Alegeti:

- [Cumparator](#)
- [Vanzator](#)

### Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	<a href="#">Papetarie</a>	3
2	<a href="#">Instrumente</a>	3
3	<a href="#">Audio-video</a>	3
4	<a href="#">Calculatoare</a>	3
5	<a href="#">Jucarii</a>	2

Total produse: 14

Categorie noua de produse:

### Lista produse in categoria Calculatoare

Nr.	Produs	Descriere	Pret	Cantitate	Actiuni
1	Laptop	calculator mic	2000	2	<a href="#">modifica</a>
2	Desktop	calculator mare	1000	5	<a href="#">modifica</a>
3	Imprimanta	prn	200	2	<a href="#">modifica</a>
-	Produs nou				<a href="#">adauga</a>

### Produs in categoria Calculatoare

Produs	<input type="text" value="laptop"/>
Descriere	<input type="text" value="calculator mic"/>
Pret	<input type="text" value="2000"/>
Cantitate	<input type="text" value="2"/>





# Tabel Categorii

The screenshot shows the MySQL Table Editor window for a table named 'categorii' in the 'tmpaw' database. The table is currently empty. The editor is configured with the following settings:

- Table Name:** categorii
- Database:** tmpaw
- Comment:** InnoDB free: 11264 kB

The **Columns and Indices** tab is active, showing the following columns:

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id_categ	INT(10)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
nume	VARCHAR(45)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
detalii	VARCHAR(150)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY	NULL	

The **Indices** tab is also active, showing a primary index named 'PRIMARY' on the 'id\_categ' column. The index settings are:

- Index Name:** PRIMARY
- Index Kind:** PRIMARY
- Index Type:** BTREE
- Index Columns:** id\_categ

The **Apply Changes**, **Discard Changes**, and **Close** buttons are visible at the bottom of the editor window.

# Tabel Prognose

The screenshot shows the MySQL Table Editor interface for a table named 'produse' in the 'tmpaw' database. The table is currently empty. The editor displays the table's structure with columns and their attributes.

Column Name	Datatype	NOT NULL	AUTO INC	Flags	Default Value	Comment
id_producs	INT(10)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
id_categ	INT(10)	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL		
nume	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY		
detalii	VARCHAR(150)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> BINARY	NULL	
cant	INT(10)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	
pret	FLOAT	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/> UNSIGNED <input type="checkbox"/> ZEROFILL	NULL	

The 'Indices' tab is active, showing a PRIMARY index named 'PRIMARY' on the 'id\_producs' column. The index settings are:

- Index Name: PRIMARY
- Index Kind: PRIMARY
- Index Type: BTREE

The 'Index Columns' list contains 'id\_producs'. Buttons for 'Apply Changes', 'Discard Changes', and 'Close' are visible at the bottom of the editor.

# Laborator 6 – Mod de lucru

- Se continua lucrul la aplicatie (L5)
- Se recomanda laboratorul **asincron** – S2
- Se poate folosi fisierul cu surse cpypaste.txt  
(site-<http://rf-opto.etti.tuiasi.ro>)

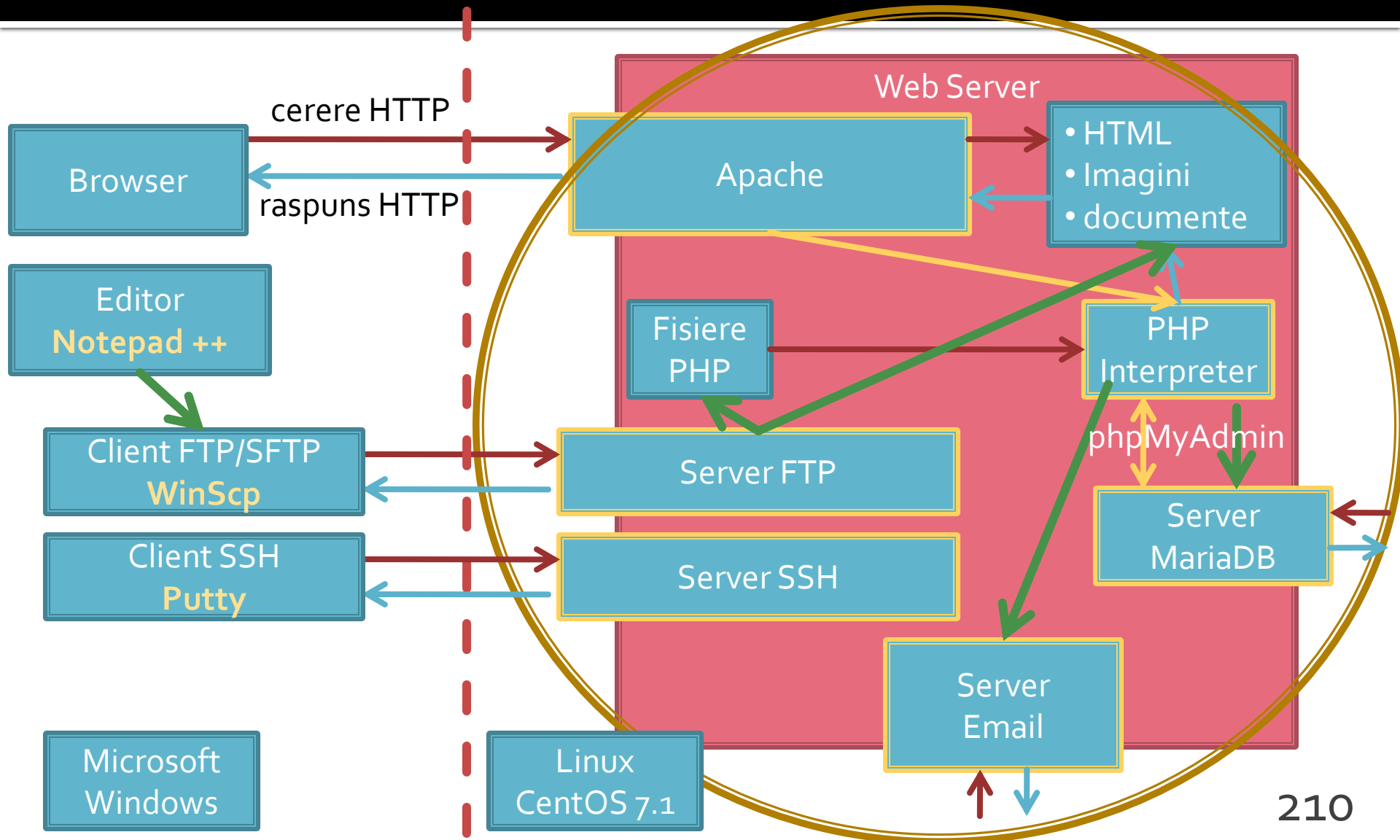
# Laborator 6 – Mod de lucru

- Se ia o decizie relativ la relatia dintre produse si categorii (S63-S67)
  - One to Many
  - Many to Many
- Se creaza cele 2(3) tabele corespunzatoare
- Se populeaza cu date
- Se actualizeaza planul aplicatiei pentru a corespunde cu aplicatia proprie
  - nume de fisiere, tipuri de transfer a datelor

# Laborator 6 – Mod de lucru

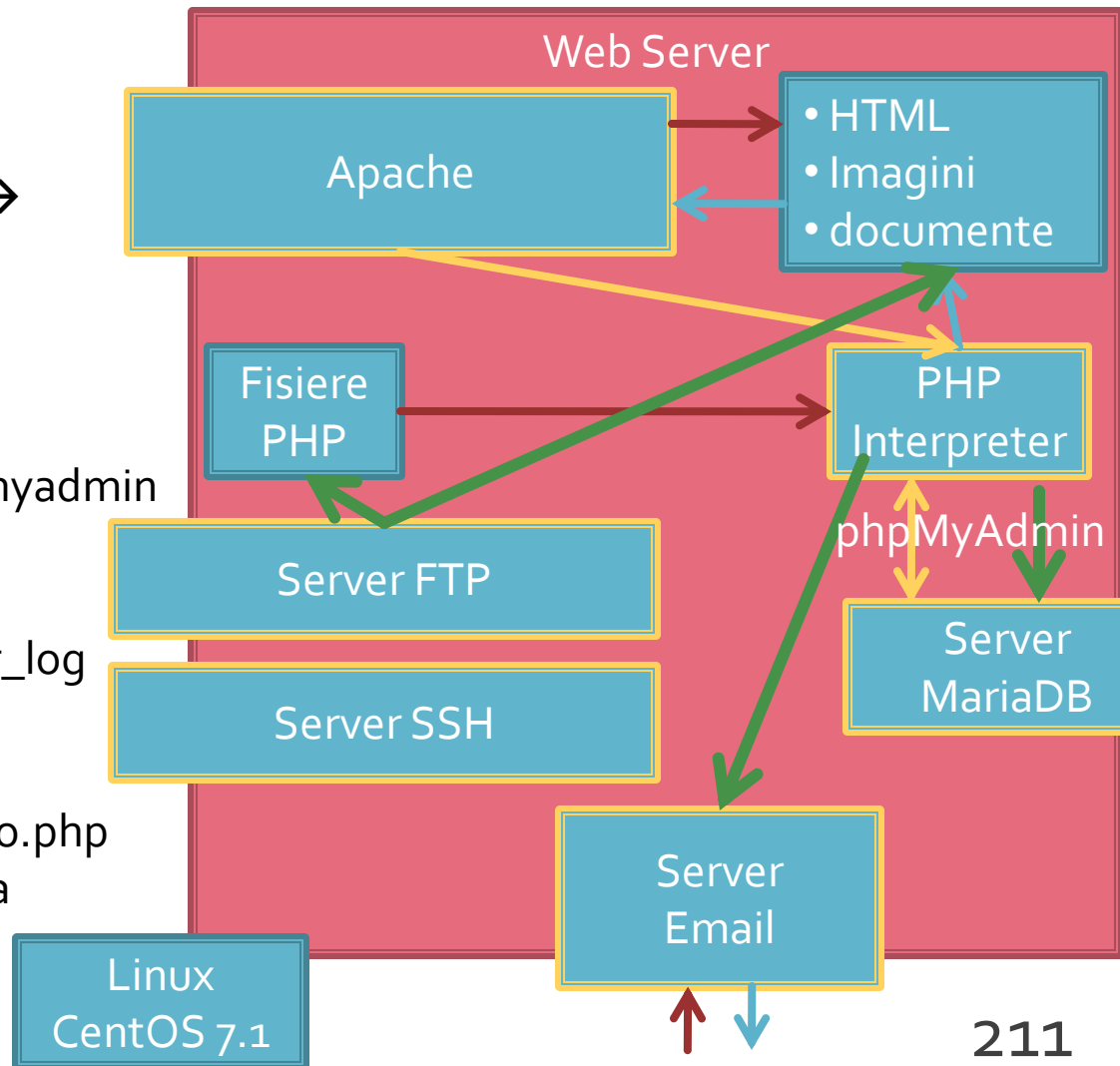
- Se creaza firul de executie paralel pentru vanzator
  - fisierele pentru cumparator reprezinta o buna cale de pornire (Save As, Copy/Paste) pentru 2 din cele 3 fisiere
- Se lucreaza cat mai mult la conversia text -> MySQL
  - activitatea se continua la laboratorul 7

# Utilizare LAMP



# Utilizare LAMP

1. login → root:masterrc
2. ifconfig → 192.168.30.5
3. putty.exe → 192.168.30.5 → SSH → root:masterrc (remote login)
4. [alte comenzi linux dorite]
5. FTP → Winscp → SFTP → student:masterrc@192.168.30.5
6. MySql → http://192.168.30.5/phpmyadmin → root:masterrc
7. Apache Error Log →
  - 7a. putty → nano /var/log/httpd/error\_log
  - 7b. http://192.168.30.5/logfile.php (nonstandard)
8. PHP info → http://192.168.30.5/info.php
9. daca serviciul DHCP duce la oprirea Apache: `service httpd restart`



# Faza de verificare/depanare

- Se recomanda utilizarea posibilitatii vizualizarii matricilor
  - In fisierul care receptioneaza datele
  - temporar pina la definitivarea codului
- utilizarea de cod "verbose" (manual) in etapele initiale de scriere a surselor PHP poate fi extinsa si la alte tipuri de date
  - singura (aproape) metoda de depanare(debug) in PHP
  - `<p>temp <?php echo "a=";echo $a; ?> </p>`

```
echo "<pre>";  
print_r($_POST);  
echo "</pre>";
```



# Depanare

```
echo "<pre>";  
print_r($_POST);  
echo "</pre>";
```

```
<p>temp <?php echo  
"a=";echo $a; ?> </p>
```

# Contact

- Laboratorul de microunde si optoelectronica
- <http://rf-opto.etti.tuiasi.ro>
- [rdamian@etti.tuiasi.ro](mailto:rdamian@etti.tuiasi.ro)