Lecture 12 (w13)

2025/2026

Databases, Web Programming and Interfacing

DWPI

- Databases, Web Programming and Interfacing
  - An VI IT4T
    - 1C/1L/1P
- Timetable
  - Friday, every week (fiecare saptamana) 1C + 2L (17-20)
  - Test + Project Presentation: **last week** (Weekend)

Grade

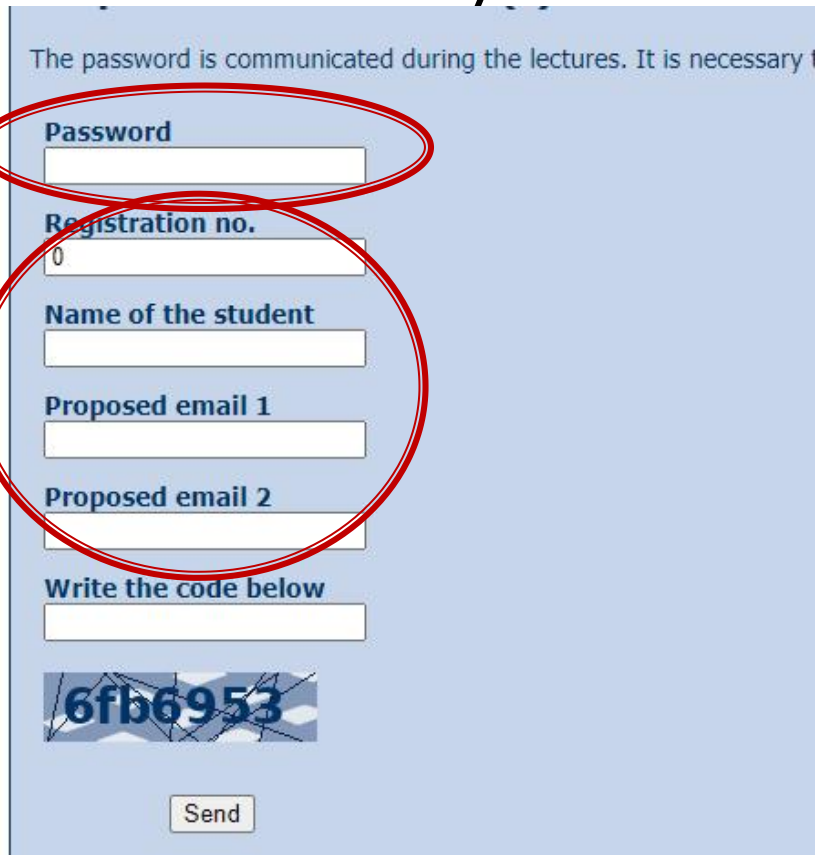
- 10% - Test/Exam – last week – 1h
- 40% - Project
  - Personal(80%)/Team(20%)
- Previous database project
  - receives a **web user interface**
  - **split** in individual assignments
  - bonus (individual) for **supplemental embedded systems access!**
    - admin/user – web user interface
    - data supplier – embedded system (even partial)

Info

- https://rf-opto.etti.tuiasi.ro/master_it.php

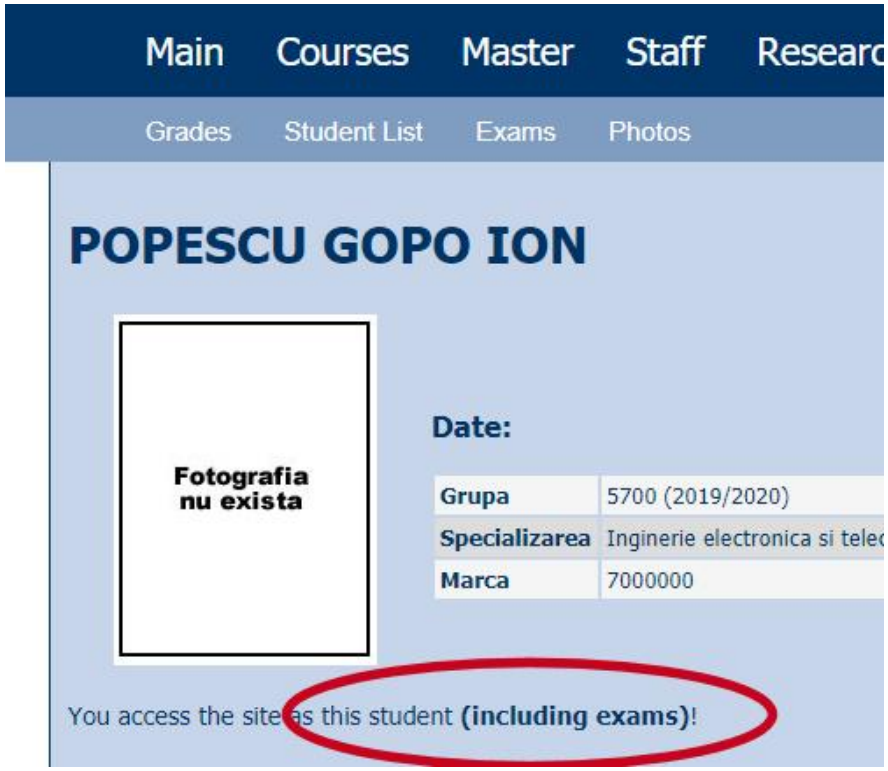
Online – Registration no.

- access to **online exams** requires the **password** received by email



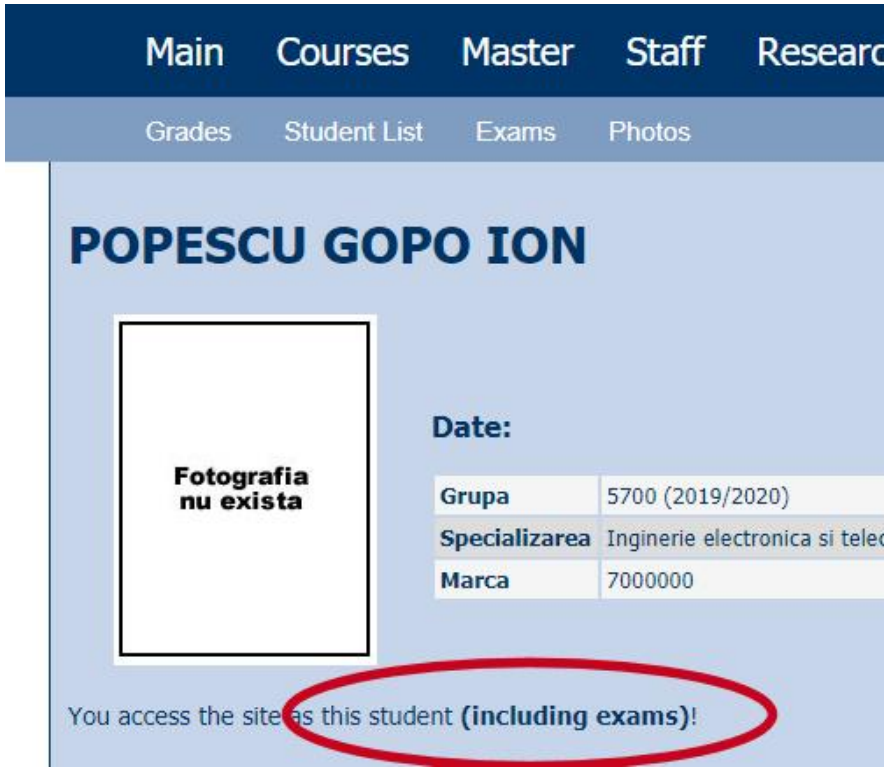
Online

- access email/password



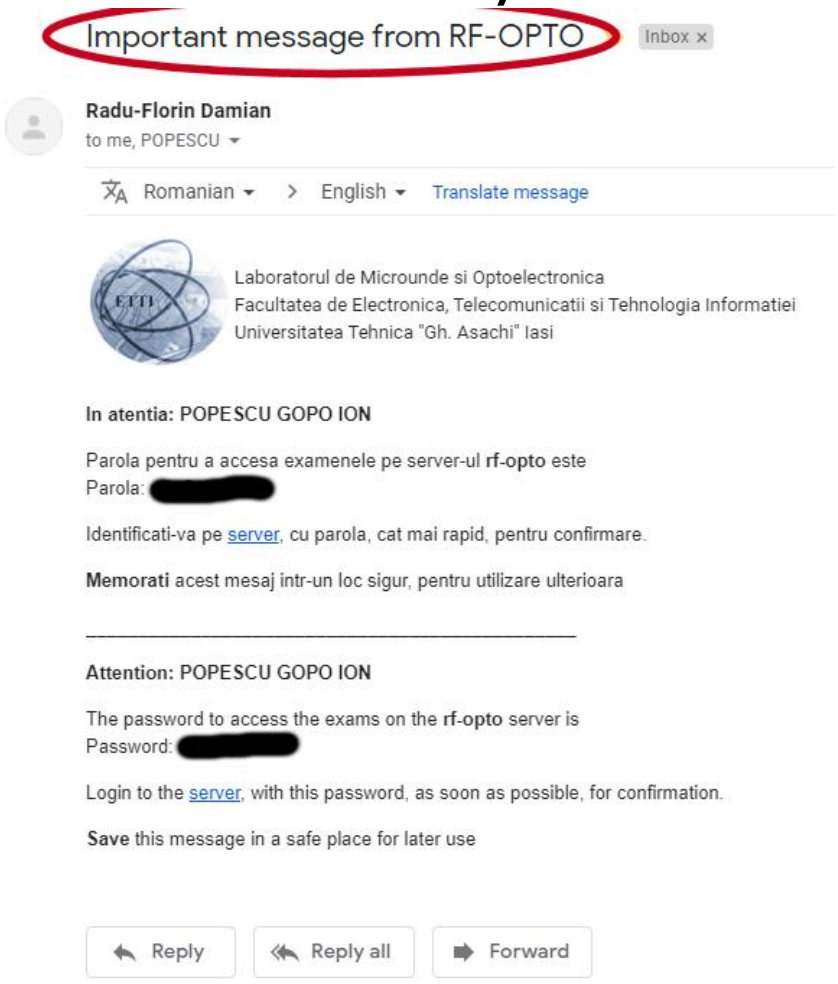
Online

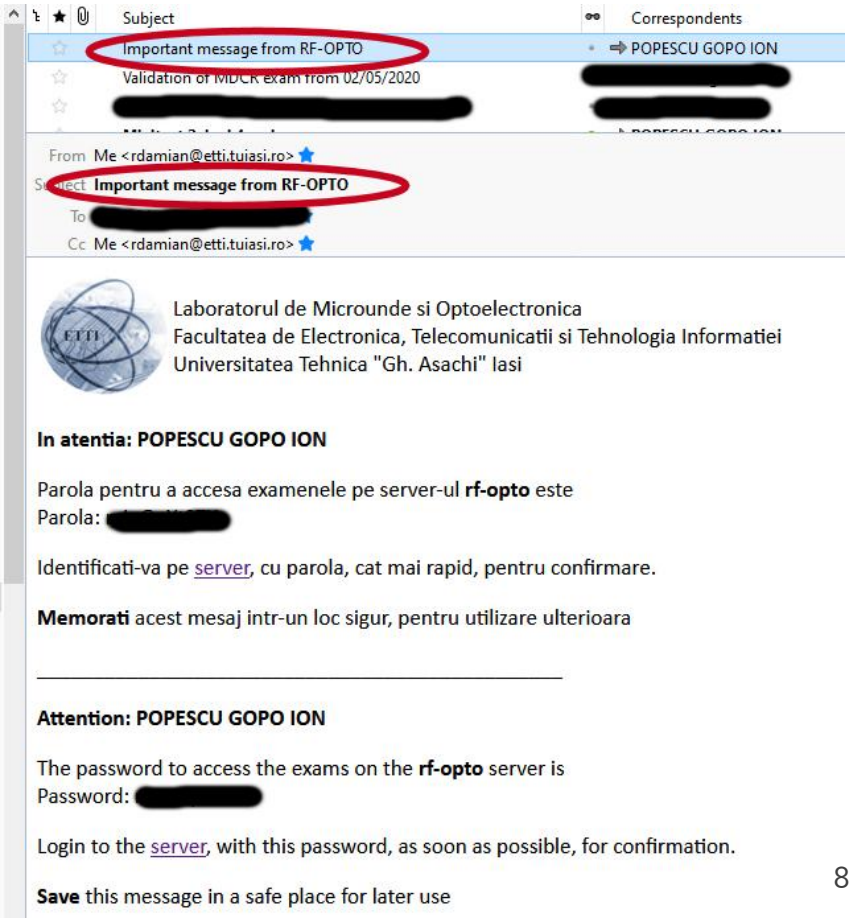
- access email/password



Password

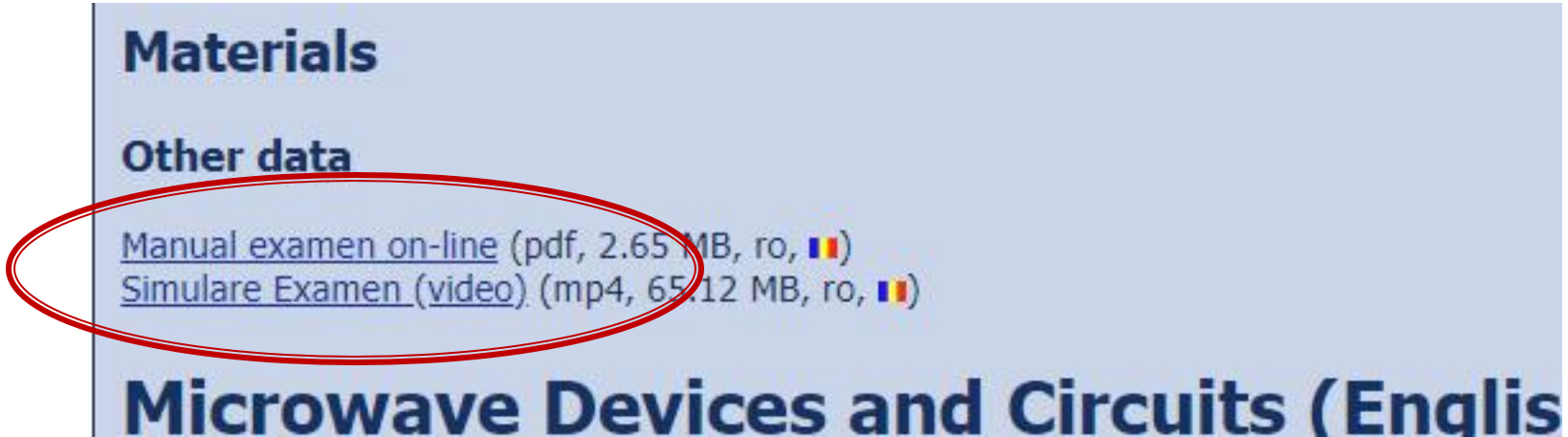
- received by email





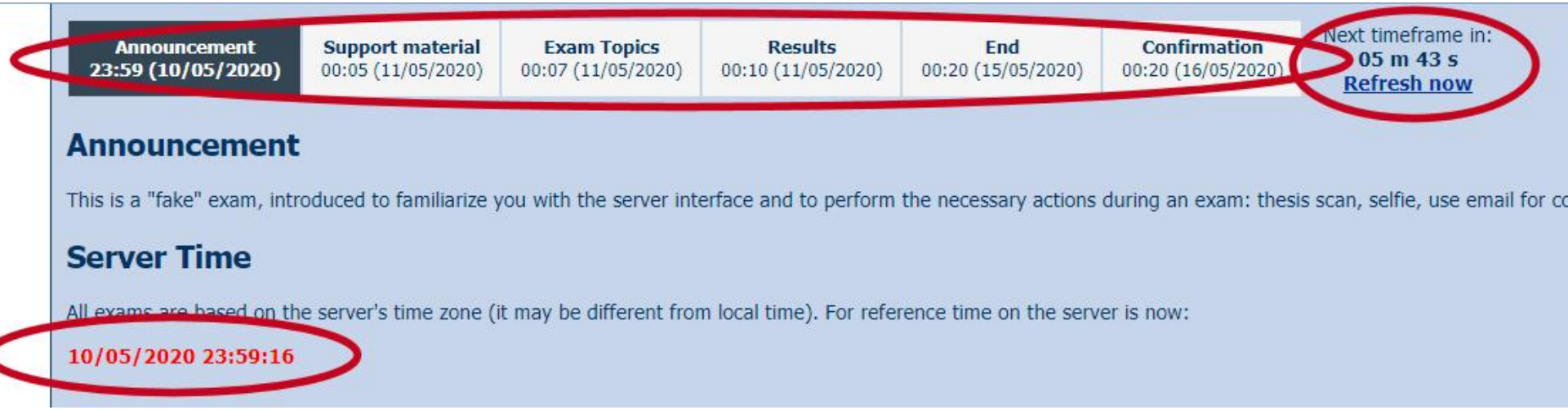
Manual examen online

- The online exam app used for:
  - lectures (attendance)
  - laboratory
  - project
  - examinations



Examen online

- always against a **timetable**
  - long period (project submission/laboratory results)
  - short period (tests: 15min, exam: 2h)



2025/2026

Project

Project

- Submission: **On-site**
- Presentation (in front of the colleagues) + files submission
- 3 files
  - **1 *.pdf** (print-screen while using the application, short usage instructions, a mini-user manual for the application) (**personal**)
  - **1 *.sql** (backup of the database required to run the application) (**team**)
  - archive of the application (inside: files *.php, *.jpg, folder tree etc., archived: ***.zip, *.7z** etc.) (**team/personal**)

Project grading

- **(2p)** the application runs on the **reference server** (can be downloaded from rf-opto: Debian, php 8 **or** Ubuntu, php 7 **or** CentOS 7, php 5): extract files from the ***.zip** archive in a folder on the server, restore database from the ***.sql** backup file
- **(2p)** the ***. pdf** file containing the user manual exists and is appropriate for the submitted application
- **(2p)** the application **flowchart** has been submitted and contains appropriate data
- **(4p)** presentation on-site of the **application**

Online exam 19.12 – xx.06

- on-line exam for the submission of the final individual work inside the team (/individual) web application.
- Data:
  - 1. **Selfie**, **file**, taken during the submission process (acting as a signature for the submission)
  - 2. **Title** of your (team) application/project, **text**
  - 3. **Team members**, **text**
  - 4. **PDF file, file**, includes a **flowchart** of your team application with identification of **individual assignment** inside the team application, and print-screen while using the application, short usage instructions, a mini-user manual for the application (**only** for your **individual assignment**)
  - 4. **SQL file**, **file**, (*.sql), backup of the final database required to run your submitted application
  - 5. **Archive file**, **file**, one archive of the application (inside: files *.php, *.jpg, folder tree etc., archived: *.zip, *.7z etc.)

Online exam 19.12 – xx.06

- The files must be uploaded by **all team members, even if they are identical** (access to the common files is a minimal proof of team membership).
- The "SQL file" and "Archive file" can (should) be common, as is the case for the overall flowchart of the team application.
- However, the identification of individual assignment on the flowchart and the mini-user manual (print-screen, usage instructions) **must** be different between team members, without overlaps.

Hypertext PreProcessor

PHP

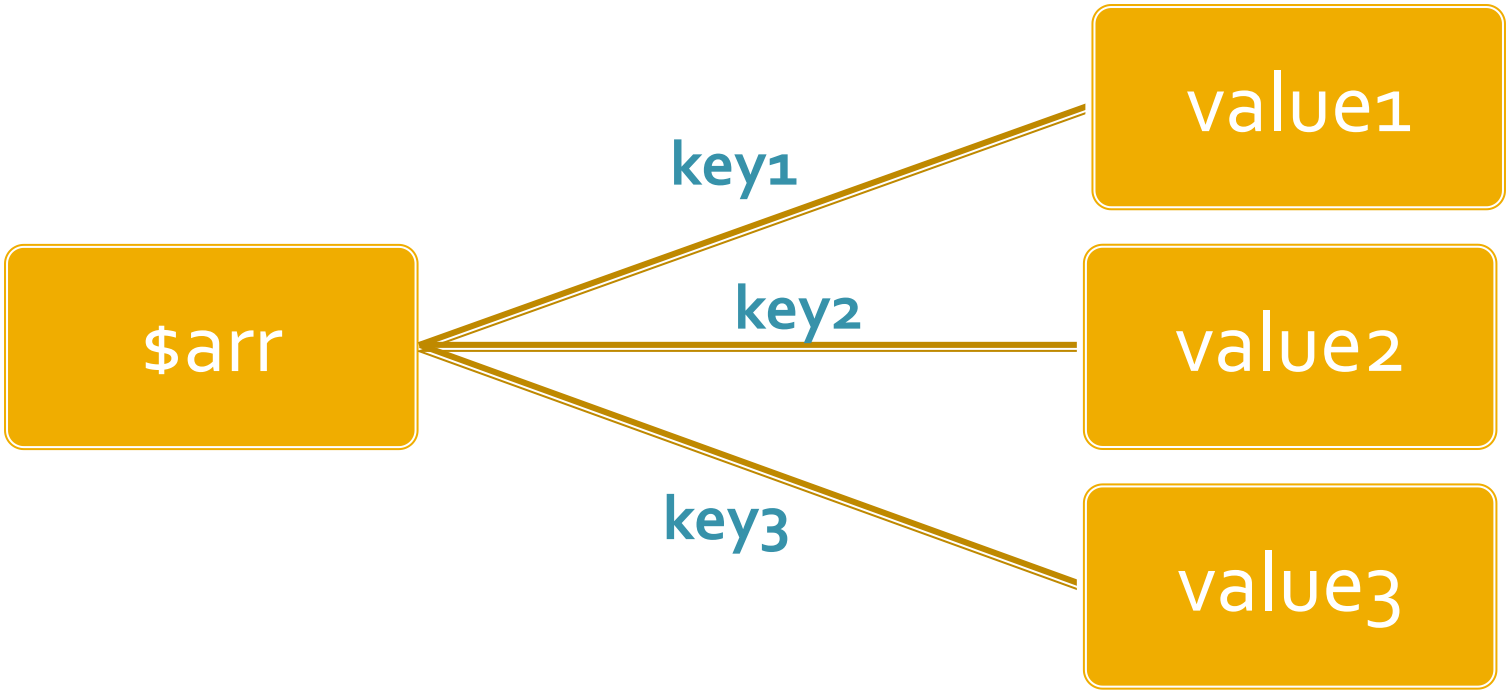
Arrays

Arrays in PHP

- An array in PHP is actually an ordered map. A map is a type that associates **values** to **keys**
- unlike C, Basic, **keys** are **not** required to be **integers**, can be **strings**
- default keys (if not otherwise specified) are consecutive integers with first key 0 (C syntax).
- defining a key / value pair
  - key **=>** value
- create an array
  - $arr = **array**("definition of key / value pairs")
  - pairs: key **=>** value, key **=>** value, …

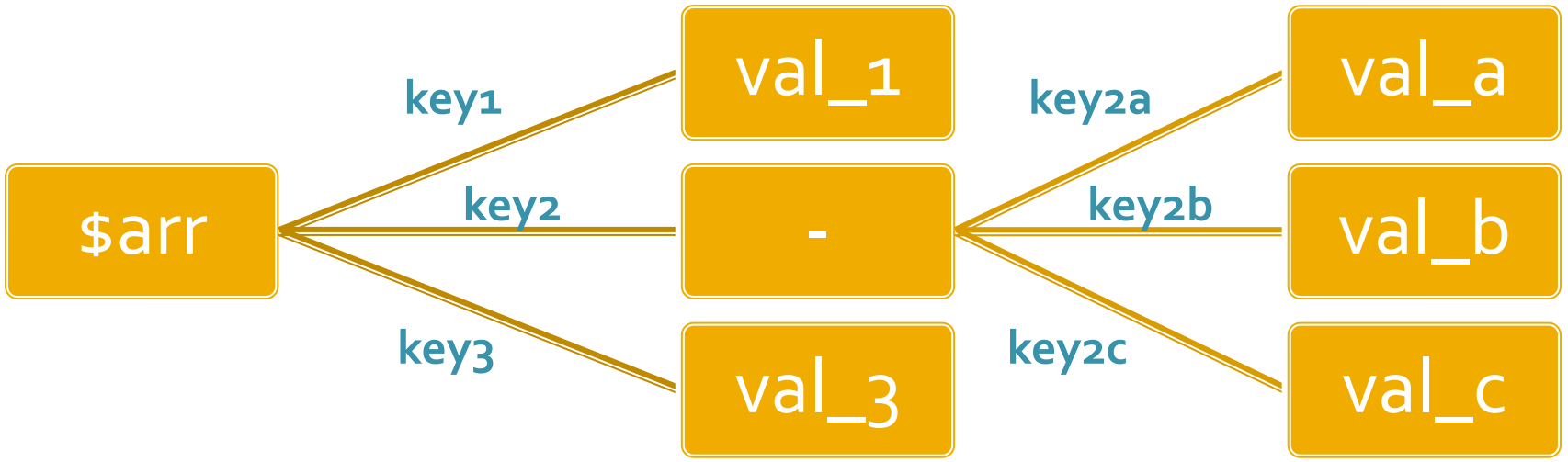
Array = Logical tree

- $arr = array(key1 => value1, key2 => value2, key3 => value3)

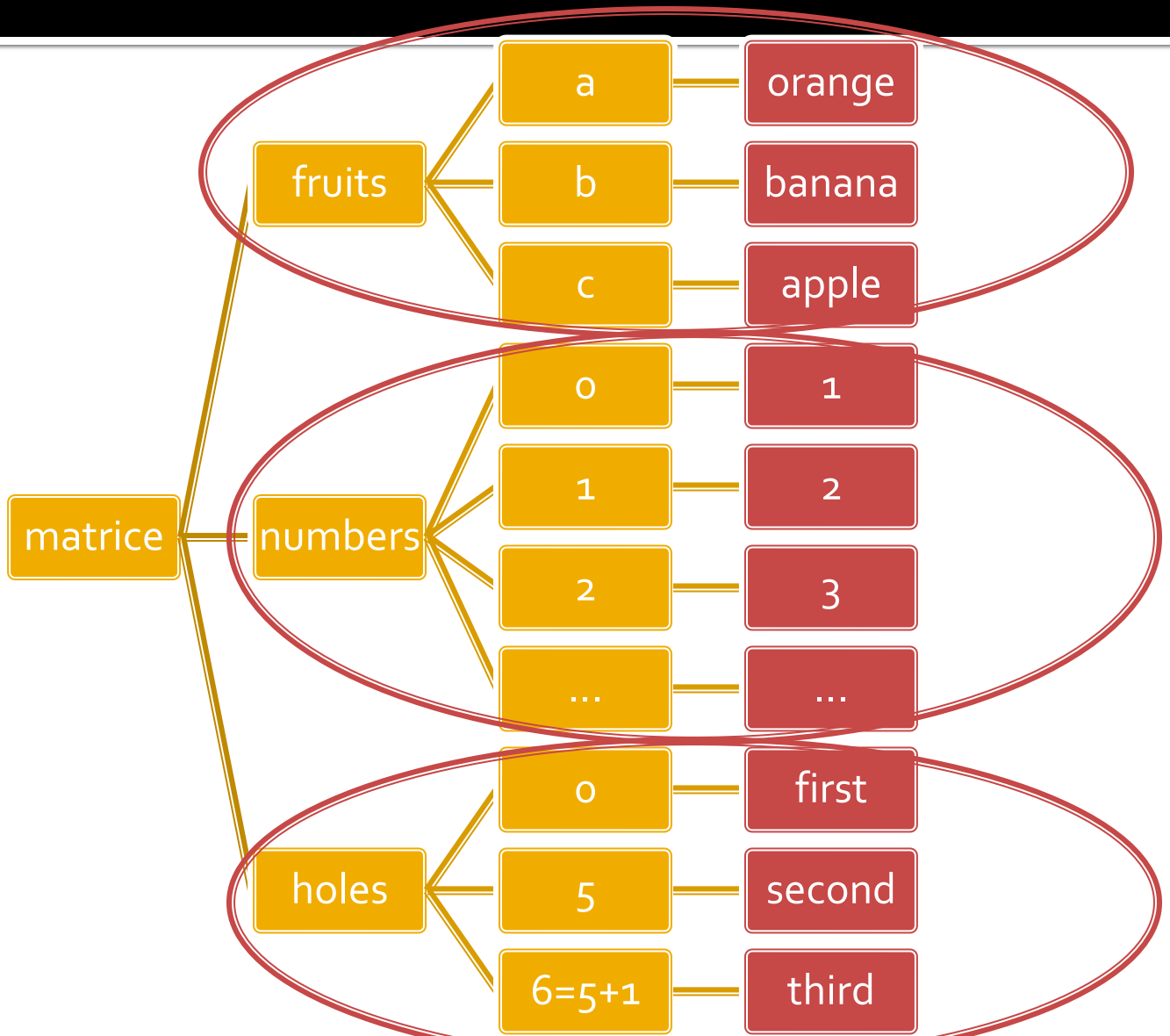


Array = Logical tree

- In particular, one or more of the values can in turn be an array, leading to **branching** of the tree
- $arr = array(key1 => val_1, **key2 => array**(key2a => val_a, key2b => val_b, key2c => val_c), key3 => val_3)



Arrays in PHP



View array content (debug)

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

```
$matr= array (
"fruits"  =>
array("a" => "orange", "b" => "banana", "c" => "apple" ,
"ultim"),
"numbers" =>
array(1, 2, 3, 4, 5, 6),
"holes"   =>
array("first", 5 => "second", "third")
);
echo $matr;
echo "<pre>";
print_r ($matr);
echo "</pre>";
```

```
Array

Array
(
    [fruits] => Array
        (
            [a] => orange
            [b] => banana
            [c] => apple
            [0] => ultim
        )

    [numbers] => Array
        (
            [0] => 1
            [1] => 2
            [2] => 3
            [3] => 4
            [4] => 5
            [5] => 6
        )

    [holes] => Array
        (
            [0] => first
            [5] => second
            [6] => third
        )

)
```

Foreach loop

- foreach (array_expression as $key => $value)
  statement
- foreach (array_expression as $value) statement
- foreach construct is used to loop through each key/value pair in an array
- On each iteration assign the current element's key to the local variable **$key** and the value of the current element is assigned to the local variable **$value** (scope: statement)
- foreach() works with a **copy** of the array, you cannot change the original array inside the statement
  - foreach ($arr as $key => $value) {
    $value = 'other value'; //**doesn't** work
    $arr[$key] = 'other value'; //works
    }

Example – foreach

```
$matr = array (
  "fruits"  => array("a" => "orange", "b" => "banana", "c" => "apple", "ultim"),
  "numbers" => "in loc de numere",
  "holes"   => "in loc de ce era"
);
foreach ($matr as $cheie => $continut)
            echo "matr[".$cheie."]=".$continut."<br />";
```

```
matr[fruits]=Array
matr[numbers]=in loc de numere
matr[holes]=in loc de ce era
```

PHP Global Variables - Superglobals

PHP Global Variables - Superglobals

- PHP Global Variables - Superglobals (predefined variables)
  - are always accessible, regardless of scope
  - Examples:
    - **$_SERVER** — Server and execution environment information
    - **$_GET** — HTTP GET variables
    - **$_POST** — HTTP POST variables
    - **$_FILES** — HTTP File Upload variables
    - **$_REQUEST** — HTTP Request variables
    - $_SESSION — Session variables
    - $_ENV — Environment variables
    - $_COOKIE — HTTP Cookies

Getting user submitted data

- When a user submits the data by clicking on "Submit", the form data is found in the file specified in the **action** attribute of the <form> tag in one of the superglobal variables:
  - $_POST – method="post"
  - $_GET – method="get"
  - $_REQUEST – both methods
- the superglobal variables are **arrays** with **string keys** controlled by the **name** attribute of the input element
  - <input type="text" name="**books_quant**" size="3" maxlength="3" />
  - $_POST['**books_quant**'] contains the user input in the receiving script

Organizing $_POST

- **name** attributes in the form inputs become **keys** in the superglobal array $_POST
  - <input type="text" name="**books_quant**" size="3" maxlength="3" />
  - $_POST['**books_quant**'] contains the user input
- creating **name** "array like", we can control branching of $_POST grouping input elements in the form as required
  - <input type="text" name="**quant[books]**" size="3" maxlength="3" />
  - $_POST ['**quant**'] ['**books**'] contains the user input

Organizing $_POST

- foreach($_POST as $value) { // do something with $value } will loop over all submitted data, including buttons, hidden fields
- for alternate contact data:
  - <input name="**adr1**" type="text" value="" />
  - <input name="**adr2**" type="text" value="" />
  - <input name="**tel1**" type="text" value="" />
  - <input name="**tel2**" type="text" value="" />
  - <input name="**email1**" type="text" value="" />
  - <input name="**emal2**" type="text" value="" />
- processing of different types difficult

Organizing $_POST

- for alternate contact data:
  - <input name="**adr[1]**" type="text" value="" />
  - <input name="**adr[2]**" type="text" value="" />
  - <input name="**tel[1]**" type="text" value="" />
  - <input name="**tel[2]**" type="text" value="" />
  - <input name="**email[1]**" type="text" value="" />
  - <input name="**emal[2]**" type="text" value="" />
- process only selected data
  - foreach(**$_POST["adr"]** as $adr_i) { // do something with $adr } will loop only over address data
  - foreach(**$_POST["tel"]** as $tel_i) { // do with $tel_i }
  - foreach(**$_POST["email"]** as $em_i) { // do with $em_i }

Accessing a MySQL server from PHP

Accessing a MySQL server from PHP

- PHP has two extensions in order to interact with a MySql server (local or remote), these must be activated in php.ini.
  - mysql
  - mysqli (improved: functions for MySql > 4.1)
- A database can be accessed if the PHP script knows a MySql server user with access rights
  - usually every application has its specific MySql user with specific access rights
- A database can also be created from PHP, but it is not the recommended method if it is not necessary
  - the code is difficult to implement and used only once

Accessing a MySQL server from PHP

- mysql_connect
  - Open a connection to a MySQL Server
  - resource **mysql_connect** (string server , string user, string password)
  - returns a MySQL link identifier on success or false on failure
- mysql_query
  - Send a MySQL query
  - resource **mysql_query** (string query [, resource link_identifier])
  - result
    - SELECT, SHOW, DESCRIBE or EXPLAIN: returns a resource or false
    - UPDATE, DELETE, DROP, etc: returns true/false

Accessing a MySQL server from PHP

- mysql_fetch_assoc
  - Returns an **associative array** that corresponds to the fetched row and moves the internal data pointer ahead, or **false** if there are no more rows. The **string** keys of the array are the field names (columns) in the DB table
  - array **mysql_fetch_assoc** (resource result)
- mysql_fetch_row
  - Returns an **numerical** array that corresponds to the fetched row, or false
  - array mysql_fetch_row (resource result)

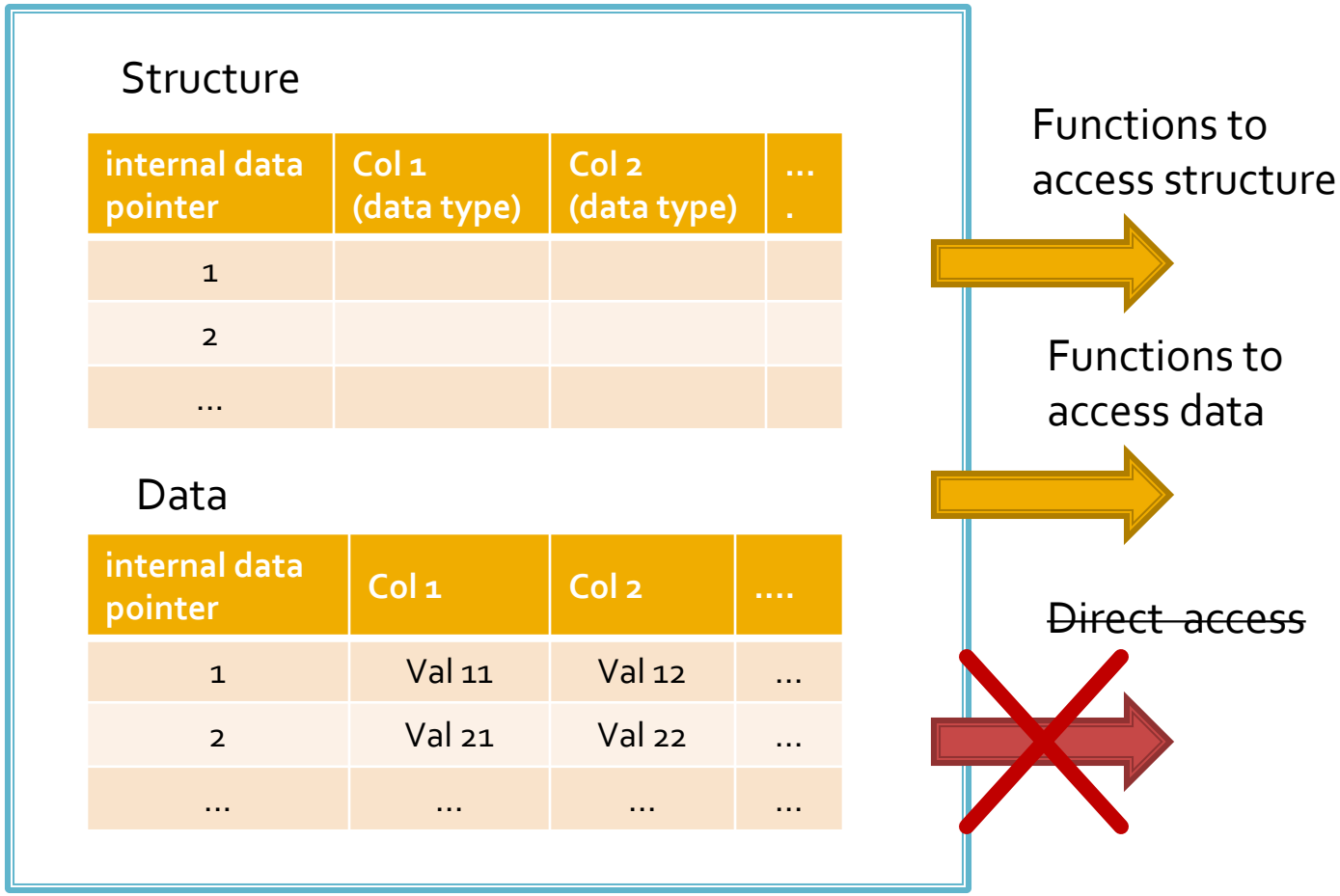
Accessing a MySQL server from PHP

- mysql_fetch_array
  - groups functionality of mysql_fetch_assoc and mysql_fetch_row
  - array **mysql_fetch_array** (resource result [, int result_type])
  - MYSQL_ASSOC, MYSQL_NUM, MYSQL_BOTH (default)
- mysql_data_seek
  - moves the internal row pointer of the MySQL result associated with the specified result identifier to point to the specified row number. The next call to a MySQL fetch function would return that row
  - bool **mysql_data_seek** (resource result, int row_number)

MySQL resources

- Resources are a combination between
  - Structured data (values + structure) resulted from a SQL query
  - functions to access those values/structure
- Analogy with OOP
  - a special "class" created following a SQL query with predefined procedures to access the results of that query

MySQL resources



MySQL resources

- Structure access functions are rarely used
  - most applications are designed on a fixed DB structure, and the structure of the received data is known
  - exception: general DB applications, eg: PhpMyAdmin
- Most data access functions are characterized by sequential access
  - the data is read line by line
  - simultaneously, internal data pointer advances to the next position, preparing the next read

MySQL resources

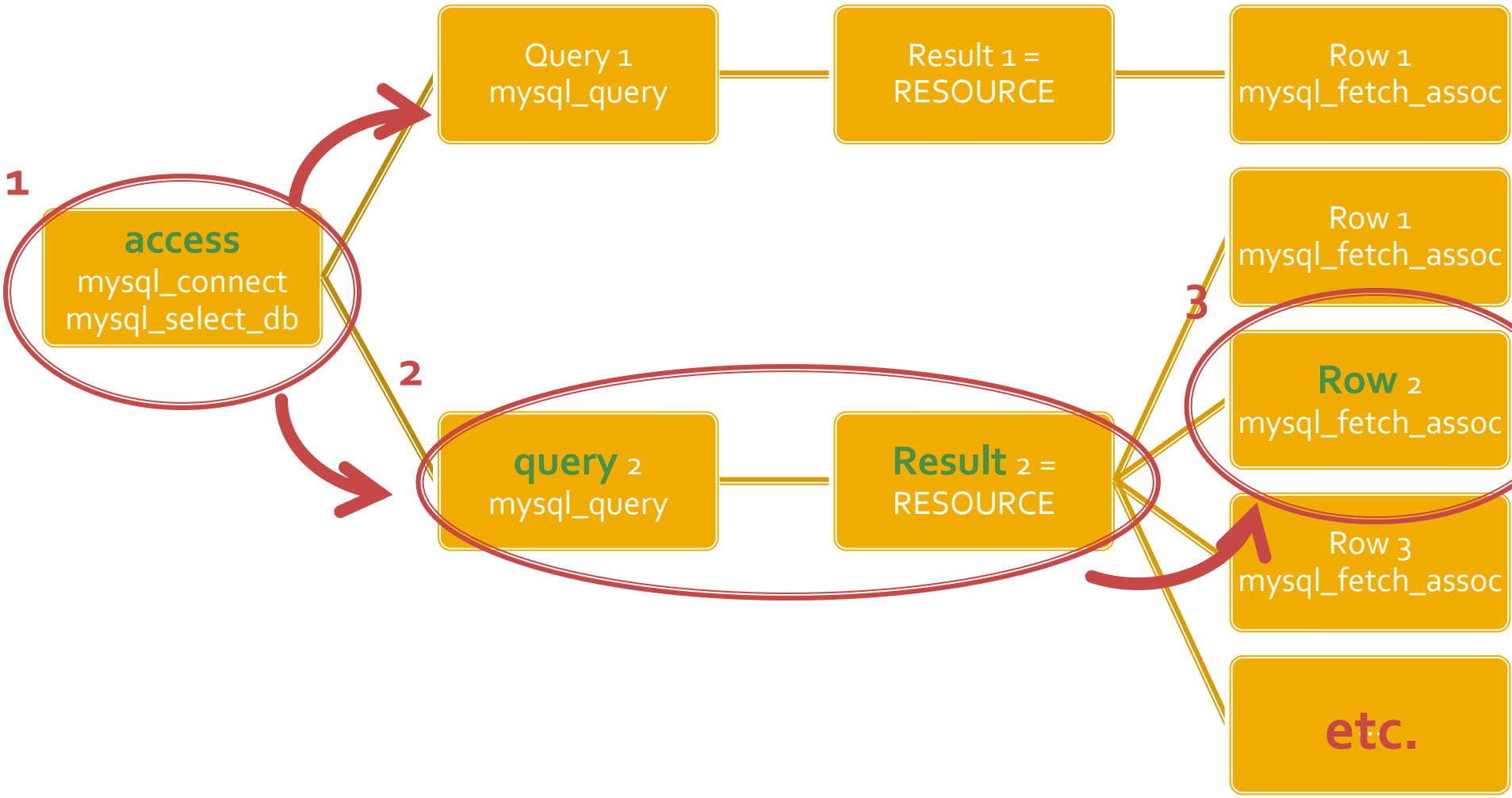
- Functions are optimized for use in a **do { } while()** loop or a **while() { }** loop
  - Returns **false** if there are no more rows
- typically we fetch a single row (mysql_fetch_assoc) followed by a **do { } while()** loop
  - to allow a "problem detection" code to run only once
  - or generate "single steps" for displaying a successful result (eg: table head)

Example

```
$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 );
```

Accessing a MySQL server from PHP



!! IMPORTANT

PHP > 5.5

PHP 5.5, 7, 8

- Starting with PHP 5.5.0 the mysql extension was **deprecated**
  - any function from this extension generates an **E_DEPRECATED** error/warning
  - Starting with PHP 7.0.0 the mysql extension was **removed**
- Instead we must use:
  - mysqli extension (MySQL Improved)
  - PDO extension (PHP Data Objects)

mysqli extension

- Other than enhanced security offers access to newer facilities of the DB server:
  - Prepared Statements (speed, security)
    - server side
    - client side
  - server Stored Procedures (speed, security)
  - Multiple Statements
  - Transactions (integrity)

mysqli extension

- Supports two interfaces
  - procedural interfaces (similar to mysql)
  - OOP (similar to PDO)
- Procedural interface (almost) identical to the original mysql extension
  - easy transition
  - small differences (parameter)

mysqli – Procedural

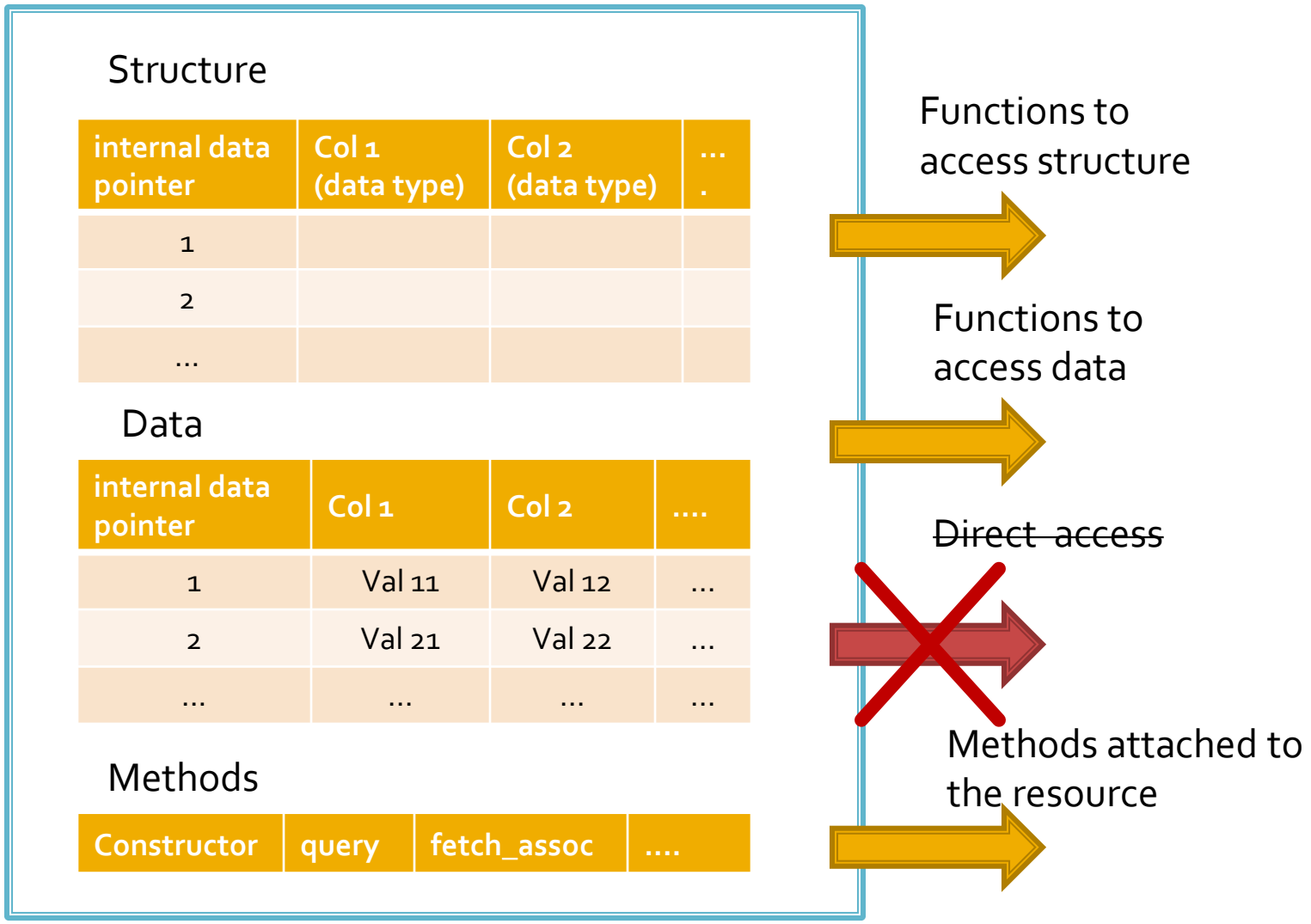
```php
<?php
$mysqli = mysqli_connect("example.com", "user", "password", "database");
$res = mysqli_query($mysqli, "SELECT 'Please do not use the mysql extension ' AS _msg FROM DUAL");
$row = mysqli_fetch_assoc($res);
echo $row['_msg'];

$mysql = mysql_connect("example.com", "user", "password");
mysql_select_db("test");
$res = mysql_query("SELECT ' for new developments.' AS _msg FROM DUAL", $mysql);
$row = mysql_fetch_assoc($res);
echo $row['_msg'];
?>
```

- all mysql functions have a mysqli equivalent
- most functions have the same parameters in the same order
- there are functions with small differences (Ex: **mysqli_connect**, **mysqli_query**)

mysqli – OOP

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

$mysql = mysql_connect("example.com", "user", "password");
mysql_select_db("test");
$res = mysql_query("SELECT ' for new developments.' AS _msg FROM DUAL", $mysql);
$row = mysql_fetch_assoc($res);
echo $row['_msg'];
?>
```

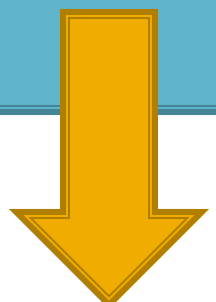
MySQL resources – mysqli



Conversion to mysqli (mandatory)

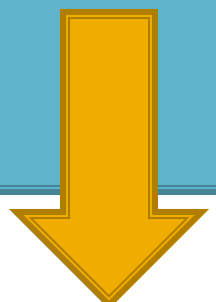
- example

```
$hostname = "localhost";
$database = "dbwpi";
$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 (OOP)

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

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

Textbooks

- https://www.php.net/
- https://rf-opto.etti.tuiasi.ro/master_it.php
  - **3 X PHP and MySQL Bible !!**

Template

Template

- simultaneous control of the esthetic and functional design for all pages in the site
- separation the application from the esthetic design

Example

Magazin online Firma X SRL

Lista Produse

Nr.	Produs	Pret
1	Carti	100
2	Caiete	50
3	Penare	150
4	Stilouri	125
5	Creioane	25
Comanda		

```
<html>
<head>
<title>Magazin online Firma X SRL</title>
</head>
<body bgcolor="#CCFFFF">
<table width="600" border="0" align="center">
<tr><td><img src="images/antet.gif" width="600" height="100" /></td></tr>
<tr><td height="600" valign="top" bgcolor="#FFFFCC">
Continut
</td></tr>
</table>
</body>
</html>
```

Control statements

- include()
- require()
- include_once()
- require_once()

- to insert the content of one PHP file (used as parameter) into another PHP file (that uses the include/require statement) before the server executes it
- **require** stops the execution of the current script if the parameter file is **not** found
- …**_once()** checks if the respective file has been included before and does not include it again

Example 2

- repeated sections can be moved to a separate file and inserted with require()
- first step: common areas are identified

```
<html>
<head>
<title>Magazin online Firma X SRL</title>
</head>
<body bgcolor="#CCFFFF">
<table width="600" border="0" align="center">
<tr><td><img src="images/antet.gif"
width="600" height="100" /></td></tr>
<tr><td height="600" valign="top"
bgcolor="#FFFFCC">
Continut
</td></tr>
</table>
</body>
</html>
```

Example 3

```
<html>
<head>
<title>Magazin online Firma X
SRL</title>
</head>
<body bgcolor="#CCFFFF">
<table width="600" border="0"
align="center">
<tr><td><img src="images/antet.gif"
width="600" height="100" /></td></tr>
<tr><td height="600" valign="top"
bgcolor="#FFFFCC">
Continut
</td></tr>
</table>
</body>
</html>
```

header.php

```
<html>
<head>
<title>Magazin online Firma X
SRL</title>
</head>
<body bgcolor="#CCFFFF"><?php
//orice cod comun PHP
?><table width="600" border="0"
align="center">
<tr><td><img src="images/antet.gif"
width="600" height="100" /></td></tr>
<tr><td height="600" valign="top"
bgcolor="#FFFFCC">
<h1>Magazin online Firma X SRL</h1>
```

footer.php

```
</td></tr>
</table>
</body>
</html>
```

Using a template

- header.php
  - any common structure (HTML) code
  - any common application code (PHP) – almost all pages in an application need:
    - data access
    - check access rights
    - constant definitions
    - define/load data **from** session ($_SESSION)
- footer.php
  - any common structure (HTML) code
  - any common application code (PHP) – usually less:
    - save data **into** the session ($_SESSION)

Template

- Any php file in my application:
  - <?php require('header.php');?>
  - <?php require('footer.php');?>
- and automatically that file has the same esthetic and functional design

*.php

```
<?php require('header.php');?>
<h2>Lista Produse</h2>
<table border="1">
...
</table>
<?php require('footer.php');?>
```

Advantages working with template

- speed of application development
- clear separation of the application from the form
- unitary form
  - "don't make me think"
- simultaneous control of the esthetic and functional design for all pages in the site
- defining common data in a single file
  - define('BOOK_PRICE',100);

Active Links

Methods

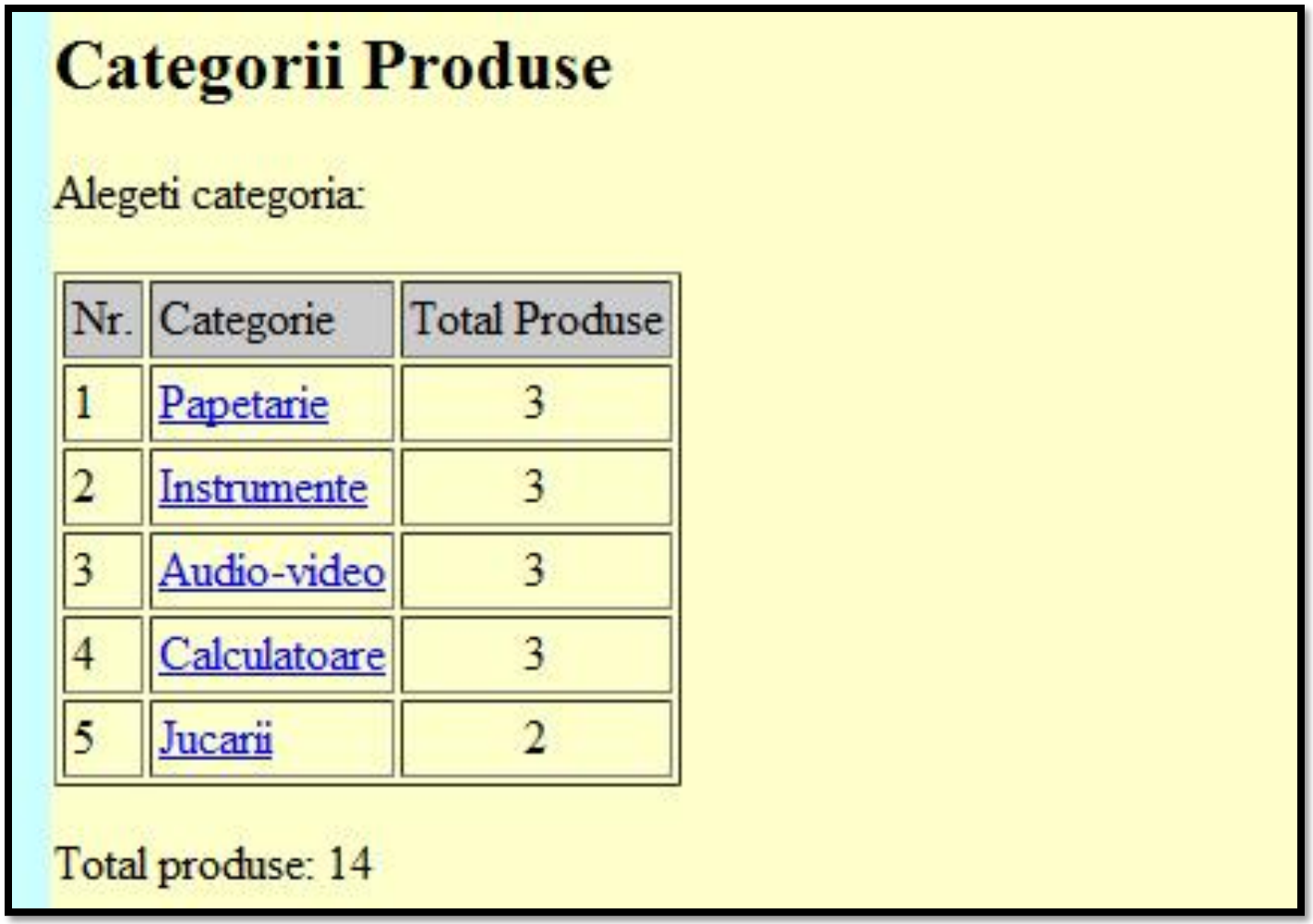
- **post** : data is transmitted as a block (inside the body of the HTTP request)
- **get** : appends form-data into the URL : results.php?prob=81&an=2009
- **get** must be used only for "idempotent" data,
  - no collateral effects
  - no change in server status (databases, etc)
- we can emulate a form (**get**) by writing links appropriately

Active Links

- used to send specific **information** to the target file
- in file1.php
  - <a href="file2.php?categ=<?php echo $cat;?>"> <?php echo $cat;?>
- in file2.php
  - $_GET['categ']="value $cat associated to that specific link"

$cat – $_GET

Active Links



Categorii Produse

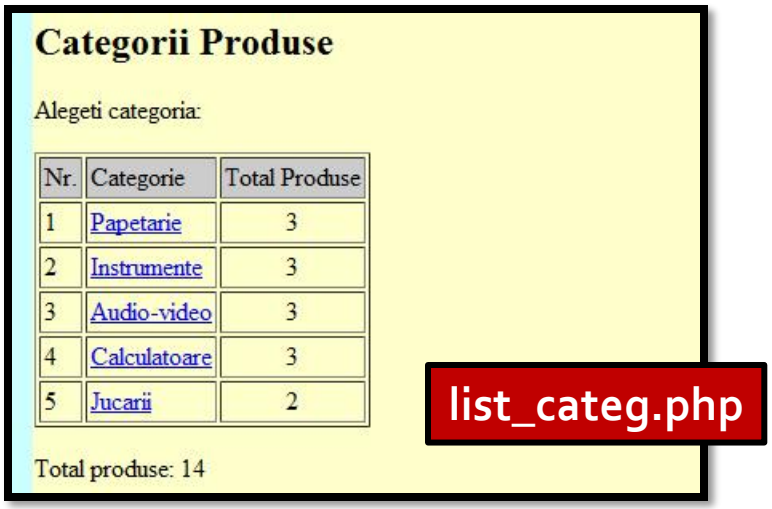
Alegeti categoria:

Nr.	Categorie	Total Produse
1	Papetarie	3
2	Instrumente	3
3	Audio-video	3
4	Calculatoare	3
5	Jucarii	2

Total produse: 14

Application flowchart

Rudimentary online shop app



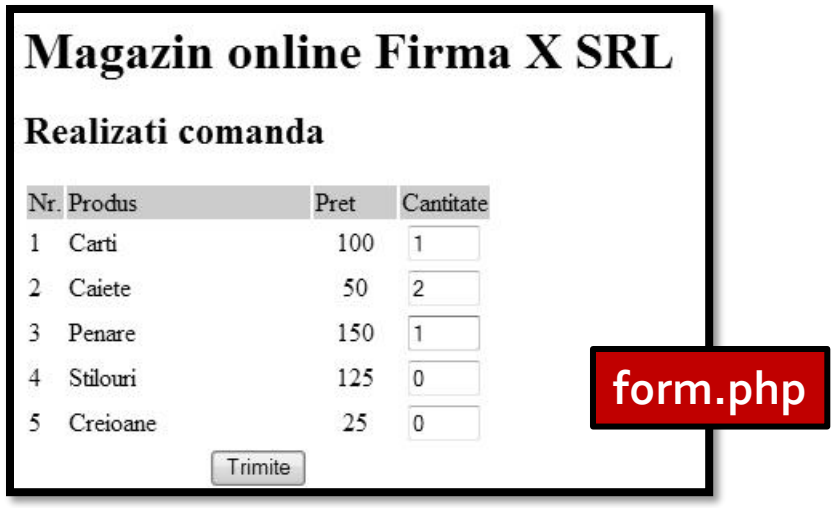
Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	Papetarie	3
2	Instrumente	3
3	Audio-video	3
4	Calculatoare	3
5	Jucarii	2

Total produse: 14

list_categ.php



Magazin online Firma X SRL

Realizati comanda

Nr.	Produs	Pret	Cantitate
1	Carti	100	1
2	Caiete	50	2
3	Penare	150	1
4	Stilouri	125	0
5	Creioane	25	0

Trimite

form.php

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

result.php

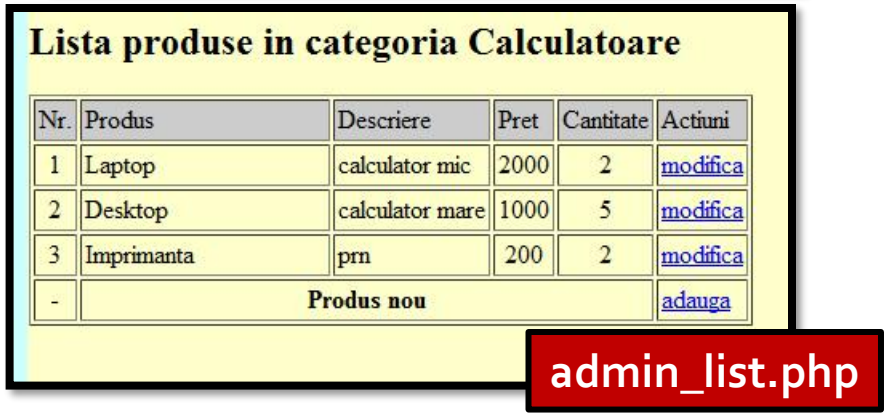
Application flowchart – Buyer

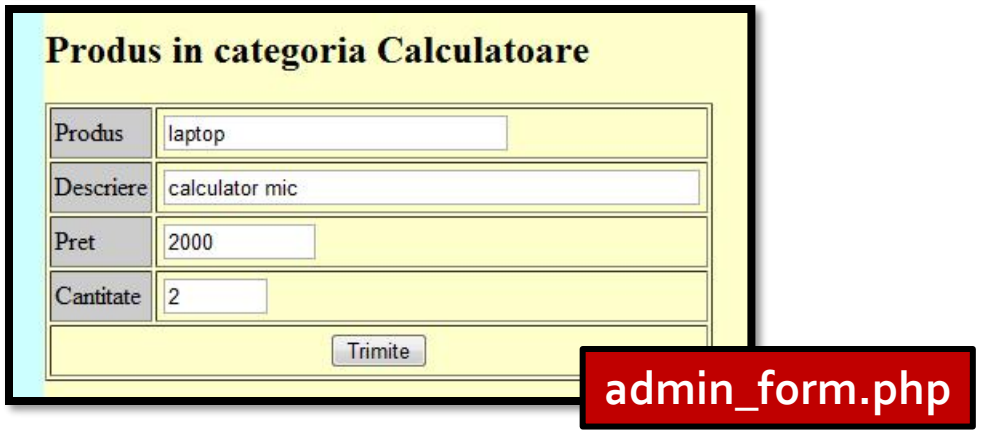
- As the application leaves a linear thread of execution, it is necessary to introduce a flowchart (tree) of the application
- Buyer
  - reading the required data (database access) is done in header.php, common for all files

Application flowchart – Seller

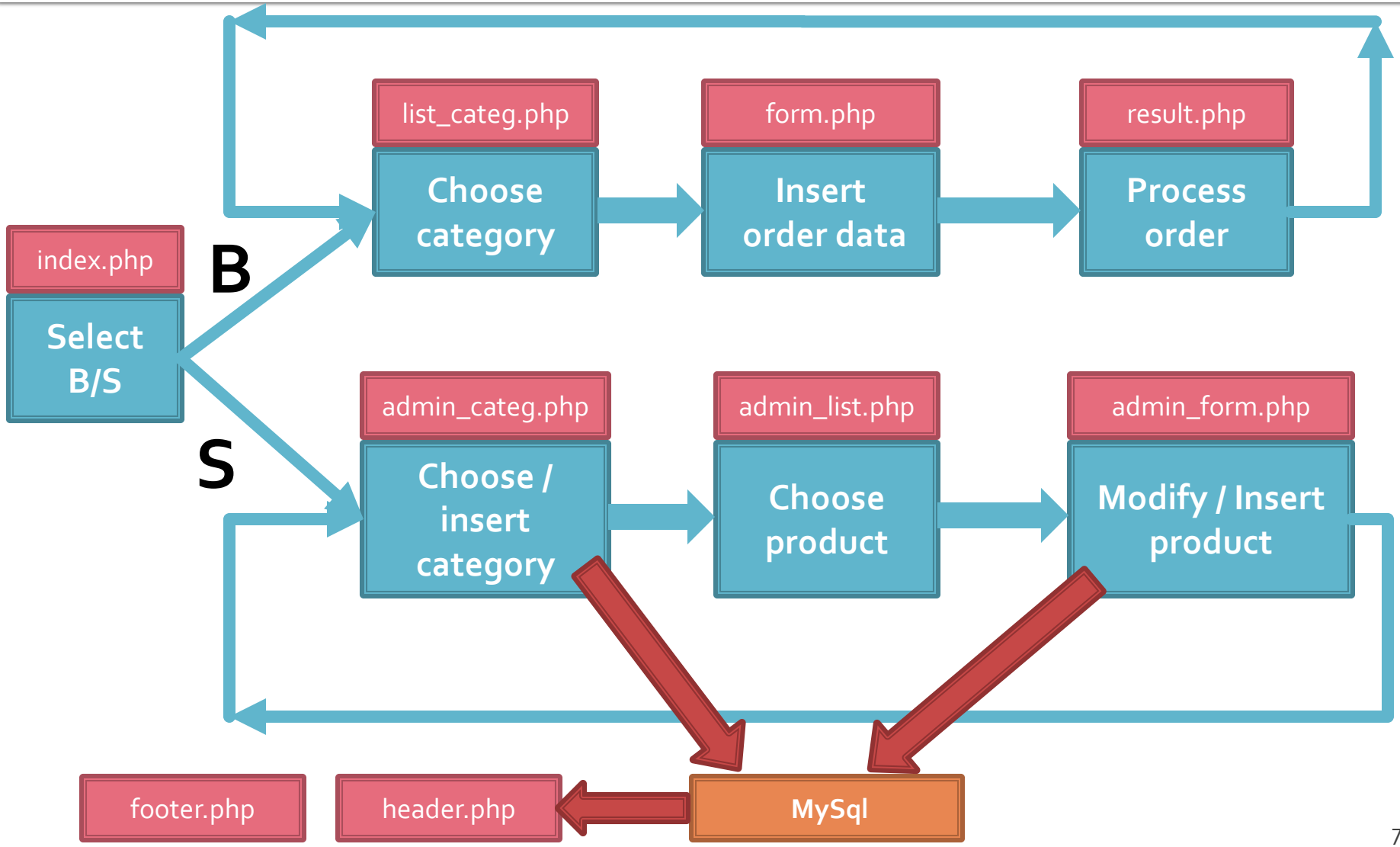
- The appearance of the application for the seller
  - introduces a parallel thread of execution with the necessity of the initial choice: buyer/seller
  - brings the possibility of writing in the database
  - various writing operations
    - insert new product category
    - insert new product in an existing category
    - modify existing product
  - writing in the database involves 2 actions:
    - collection of raw data from the user
    - data processing

Seller thread for online shop app



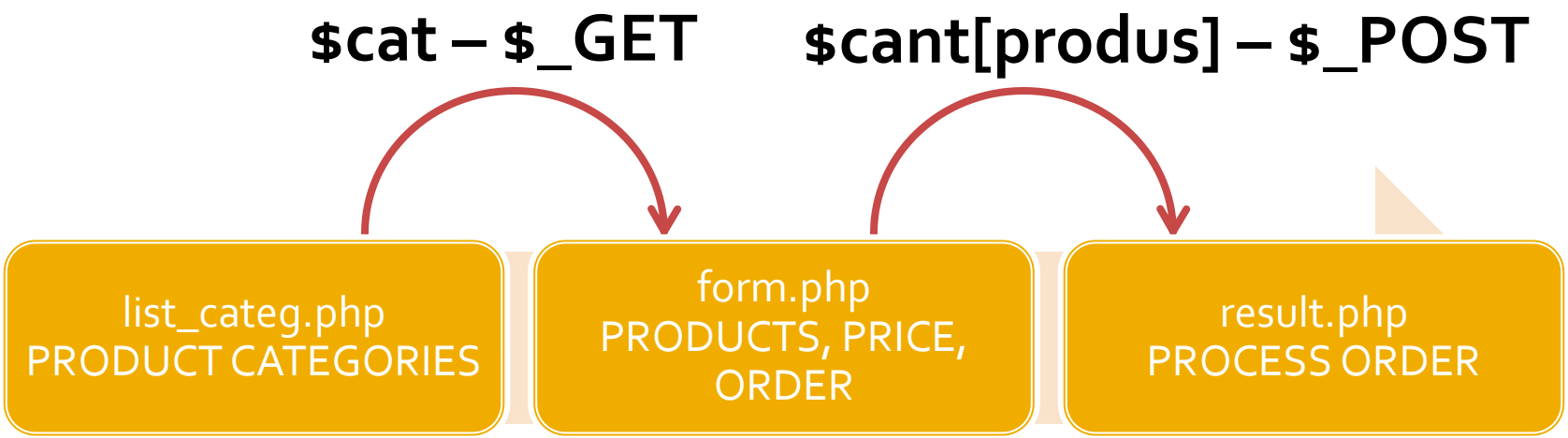


Application flowchart



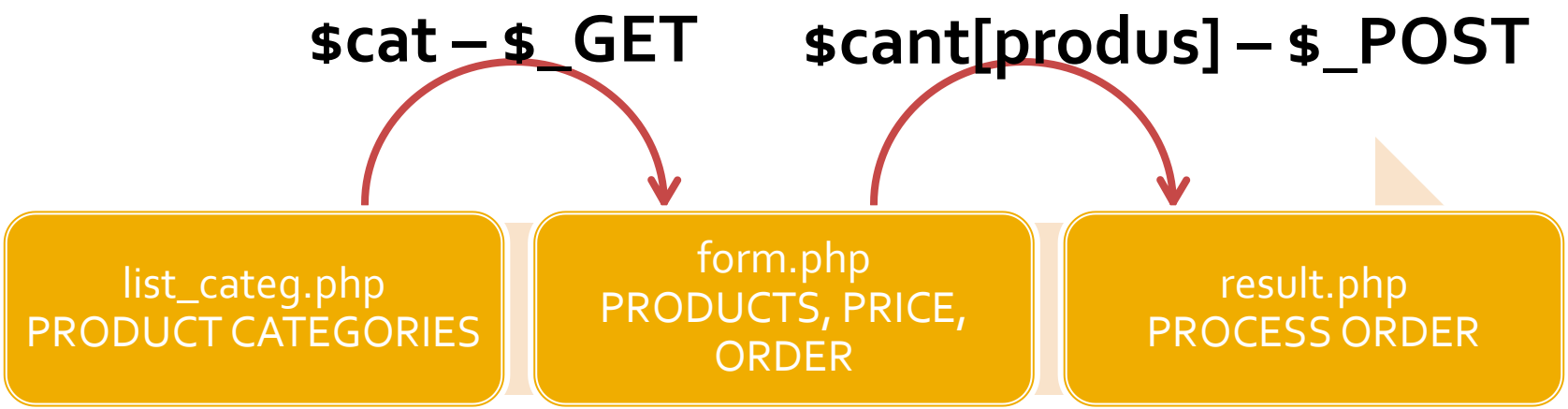
Application flowchart

- The application flowchart must also include information related to :
  - **what** data is transmitted between the different pages
  - **how** data is transmitted between pages

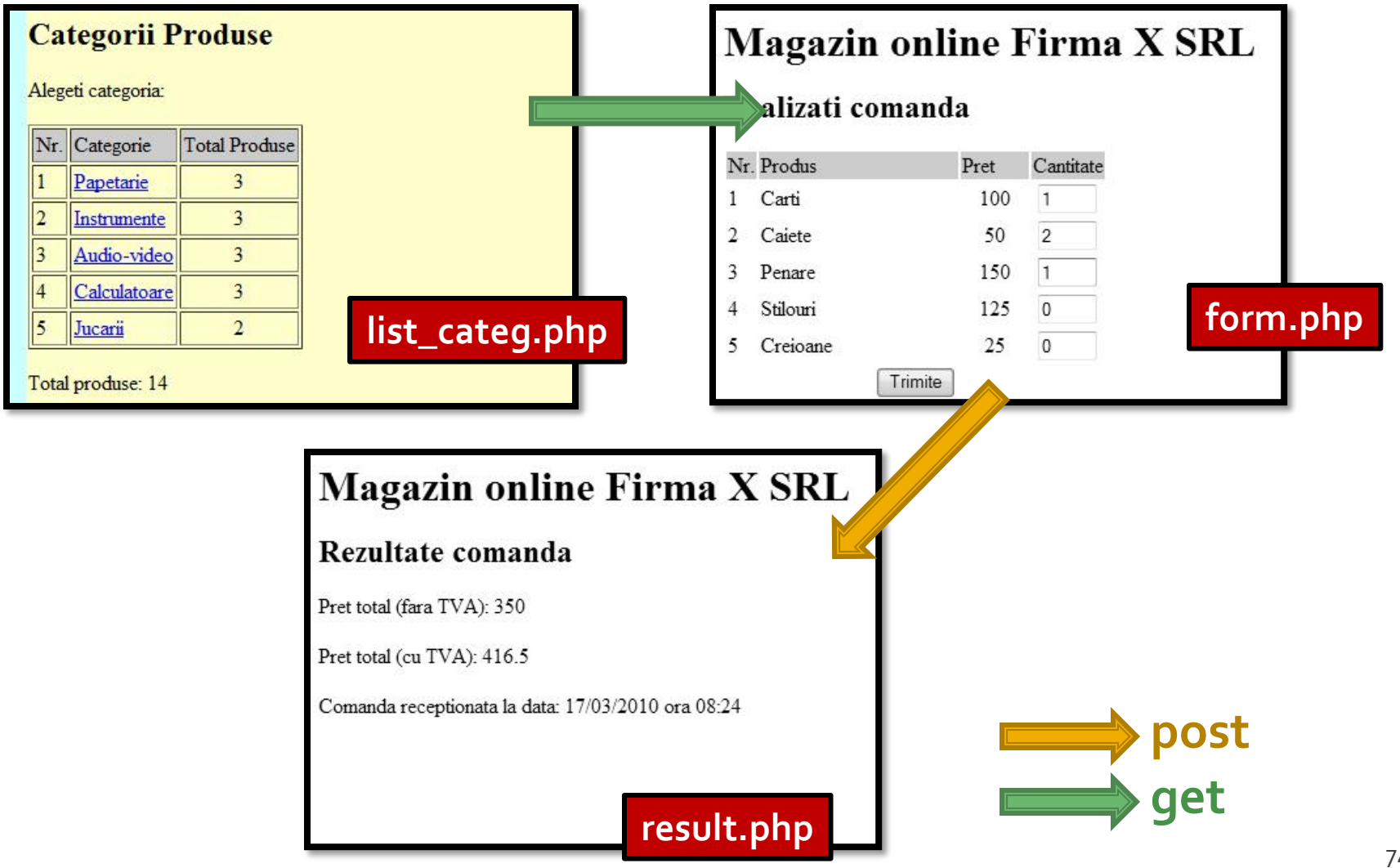


Application flowchart

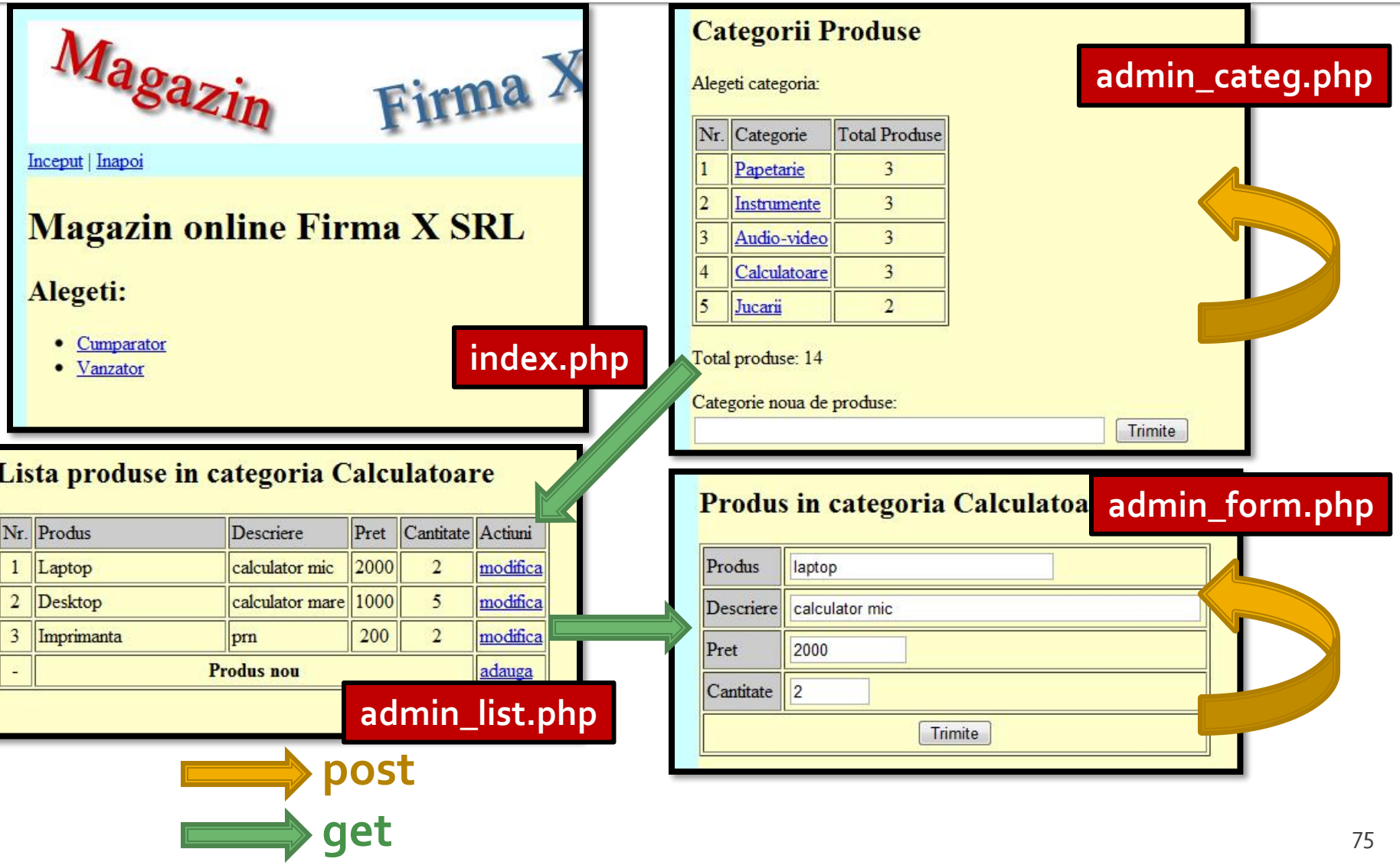
- Application flowchart – Example
  - the list of categories will transmit a single variable to the next file so we can use "active links", get method, **$_GET** in next file
  - the order form transmits multiple data included in a form, so the transmission is done with post method, **$_POST** in next file
- The choice of $_GET/$_POST has implications both in:
  - the page that transmits the data
  - as well as on the page that receives them



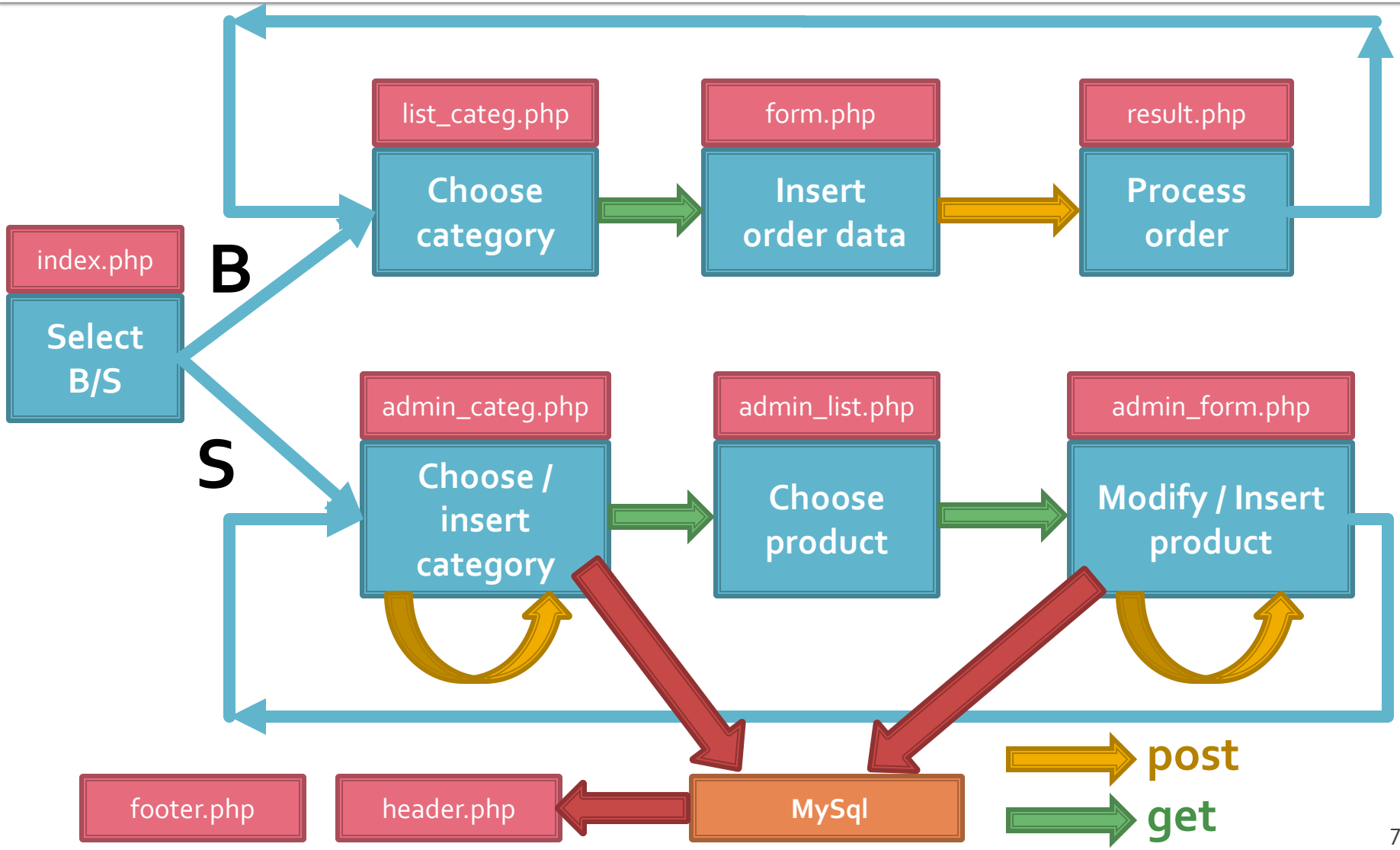
Flowchart (Buyer)



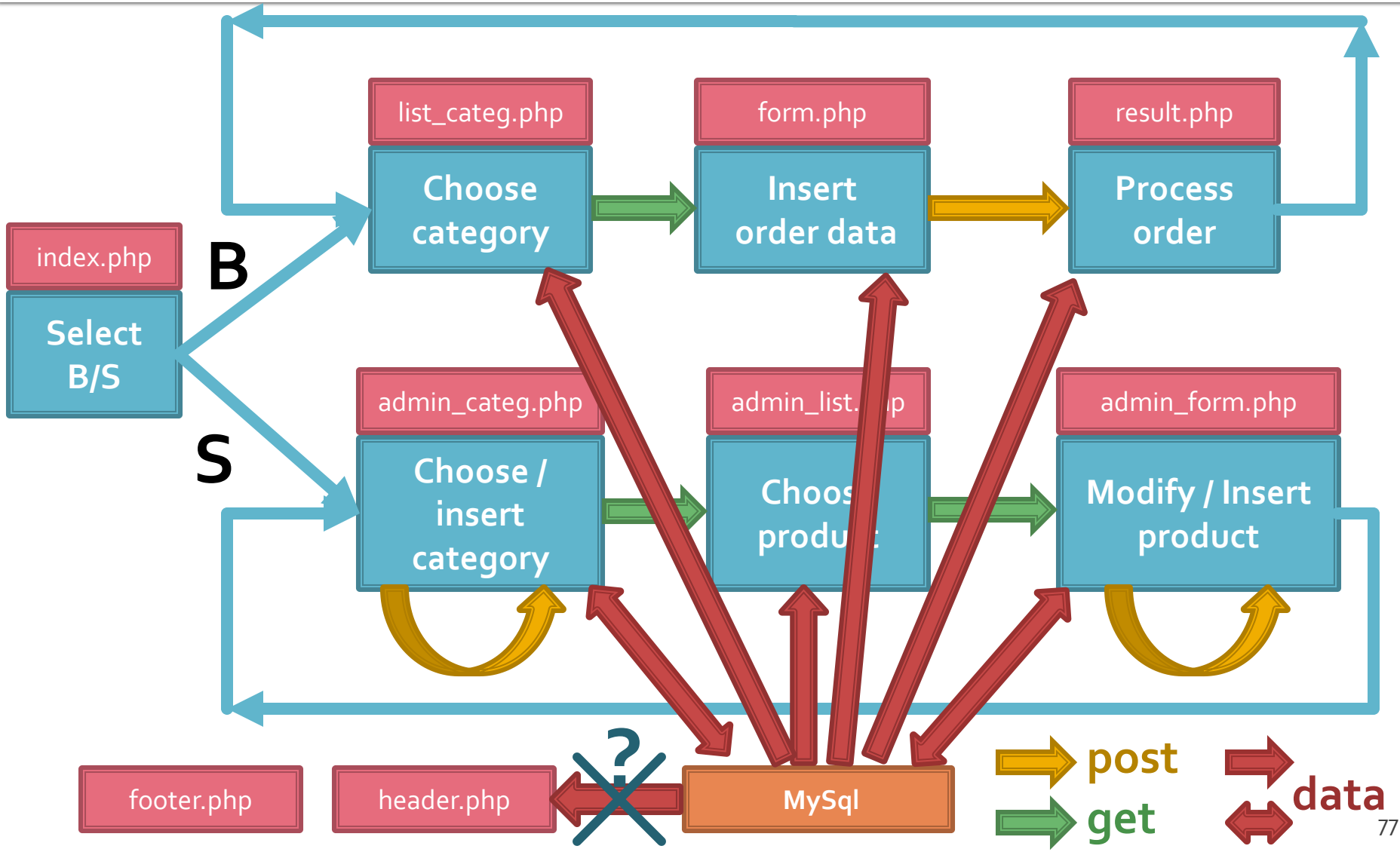
Flowchart (Seller)



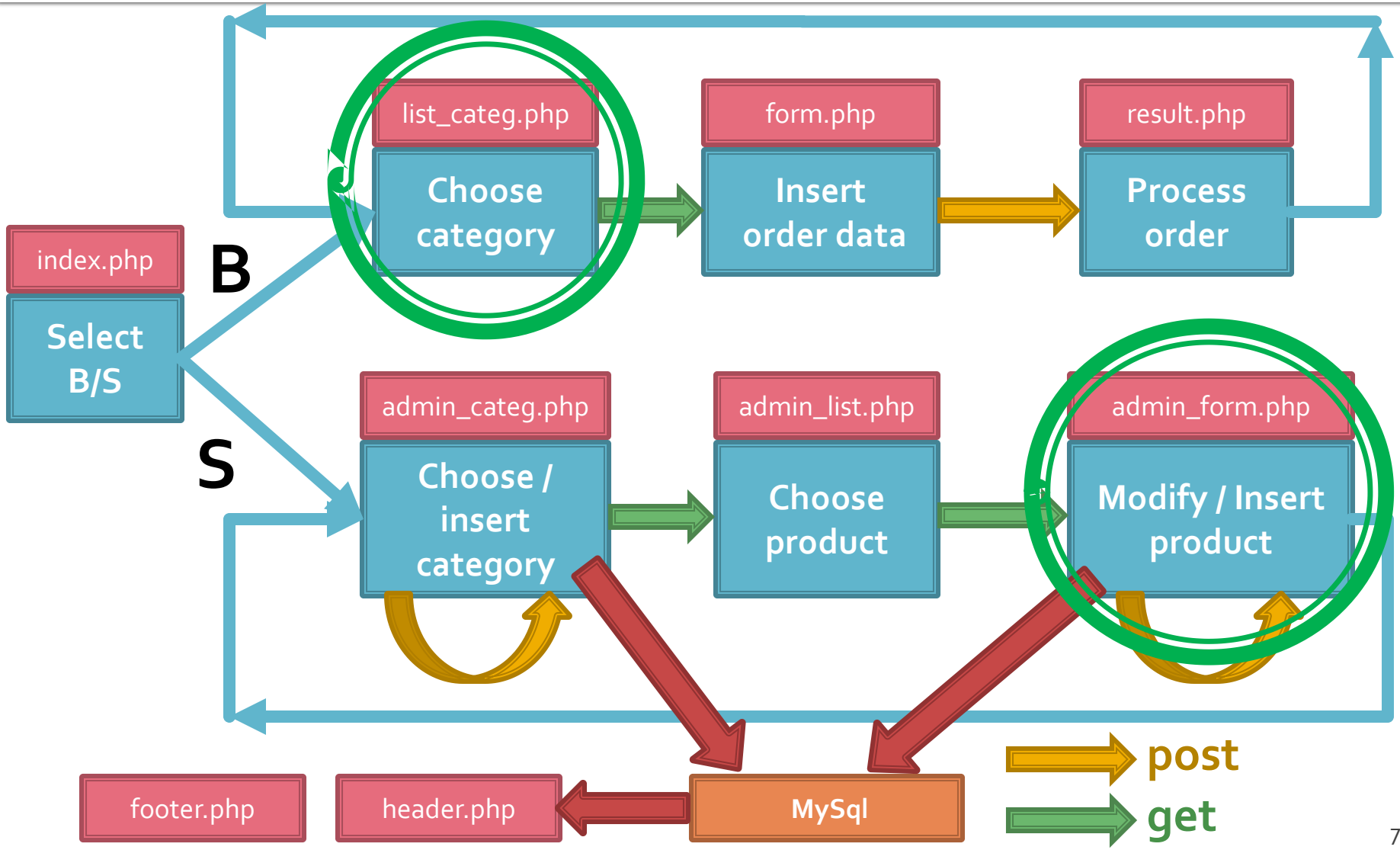
Complete application flowchart



Optimal application flowchart

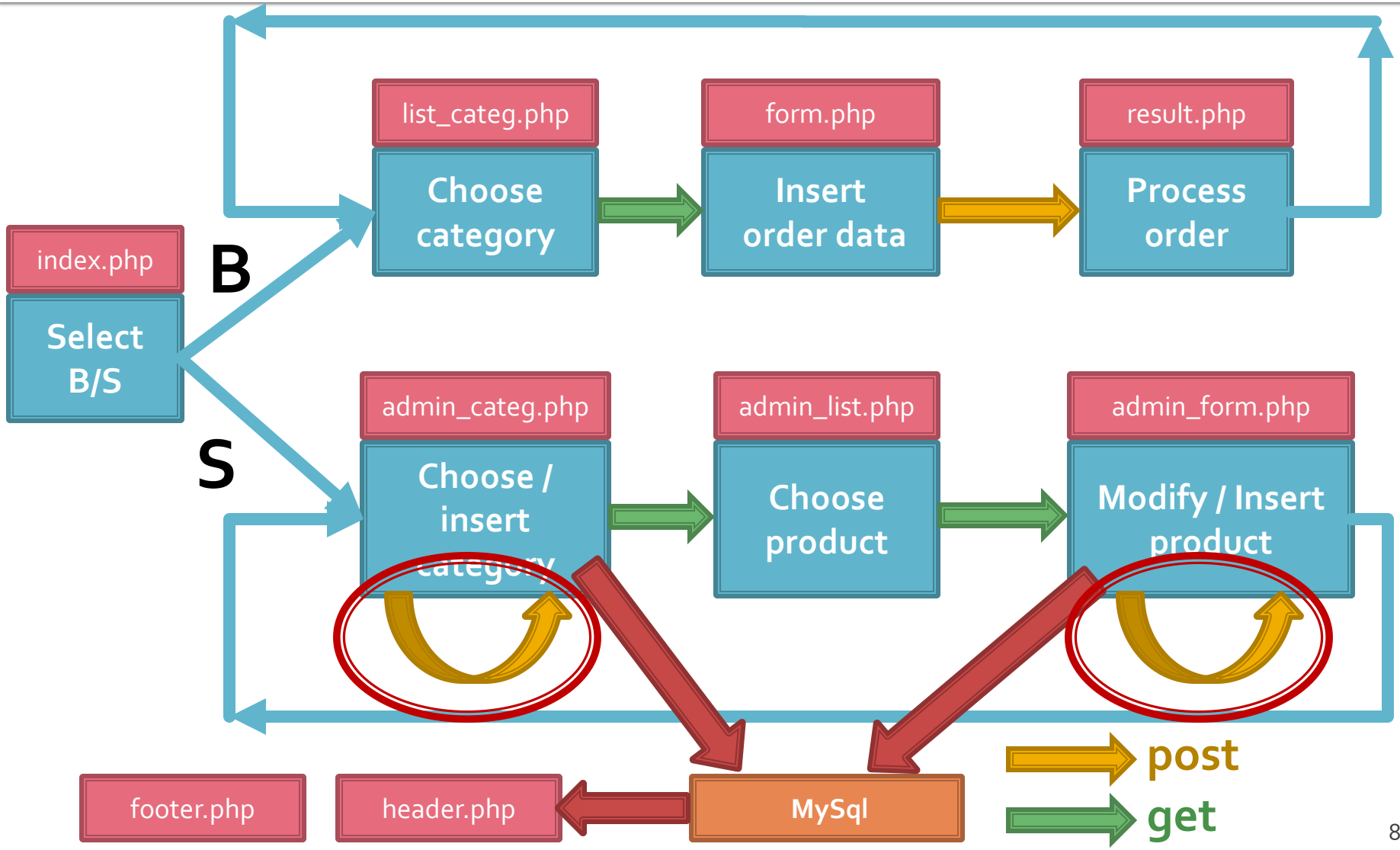


Individual assignment

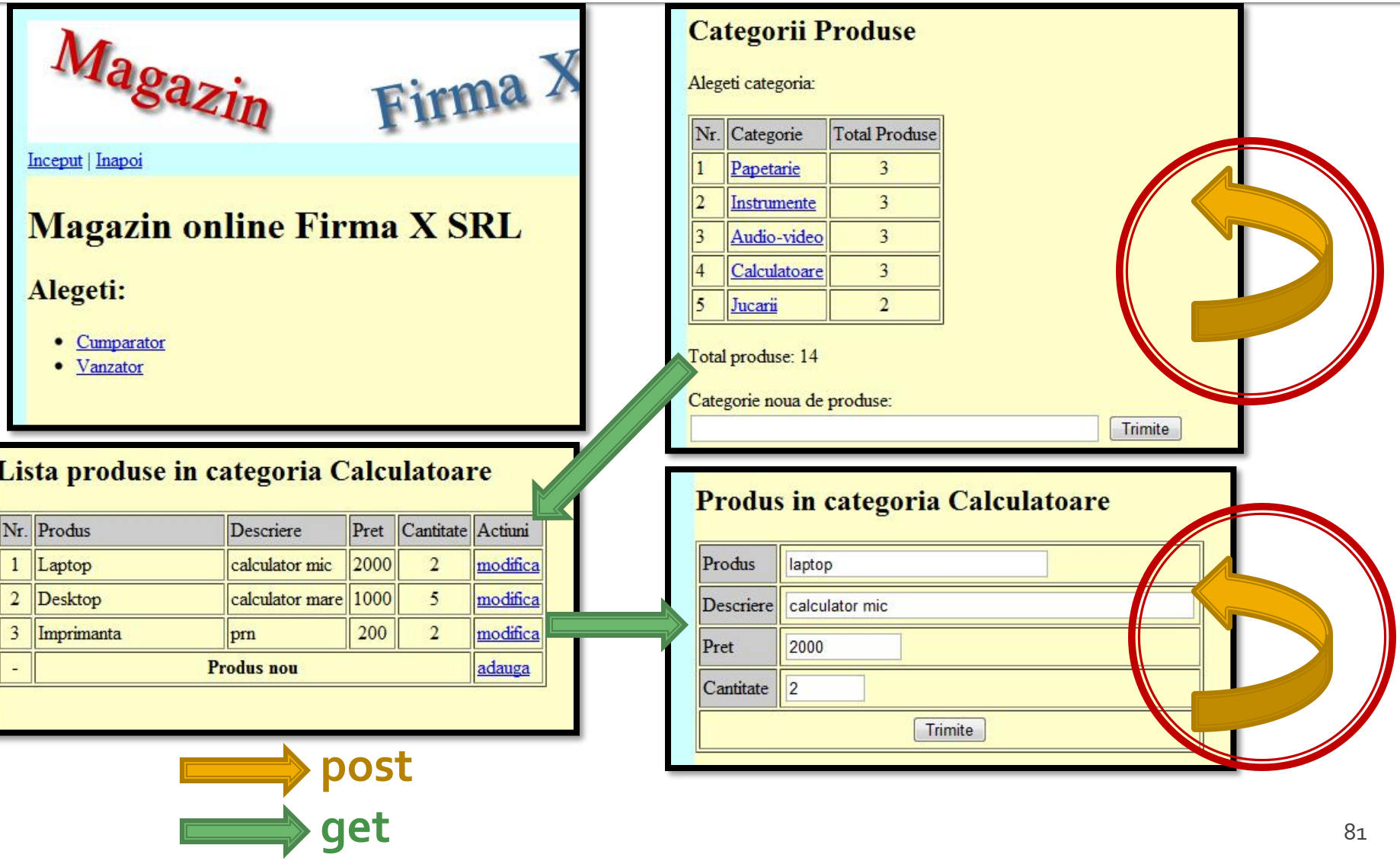


Single file for data collection AND processing

Complete application flowchart



Flowchart (Seller)



Single file for data collection AND processing

- This option is often preferred
- It allows the unitary preservation of all operations for the performance of an action
  - easier access
  - ease of programming
  - avoiding errors: File does not exist: /Server/...
- The same file is initially used to collect data and then, if their presence is detected, for their processing

Single file for data collection AND processing

- The "action" file for <form> will be the current file
- it is recommended to use the global variable $_SERVER['SCRIPT_NAME']
  - flexibility when renaming files
- alternatively $_SERVER['PHP_SELF'] is not recommended
  - security issues
- The data collection section is displayed only in the absence of data

```
<form action="<?php echo $_SERVER['SCRIPT_NAME '];?>" method="post">
<p><input name="date_ok" type="submit" value="Trimite" /></p>
</form>
```

Single file for data collection AND processing

- The detection of the existence of the data is done by checking the existence (**isset**($variable)) of the user inserted values
  - for extra protection, their content can also be checked

```
if (isset($_POST[" date_ok "]))
        { //date trimise
        if ($_POST[" date_ok "]=="Trimite" )
                { // data sent by the current file
                // data processing
                }
        }
else
        {
        // data collection
<form action="<?php echo $_SERVER['SCRIPT_NAME '];?>" method="post">
<p><input name="date_ok" type="submit" value="Trimite" /></p></form>
        }
```

PHP Debug

View array content (debug)

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

```
$matr= array (
"fruits"  =>
array("a" => "orange", "b" => "banana", "c" => "apple" ,
"ultim"),
"numbers" =>
array(1, 2, 3, 4, 5, 6),
"holes"   =>
array("first", 5 => "second", "third")
);
echo $matr;
echo "<pre>";
print_r ($matr);
echo "</pre>";
```

```
Array

Array
(
    [fruits] => Array
        (
            [a] => orange
            [b] => banana
            [c] => apple
            [0] => ultim
        )

    [numbers] => Array
        (
            [0] => 1
            [1] => 2
            [2] => 3
            [3] => 4
            [4] => 5
            [5] => 6
        )

    [holes] => Array
        (
            [0] => first
            [5] => second
            [6] => third
        )

)
```

Verify/debug PHP code

- It is recommended to use the array visualization option
  - In the file that receives the data
  - temporarily until the final version of the code
- the use of "verbose" code (manual) in the initial stages of writing PHP code can be extended to other types of data
  - the only (almost) debugging method in PHP
  - temp <?php echo "a=";echo $a; ?>

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

Debug

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

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

```
echo "<pre>".print_r ($_GET,true)."</pre>";
```

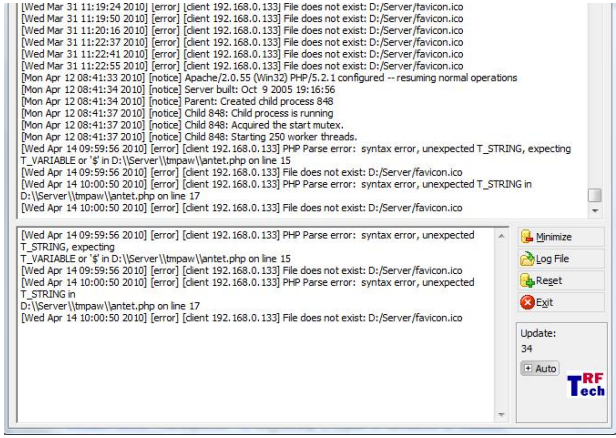
Recommended practical aspects in creating web applications

Recommended methods 1

- If you do not have easy access to the "logs" of the **MySql** server, you can see how the queries actually reach it by temporarily displaying the query text
  - $query = "SELECT * FROM `produse` AS p WHERE `id_categ` = ".$row_result_c['id_categ'];
    echo $query; // useful during testing
    - The text processed by PHP of the query will be clearly displayed on the page, making it easier to debug the program
    - These lines **must** be removed in the final form of the program as a security measure

Recommended methods 2

- Checking the error "log" of the Apache server and of the PHP interpreter is one of the main method of debugging PHP code.
  - Centos 7.1:
    - putty → nano /var/log/httpd/error_log
    - http://192.168.30.5/logfile.php (nonstandard)
    - supplemental homework (php.ini + log PHP **recommended**)



Recommended methods 3

- During the finalization of the MySql queries, it is often beneficial to first use **MySql Workbench / PhpMyAdmin** to test the queries, and then, when you are satisfied with the result, transfer the SQL query to the PHP code

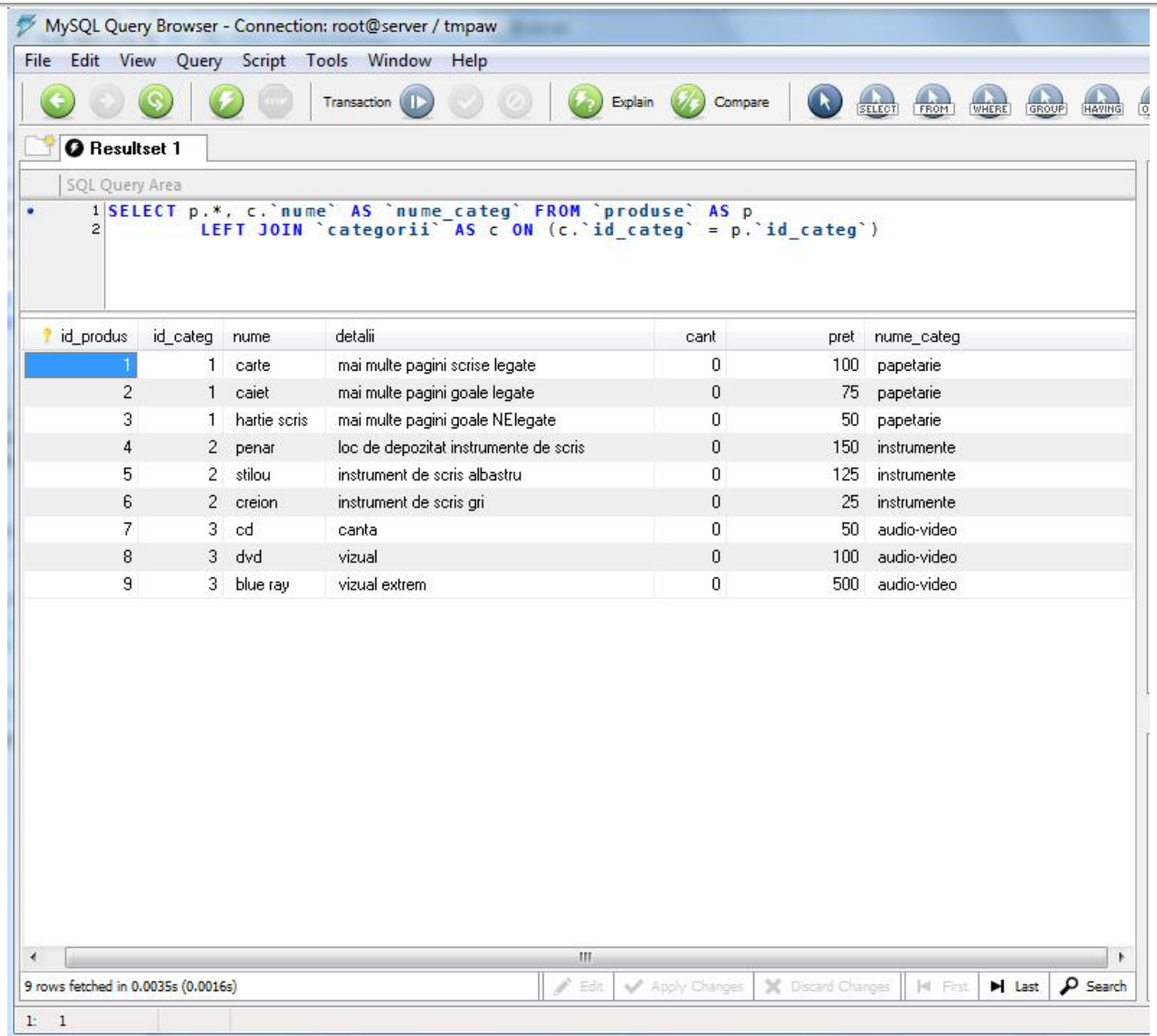
Resultset 1

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_produs	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

Recommended methods 3



Recommended methods 4

- the efficiency of a web application
  - 100% - **all processing "moved" to RDBMS**
  - PHP only to **move** and **display** data
- efficiency of a MySql application
  - 25% **correct choice of data types**
  - 25% **creating the necessary indexes in the applications**
  - 25% **correct normalization of the database**
  - 20% **increase in query complexity to "move" processing to the database server**
  - 5% **correct writing of queries**

Recommended methods 5

- When implementing a new application (project)
  1. Imagining the application flowchart (ex: S77)
     - "how would I like to work with such an application"
     - paper/pencil/time - essential
  2. Data identification/data transmission between pages
     - get/post/single file collection-processing
     - read/write database
  3. Identification of the logical structure of the data
     - "classes" of objects/phenomena treated identically
     - scalability is taken into account (the possibility of increasing the number of elements in a class)

Recommended methods 5

- When implementing a new application (project)
  4. Implementation of the database structure
     - In general, a table for each distinct logical class **BUT…**
     - scalability is taken into account (if the application grows the number of classes/tables **WILL NOT** increase) **AND…**
     - normalization
  5. Identifying the data type required for the columns
     - preferably use integers in any situation that requires ordering
     - the size of the fields not larger than necessary (it can be forced by the "size" attribute in the HTML "input" tag)
  6. Imagining the form of the pages
     - "I've seen it like this before and I liked it" (Don't make me think!)
     - investigating the possibility of introducing template functionality

Recommended methods 5

- When implementing a new application (project)
  7. Populate (manual) the database with initial data
     - MySql Query Browser (or PhpMyAdmin) / automatic / imported
     - the individual programming of some pages needs the preexistence of some data
  8. Individual programming of pages
     - Generally, in the order from the application flowchart (often a page provides the necessary data for the next one in the plan)
     - "verbose" mode active for PHP (ie: echo $a; print_r($matr))
  9. Preparation for distribution / move on production server
     - detailed testing (possibly a "guinea pig")
     - elimination of "verbose" additions
     - backup
     - generating an eventual install/setup

Tehnici PHP avansate

HTTP headers

- Permite transmiterea unor header-e specifice protocolului HTTP
- Structura mesajului
  - <initial line, different for request vs. response>
  - Header1: value1
  - Header2: value2
  - Header3: value3
  -
  - <optional message body goes here, like file contents or query data; it can be many lines long, or even binary data $&*%@!^$@>

HTTP headers

- header(string, code)

```php
<?php header("HTTP/1.0 404 Not Found");?>

<?php header("Location: http://www.example.com/");
/* Redirect browser */?>
```

```
<meta http-equiv="refresh" content="5;
url=http://www.example.com/">
```

HTTP headers

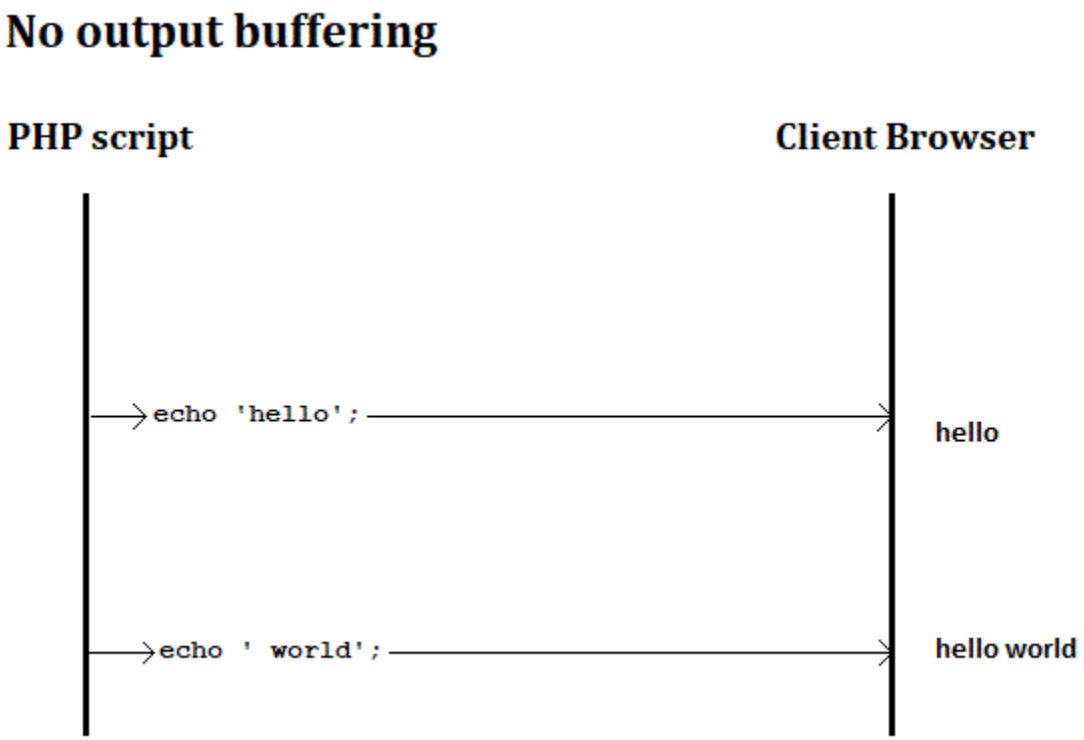
- Header-ele HTTP se trimit inaintea oricaror alte date (HTML)
  - Inceput fisier: **<?php** header("..."); ?><!DOCTYPE HTML PUBLIC ... <html>...<body>...</body></html>
  - Nici macar **un spatiu** nu trebuie sa apara inainte de primul **<?php**
  - Daca necesitatea de a trimite header-e poate aparea mai tarziu in script se foloseste obligatoriu Buffer iesire

Buffer iesire

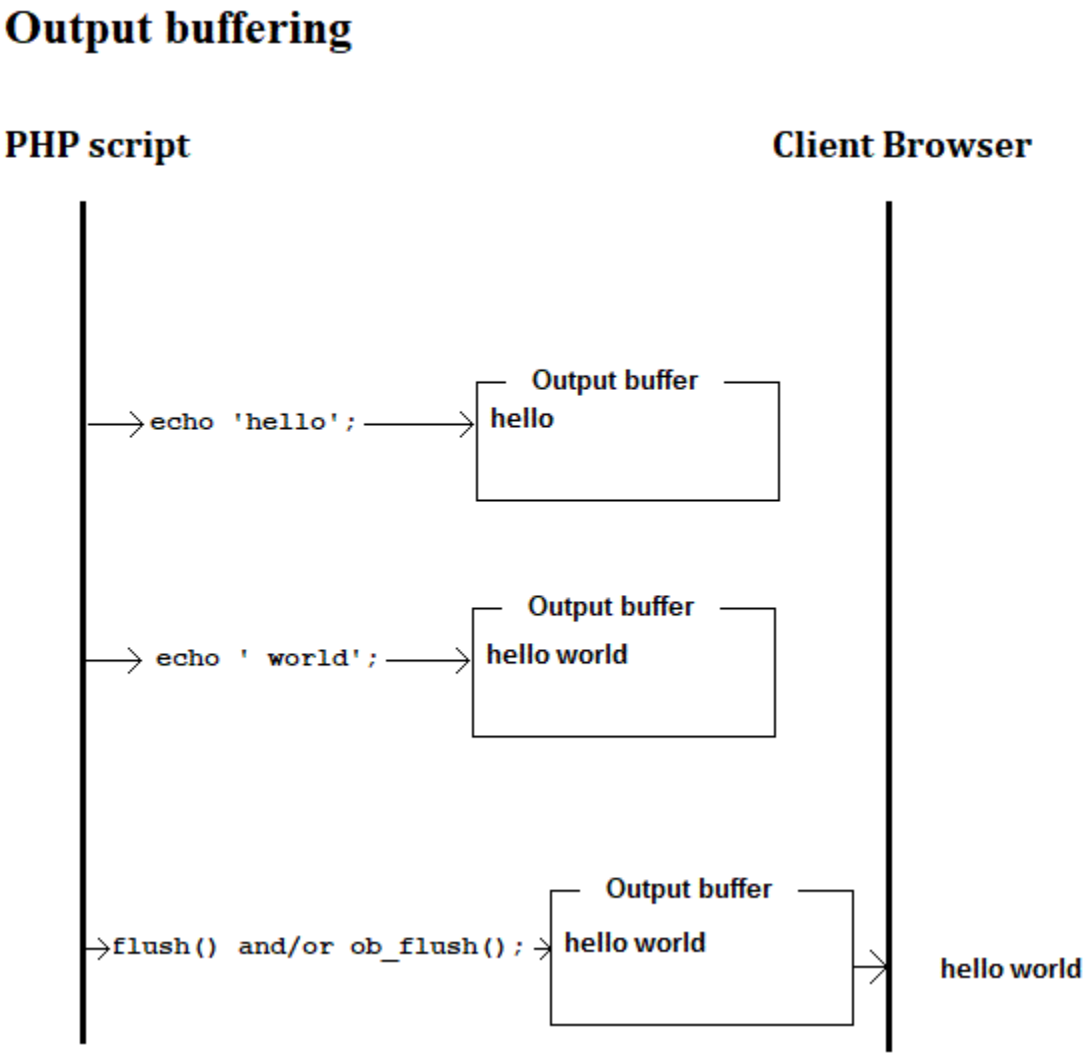
- Copie orice iesire a scriptului PHP intr-un buffer de memorie fara sa transmita nimic clientului
- Utilizat in general pentru conlucrarea cu header-e HTTP, evitarea generarii de HTML inainte de terminarea lucrului cu header-e
- ob_start();
- ob_end_flush ( );
- ob_end_clean ( );

- ob_get_contents ( )

Buffer iesire



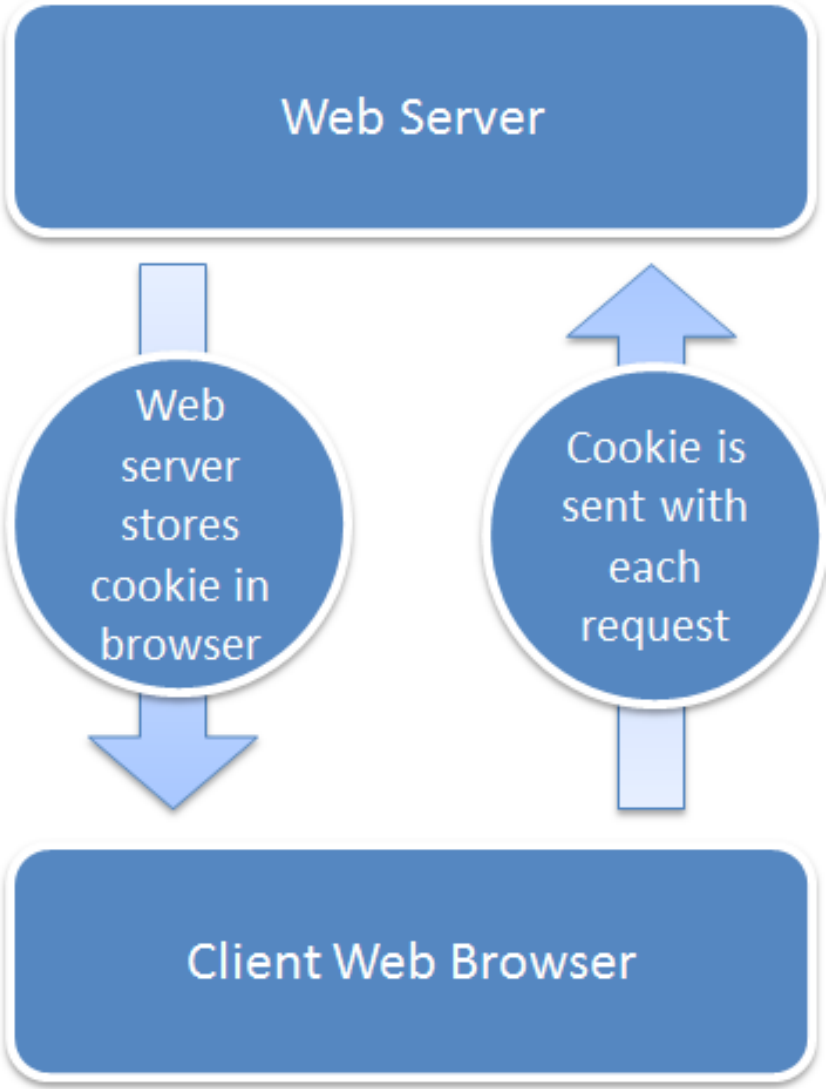
Buffer iesire



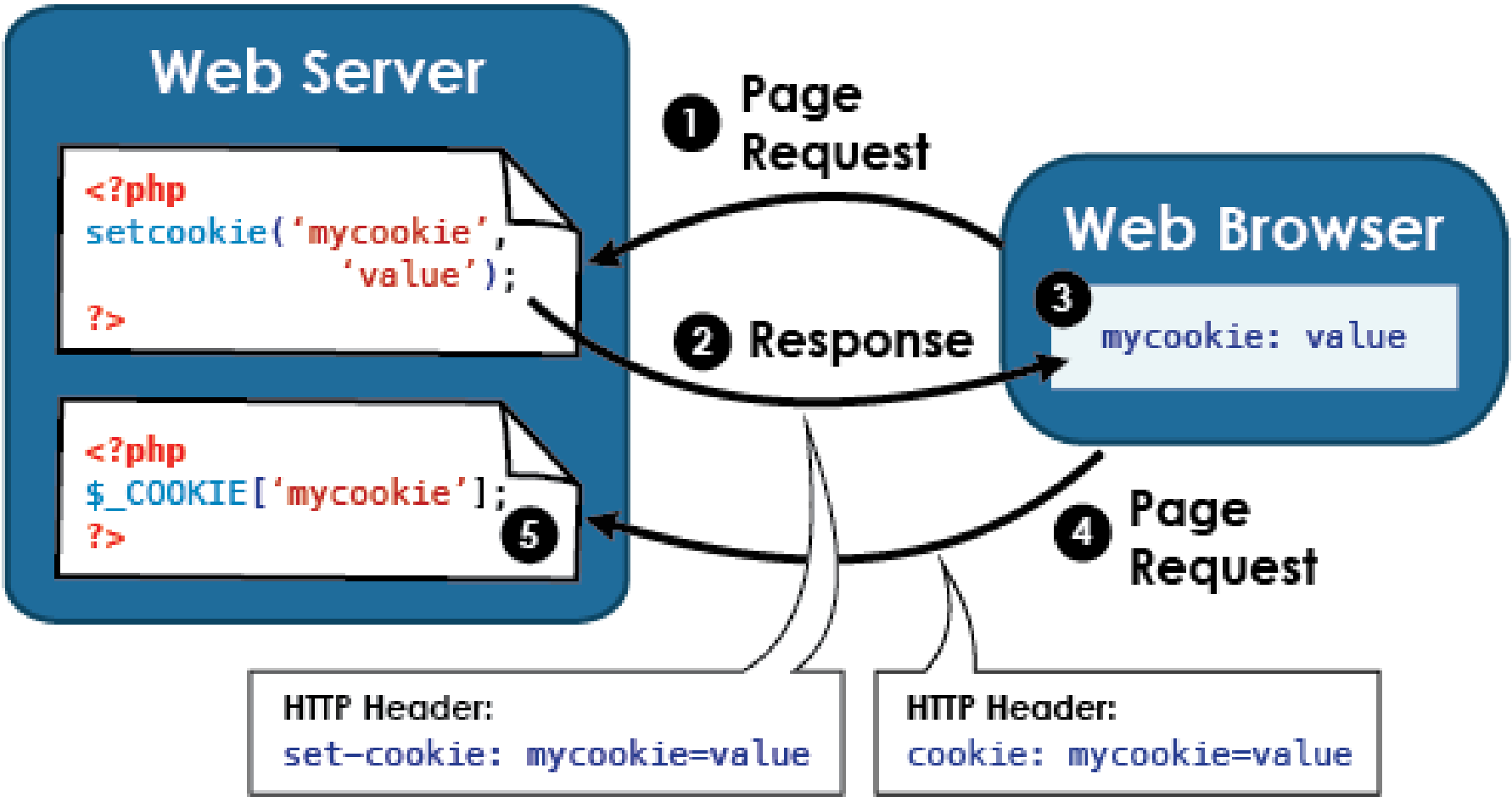
Cookies

- mici cantitati de date ce se stocheaza pe masina client (de obicei gestionat de browser)
- Circula impreuna cu (**este**) header HTTP
- setcookie (string name , string value , int expire , string path , string domain , bool secure , bool httponly)
  - nume (ptr. identificare)
  - value (valoarea/datele stocate)

Cookies


Web Server
Web server stores cookie in browser
Cookie is sent with each request
Client Web Browser

Cookies


Web Server
<?php
setcookie('mycookie',
'value');
?>
<?php
$_COOKIE['mycookie'];
?>
5
1 Page Request
2 Response
Web Browser
3
mycookie: value
4 Page Request
HTTP Header:
set-cookie: mycookie=value
HTTP Header:
cookie: mycookie=value

Cookies

- setcookie(string $name, string $value , int $expire = 0)
  - expire: UNIX time stamp, nr. sec. din 1970
  - time()+nr. sec. de viata dorite
- datele se stocheaza pe client: probleme de securitate
- Se poate obtine valoarea memorata prin variabila globala $_COOKIE['nume']
  - **NU** in acelasi script
  - daca un script php trimite un cookie cu header-ele, de-abia **urmatorul** script accesat va primi acele cookie in header-e

Cookies

```php
<?php
$value = 'something from somewhere';

setcookie("TestCookie", $value);
setcookie("TestCookie", $value, time()+3600);  /* expire in 1
hour */
setcookie("TestCookie", $value, time()+3600, "/~rasmus/",
"example.com", 1);
?>
```

```php
<?php
//Doar pe urmatoarele pagini !!!!

// Print an individual cookie
echo $_COOKIE["TestCookie"];

// Another way to debug/test is to view all cookies
print_r($_COOKIE);
?>
```

Cookies

```php
<?php
//Cookie arrays
// set the cookies
setcookie("cookie[three]", "cookiethree");
setcookie("cookie[two]", "cookietwo");
setcookie("cookie[one]", "cookieone");

// after the page reloads, print them out
if (isset($_COOKIE['cookie']))
    {
    foreach ($_COOKIE['cookie'] as $name => $value)
        {
        $name = htmlspecialchars($name);
        $value = htmlspecialchars($value);
        echo "$name : $value <br />\n";
        }
    }
?>
```

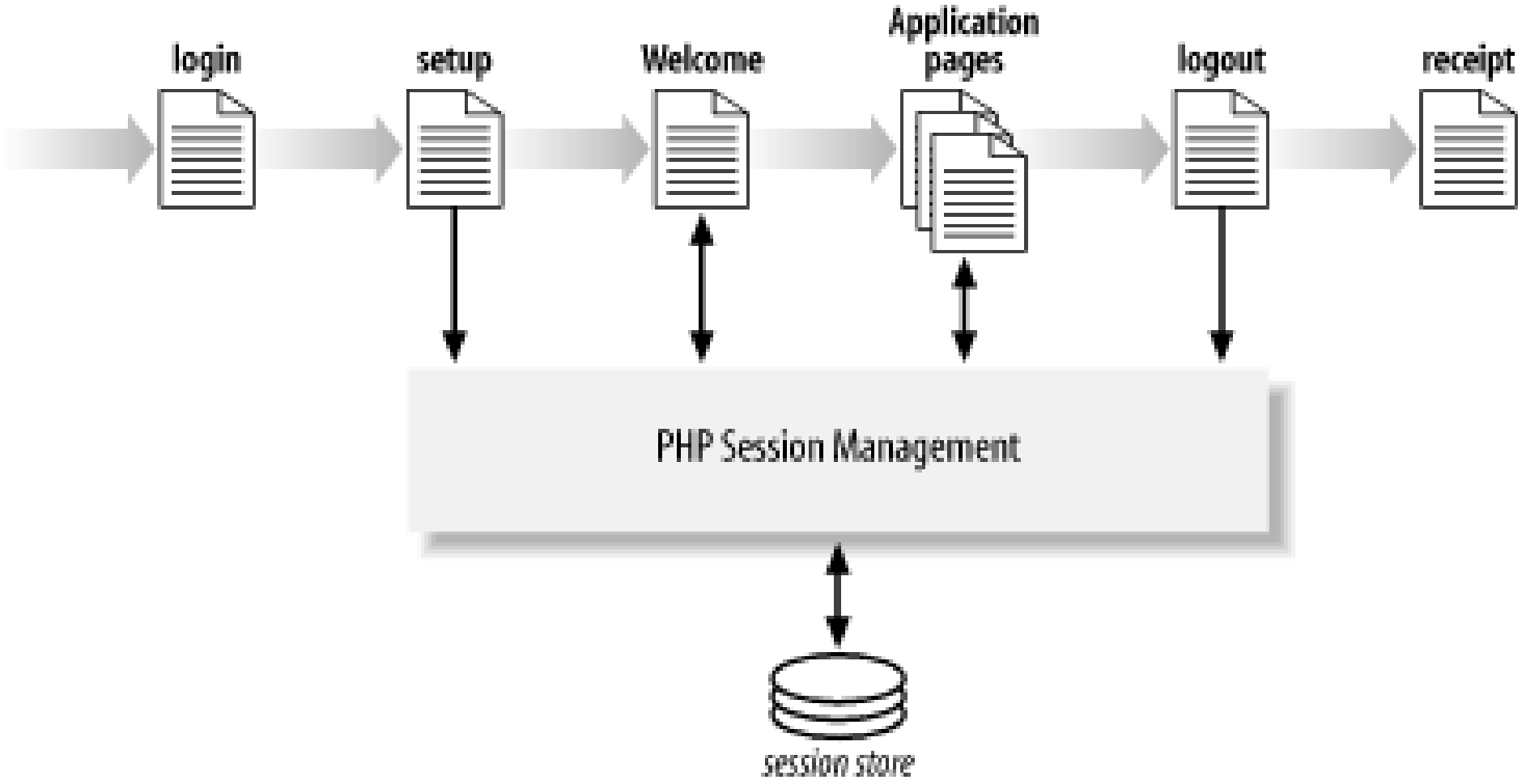
Sesiune

- cookie poate oferi "memorie" aplicatiilor web
- dezavantaje
  - datele se stocheaza la client, nu sunt in siguranta
  - nu se pot stoca oricate date (max. 20)
  - e posibil clientul sa nu accepte cookie
- Sesiunea pentru evitarea acestor dezavantaje
  - stocare pe server
  - oricat de mult date
  - daca clientul nu accepta cookie, "memoria" se realizeaza prin metoda "get"

Sesiune

- session_start(); (session_ID din GET, POST, COOKIE)
- session_write_close ( );
- session_id ([string id]);
- datele se manipuleaza prin variabila globala $_SESSION care ofera acces la citirea/scrierea datelor

Sesiune


login
setup
Welcome
Application pages
logout
receipt
PHP Session Management
session store

Sesiune


SESSIONS Management
rone
plus2net
SERVER
Maintains sessions for all currently connected users.

Sesiune

```php
<?php
// Initialize the session.
// If you are using session_name("something"), don't forget it now!
session_start();

// Unset all of the session variables.
$_SESSION = array();

// If it's desired to kill the session, also delete the session cookie.
// Note: This will destroy the session, and not just the session data!
if (isset($_COOKIE[session_name()]))
    {
    setcookie(session_name(), '', time()-42000, '/');
    }
// Finally, destroy the session.
session_destroy();?>
```

Sesiune

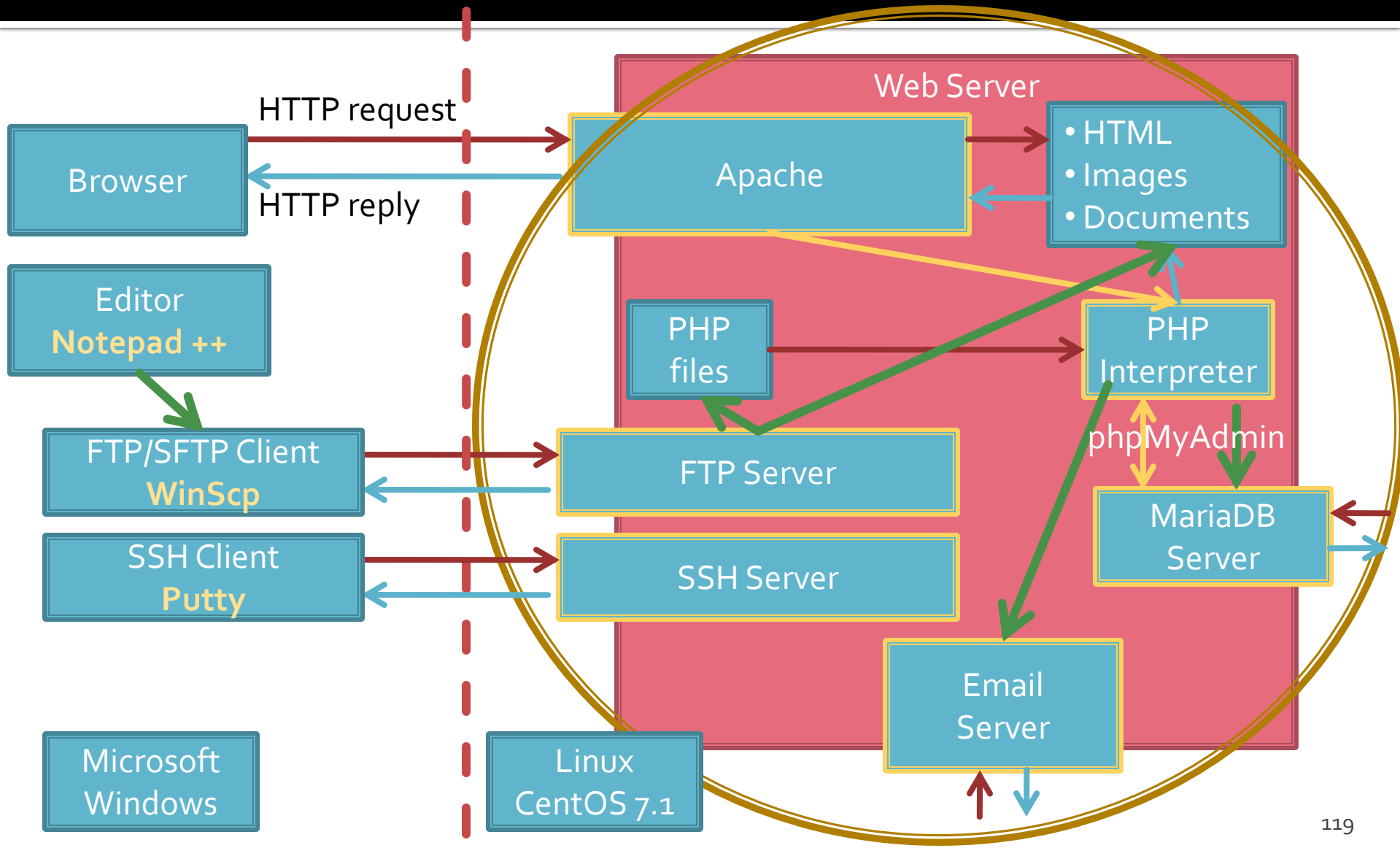
```php
<?php
// page1.php

session_start();

echo 'Welcome to page #1';

$_SESSION['favcolor'] = 'green';
$_SESSION['animal'] = 'cat';
$_SESSION['time'] = time();

// Works if session cookie was accepted
echo '<br /><a href="page2.php">page 2</a>';

// Or maybe pass along the session id, if needed
//echo '<br /><a href="page2.php?' . SID . '">page 2</a>';
echo '<a href="page2.php?' . session_name() . ' =' .
session_id() . ' ">page2</a>' ;
?>
```

Sesiune

```php
<?php
// page2.php

session_start();

echo 'Welcome to page #2<br />';

echo $_SESSION['favcolor']; // green
echo $_SESSION['animal'];   // cat
echo date('Y m d H:i:s', $_SESSION['time']);

// You may want to use SID here, like we did in page1.php
echo '<br /><a href="page1.php">page 1</a>';
?>
```

Reference server and applications

Using LAMP


Web Server
HTTP request
HTTP reply
Browser
Apache
• HTML
• Images
• Documents
Editor
Notepad ++
PHP
files
PHP
Interpreter
phpMyAdmin
FTP/SFTP Client
WinScp
FTP Server
MariaDB
Server
SSH Client
Putty
SSH Server
Email
Server
Microsoft
Windows
Linux
CentOS 7.1

Using LAMP

- differences from a Windows computer
  - system commands difficult
    - command prompt, SSH, Putty
  - files submitted by FTP
    - Copy/Paste unavailable
  - administration of MySql server:
    - using PhpMyAdmin (preloaded)
    - using other software on the host computer (eg. MySQL Workbench)

Using LAMP – Advantages

- Advantages
  - Available applications have newer versions (**2020 ~**)
    - CentOS/7.1, Apache/2.4.6, PHP/5.4.16, MariaDB/5.5.44, PhpMyAdmin/4.4.15
    - Ubuntu/20.04, Apache/ 2.4.41, PHP/ 7.4.3, MariaDB/ 10.3.31, PhpMyAdmin/4.9.5
    - Debian/12.5, Apache/ 2.4.57, PHP/ 8.2.7, MariaDB/ 10.11.6, PhpMyAdmin/5.2.1
  - Available applications very similar with real life hosting solutions
    - SSH
    - FTP
    - Email
      - for full functionality the network interface for the VM must be changed **Host-only** -> **Bridged**

Reference Server

- rf-opto.etti.tuiasi.ro > Master > Web Design

Project/Design

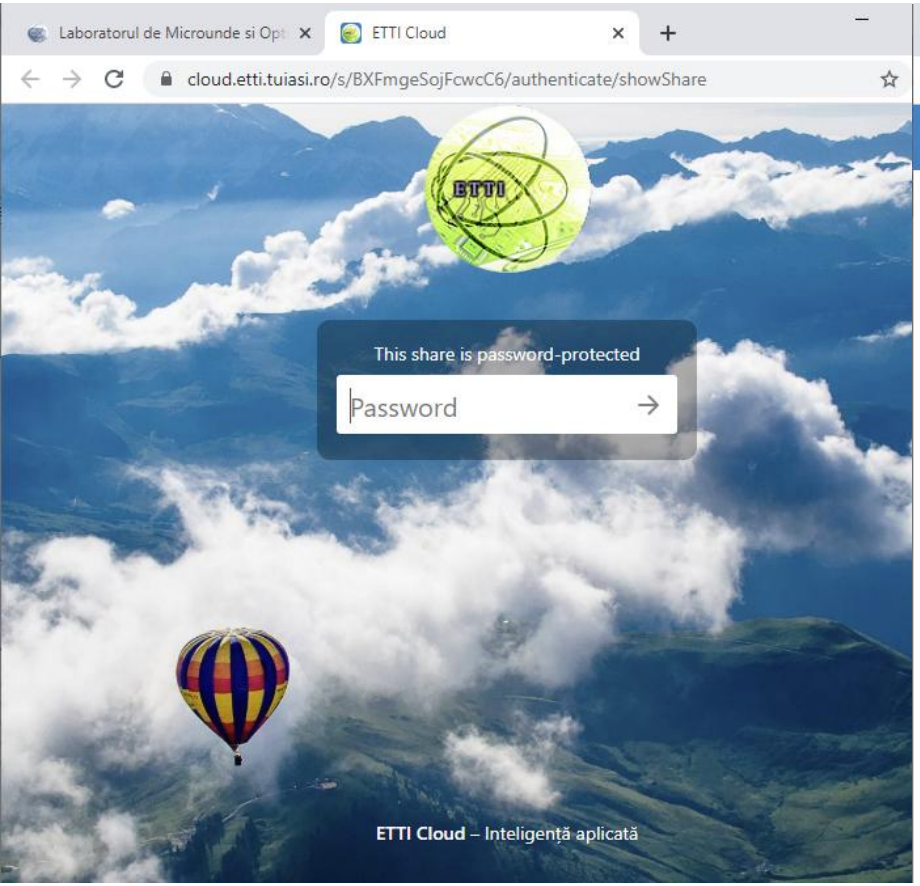
VMware Workstation Player (link, 0 Bytes, en, GB)
Ubuntu VM for VMWare (link, 0 Bytes, en, GB)
Ubuntu Setup (pdf, 1.83 MB, en, GB)
Centos VM for VMWare (link, 0 Bytes, en, GB)
Centos Setup (pdf, 2.54 MB, en, GB)

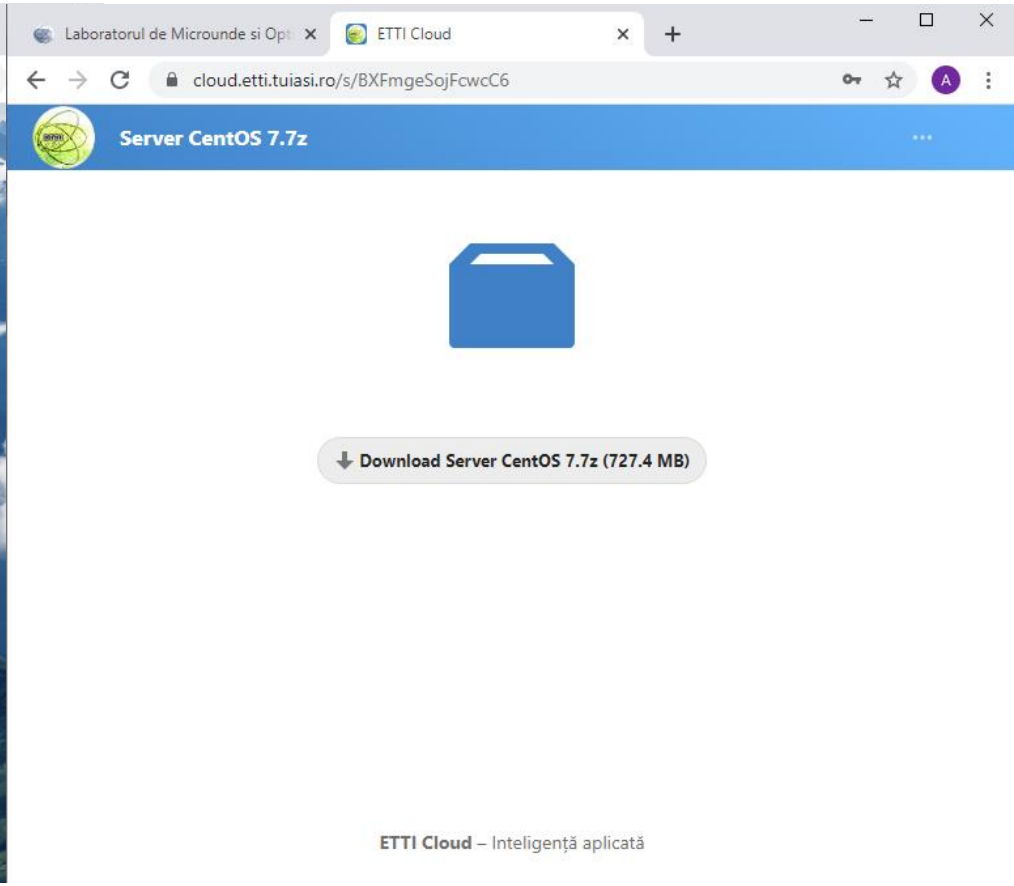
Examen

Online Exam manual (pdf, 2.56 MB, en, GB)
Manual examen on-line (pdf, 2.65 MB, ro, RO)

Reference Server

- Cloud ETTI: **RF-opto3#**



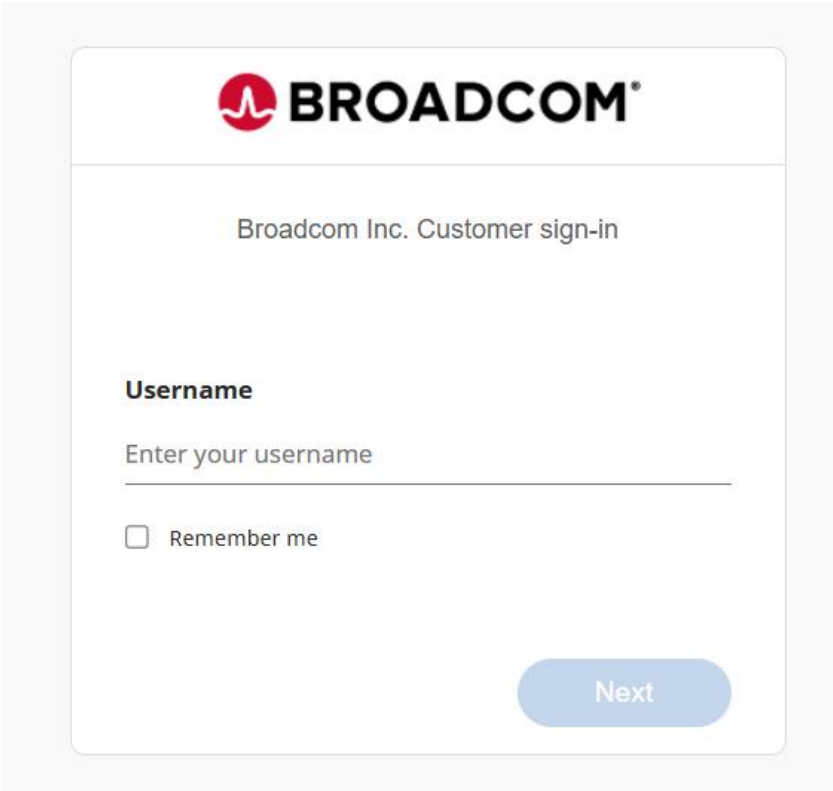


Server referinta

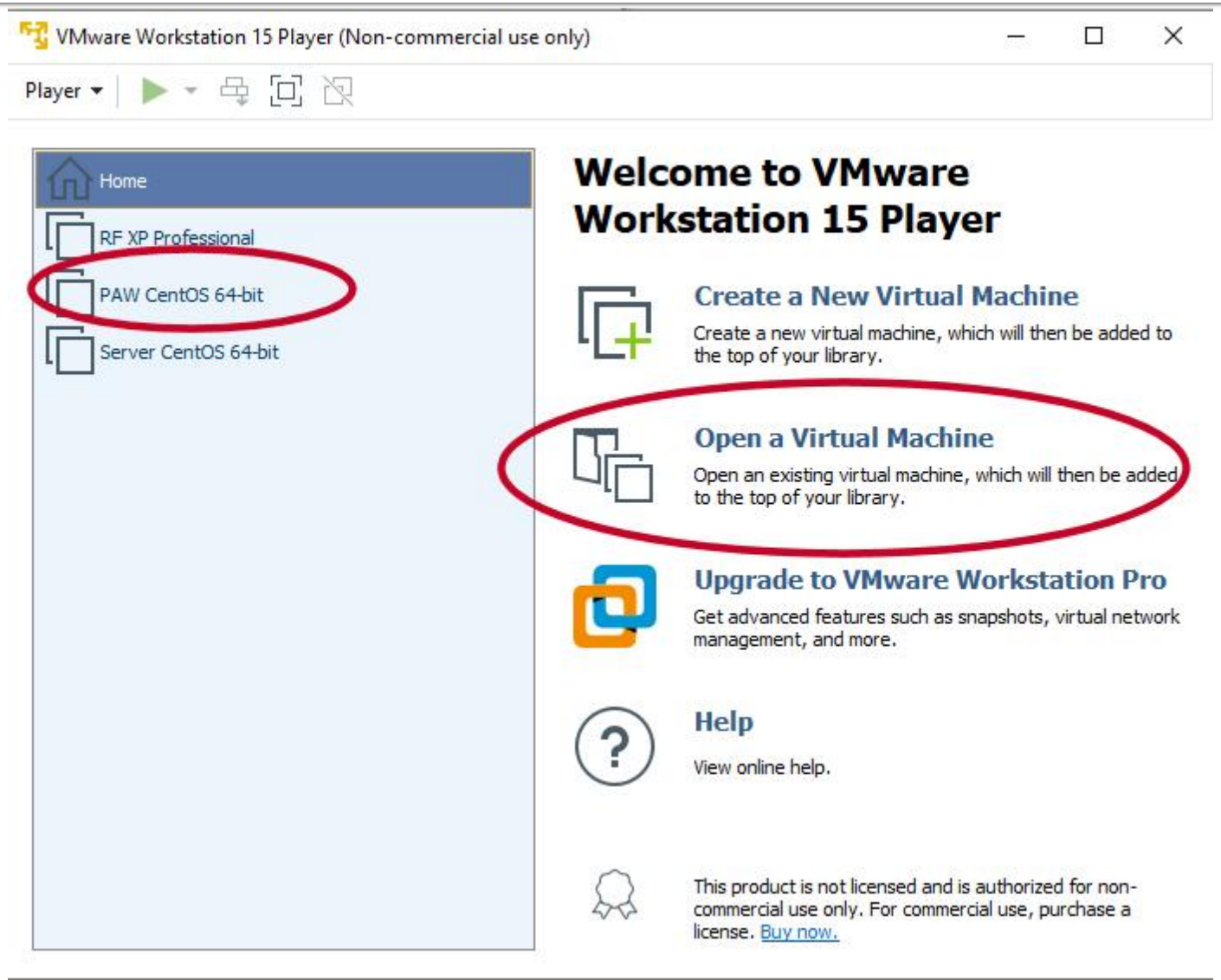
- Masina virtuala
- VMware Workstation Player
  - Gratuit (non-comercial)
  - https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html
- Inlocuit de VMware Workstation Pro

Server referinta

- Masina virtuala
- VMware Workstation Pro (Broadcom)



Reference Server

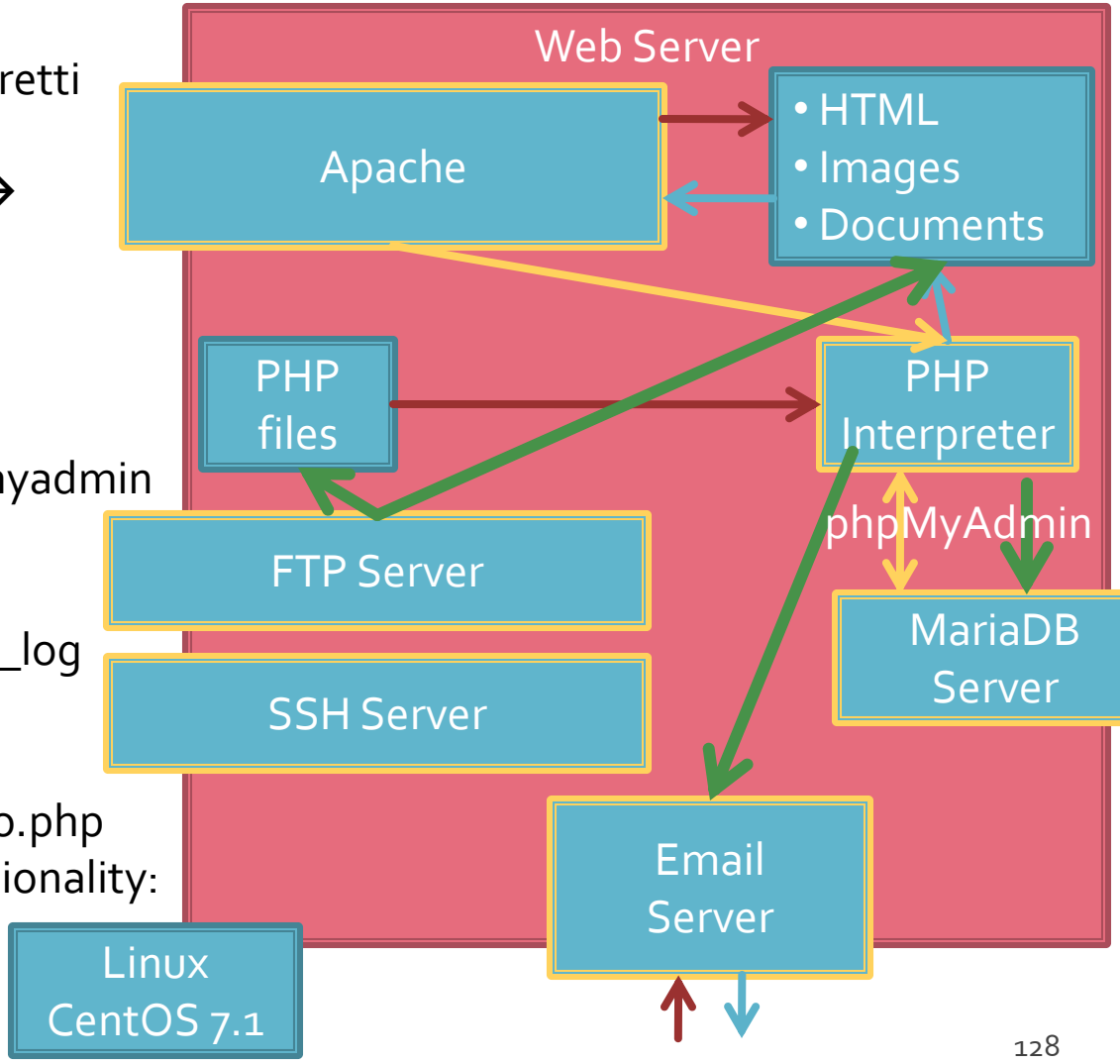


Possible problems

- Current VMWare Player runs **only** on **64bit** operating systems Windows/Linux
  - for 32bit operating systems previous (**certified originals**) can be made available on rf-opto
- The host computer **must** enable **Hardware Virtualization**
  - Hardware Virtualization is enabled in BIOS, depending on the PC manufacturer: Processor, Chipset, Northbridge
  - Options name: VT-x, AMD-V, Vanderpool, Hyper-V, SVM, Intel Virtualization Technology. if available: Intel VT-d, AMD IOMMU
- VM archive requires **7zip** native to the target operating system

Using LAMP

1. login → root:masterrc / paw:masteretti
2. ifconfig → 192.168.30.5
3. putty.exe → 192.168.30.5 → SSH → root:masterrc (remote login)
4. [other linux command line]
5. FTP → Winscp → SFTP → student:masterrc@192.168.30.5
6. MySql → http://192.168.30.5/phpmyadmin → root:masterrc / root:masteretti
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. if DHCP service stops Apache functionality: service httpd restart



LAMP Reference Server 2023

- Linux, two variants
  - Centos 7.1
    - PHP 5.4.16
    - MariaDB 5.5.44
    - Apache 2.4.6
    - **root**/student:masterrc
  - Ubuntu 20.04 (**recommended**)
    - PHP 7.4.3
    - MariaDB 10.3.31
    - Apache 2.4.41
    - **paw**/student:masteretti
    - correction **paw FTP access**:
      - sudo usermod -a -G upload paw
      - sudo chmod -R 775 /var/www

LAMP Reference Server 2025

- Linux, three proposed servers
  - CentOS 7.1
  - Ubuntu 20.04
  - **Debian 12.5**

LAMP Reference Server

- Centos 7.1
  - PHP 5.4.16
  - MariaDB 5.5.44 / root:masterrc
  - Apache 2.4.6
  - PhpMyAdmin/4.4.15
  - **root**/student:masterrc
  - Python 2.7.5
  - creat: Workstation Player 12.x (**12**)

LAMP Reference Server

- Ubuntu 20.04
  - PHP 7.4.3
  - MariaDB 10.3.31 / root:masteretti
  - Apache 2.4.41
  - **paw**/student:masteretti
  - necesar suplimentar pentru **acces FTP user paw**:
    - sudo usermod -a -G upload paw
    - sudo chmod -R 775 /var/www
  - Python 3.8.10
  - creat: Workstation Player 15.x (**16**)

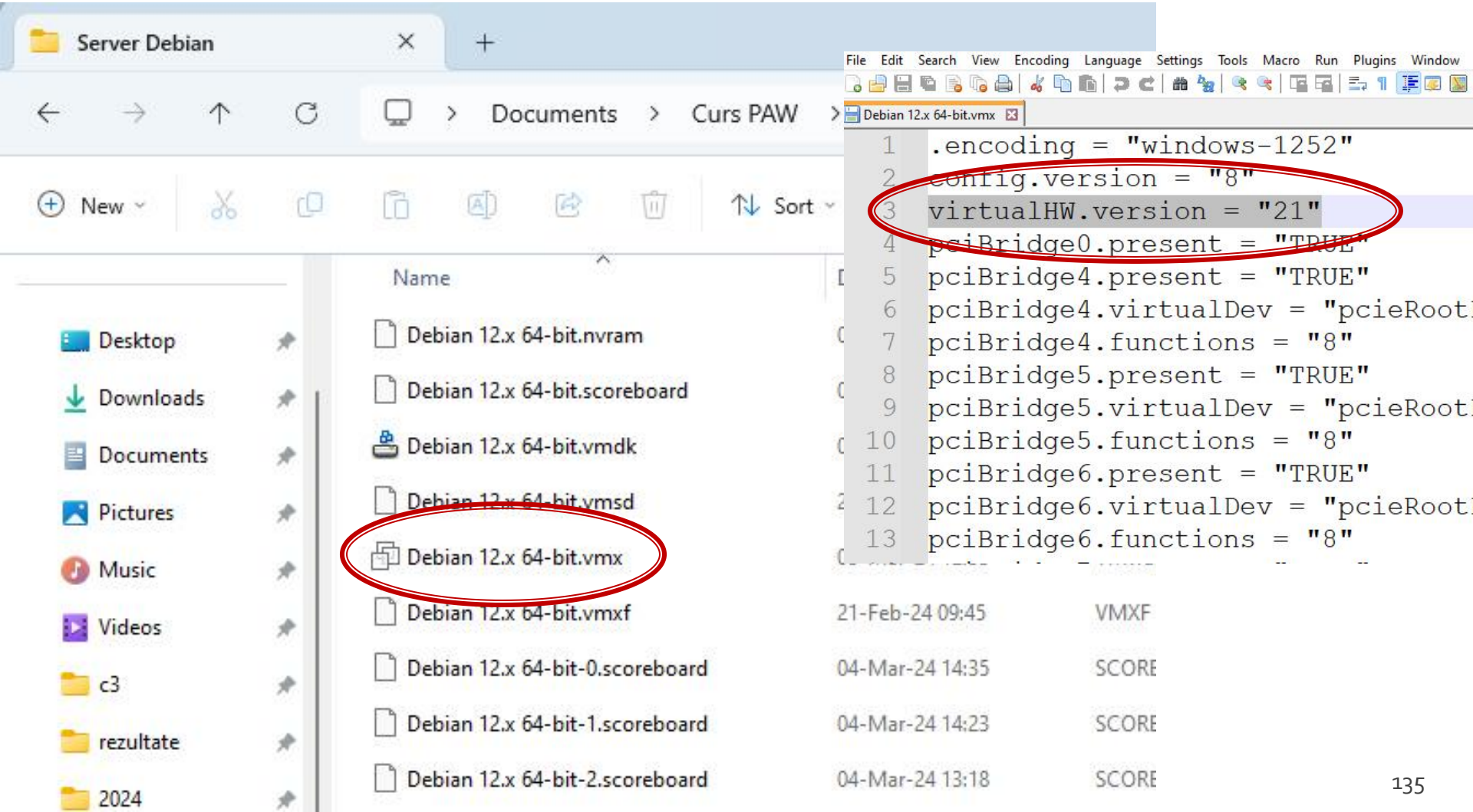
LAMP Reference Server

- Debian 12.5
  - PHP 8.2.7
  - MariaDB 10.11.6 / root:masteretti
  - Apache 2.4.57
  - PhpMyAdmin/5.2.1 deb
  - **root**/paw/student:masteretti
  - Python 3.11.2
  - creat: Workstation Player 17.5 (**21**)

Server referinta

- Pentru rularea unui server pe o versiune VMware Player anterioara:
  - se localizeaza fisierul "*.vmx" a server-ului
  - se modifica virtualHW.version = "**21**" la o valoare mai mica (anterioara)
    - in 2.13 -> **18**

LAMP Reference Server



Support applications

- WinSCP (client FTP, gratuit)
  - https://winscp.net/eng/download.php
- Notepad ++ (editor, avansat, gratuit)
  - https://notepad-plus-plus.org/downloads/
- Putty (remote access)
  - https://www.putty.org/
- MySQL Workbench (gratuit, cont Oracle)
  - https://www.mysql.com/products/workbench/
- Visual Studio Code (gratuit, Microsoft)
  - https://code.visualstudio.com/download

Aplicatii suport

- Variante portabile

Laboratory

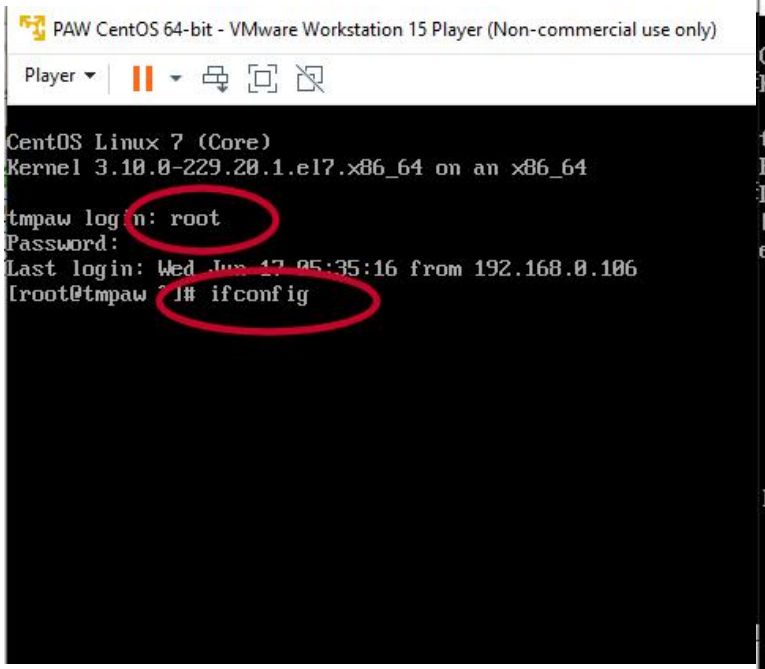
Laborator 1 (pdf, 1.48 MB, ro,)
Server Win2000 pentru VMWare Player - lab 1 (cloud) (link, 0 Bytes, en,)
Accesorii laborator (x32 - partial) (zip, 28.92 MB, en,)
Accesorii laborator (x64 - complet) (zip, 133.58 MB, en,)

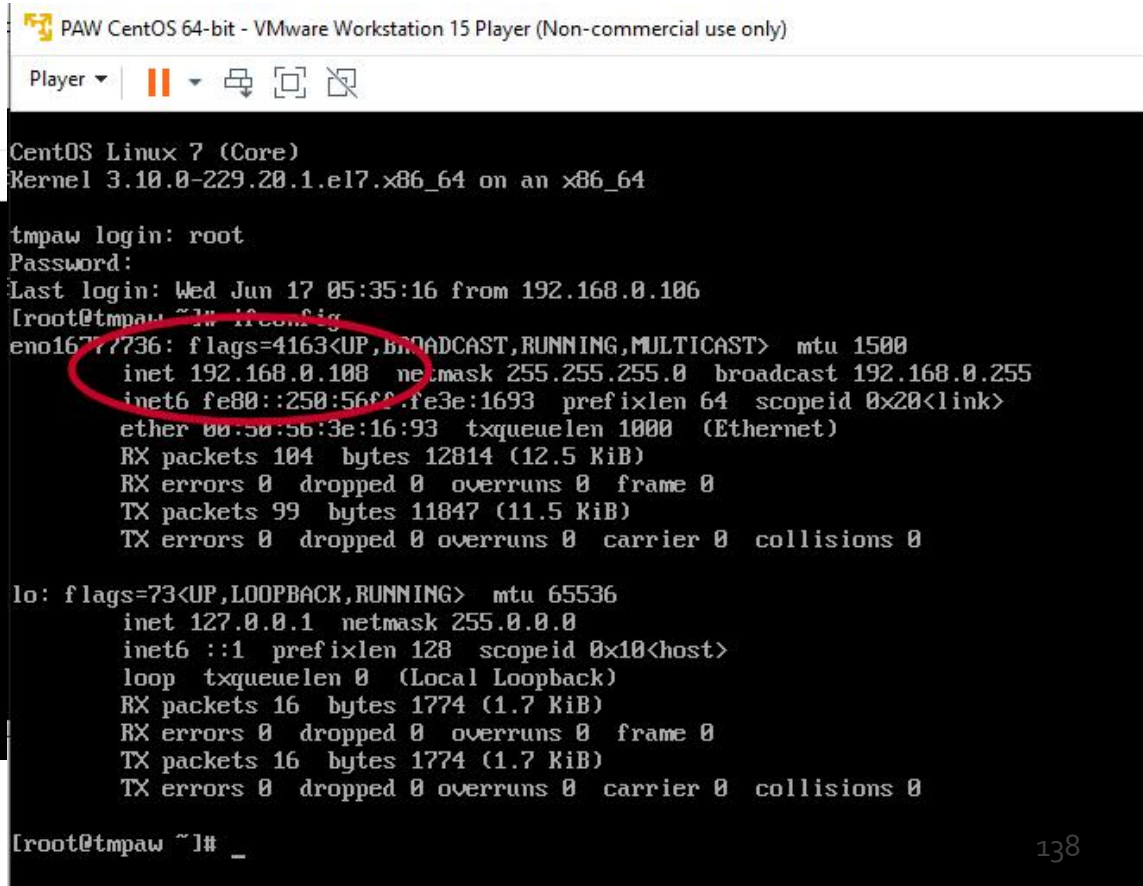
Project/Design

VMware Workstation Player (link, 0 Bytes, en,)
Server CentOS pentru VMWare Player (cloud) (link, 0 Bytes, en,)
Instalare Centos (pdf, 2.54 MB, en,)
Server Ubuntu pentru VMWare Player (cloud) (link, 0 Bytes, en,)
Instalare Ubuntu (pdf, 1.83 MB, en,)

IP address

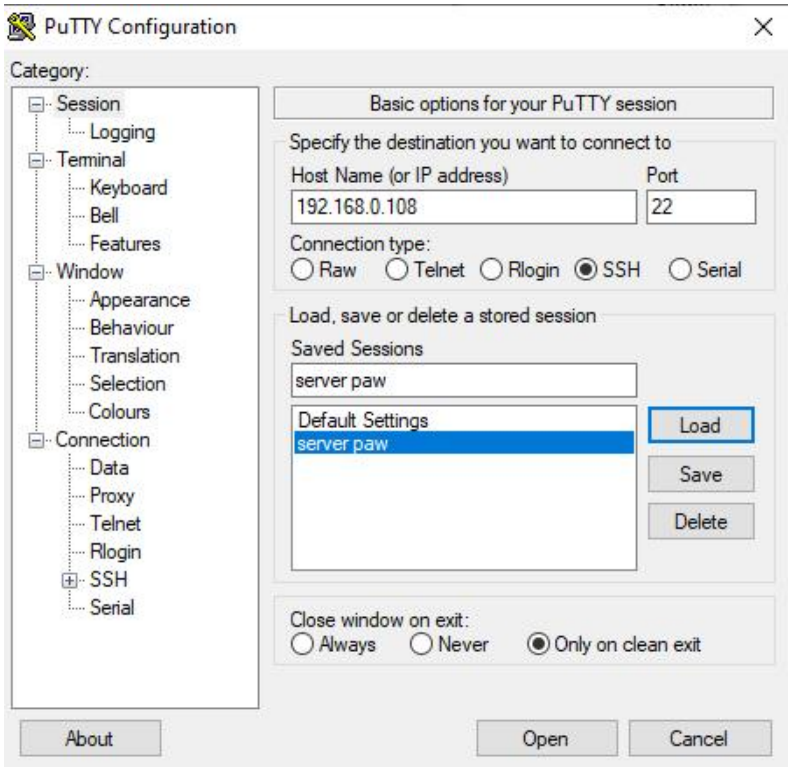
- login, ifconfig
- Ctrl + Alt + mouse

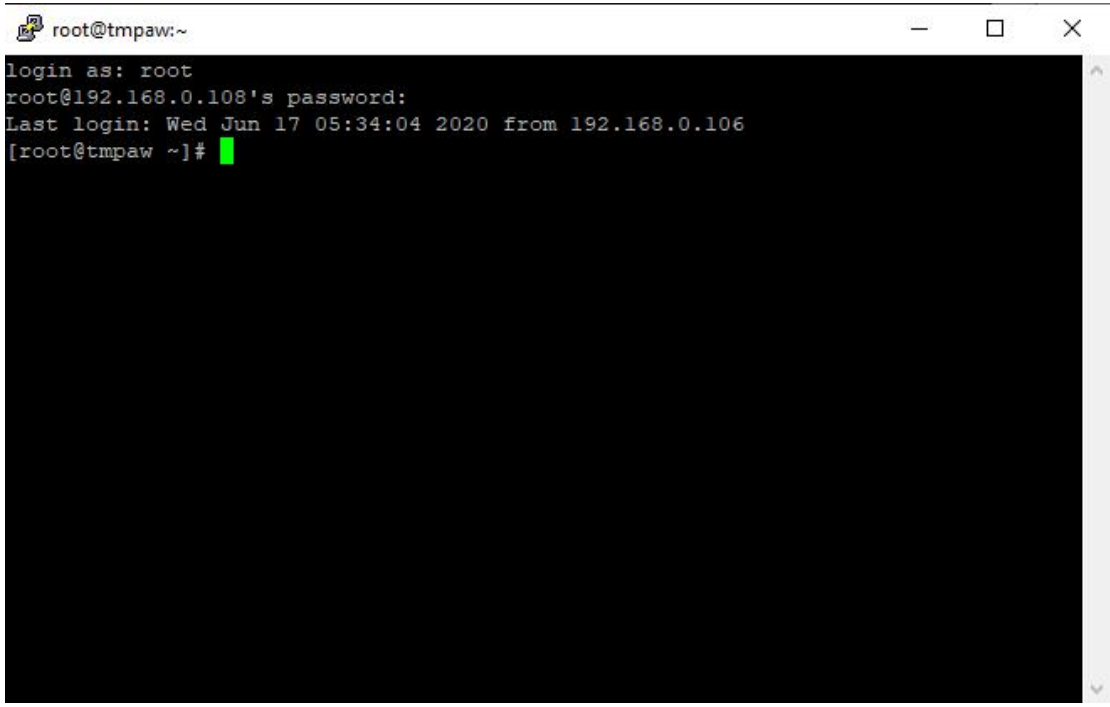




Putty

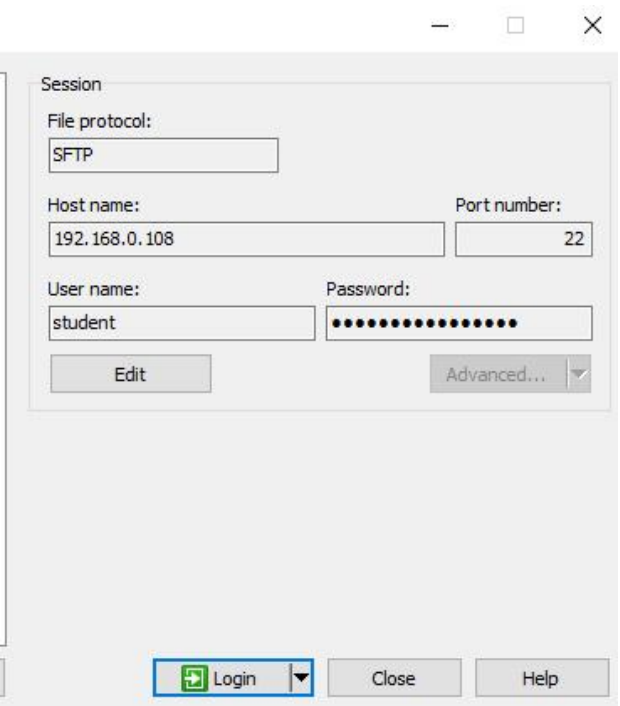
- putty.exe
- avoids mouse capture (CentOS), copy/paste etc.

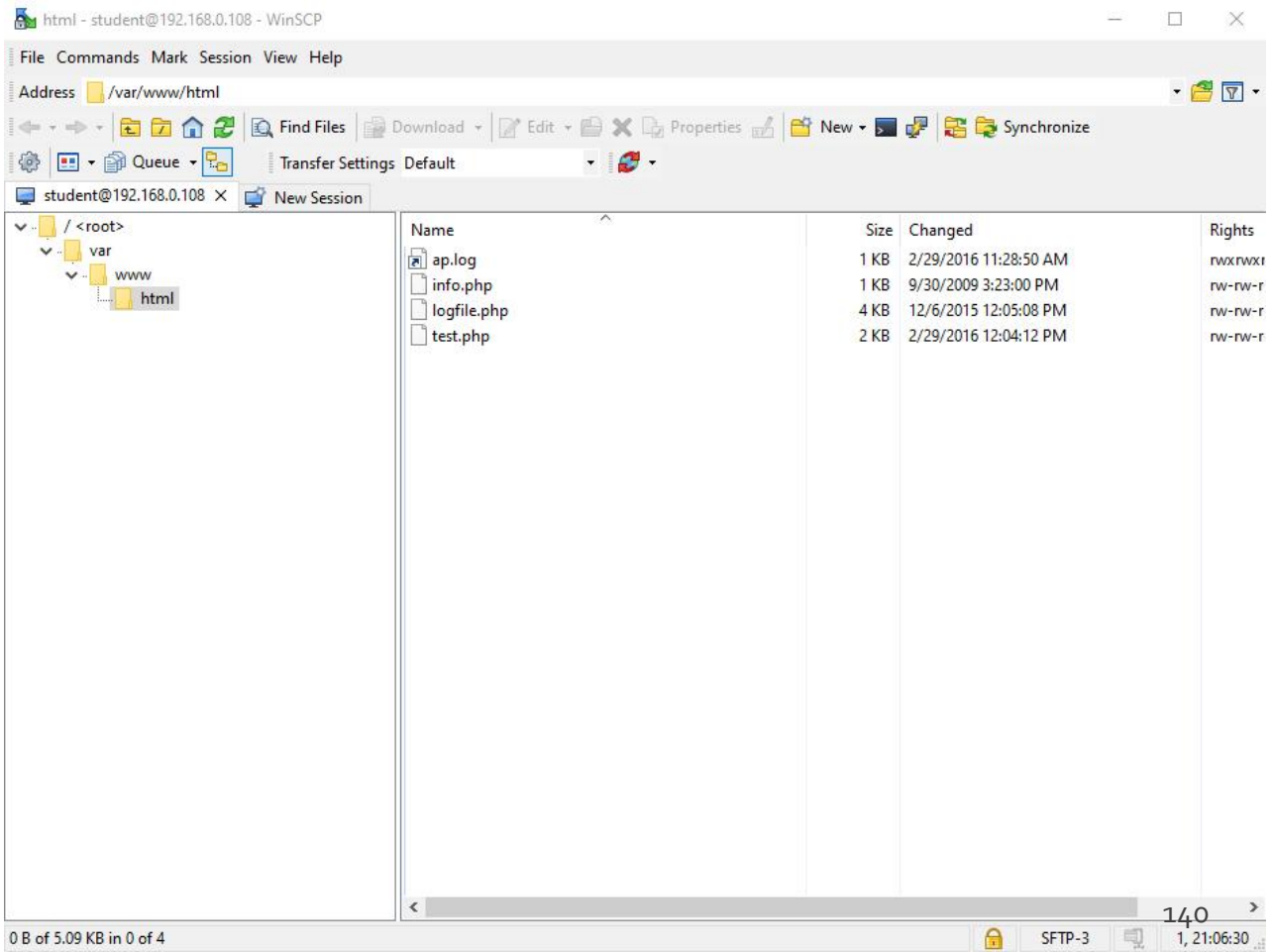




WinSCP

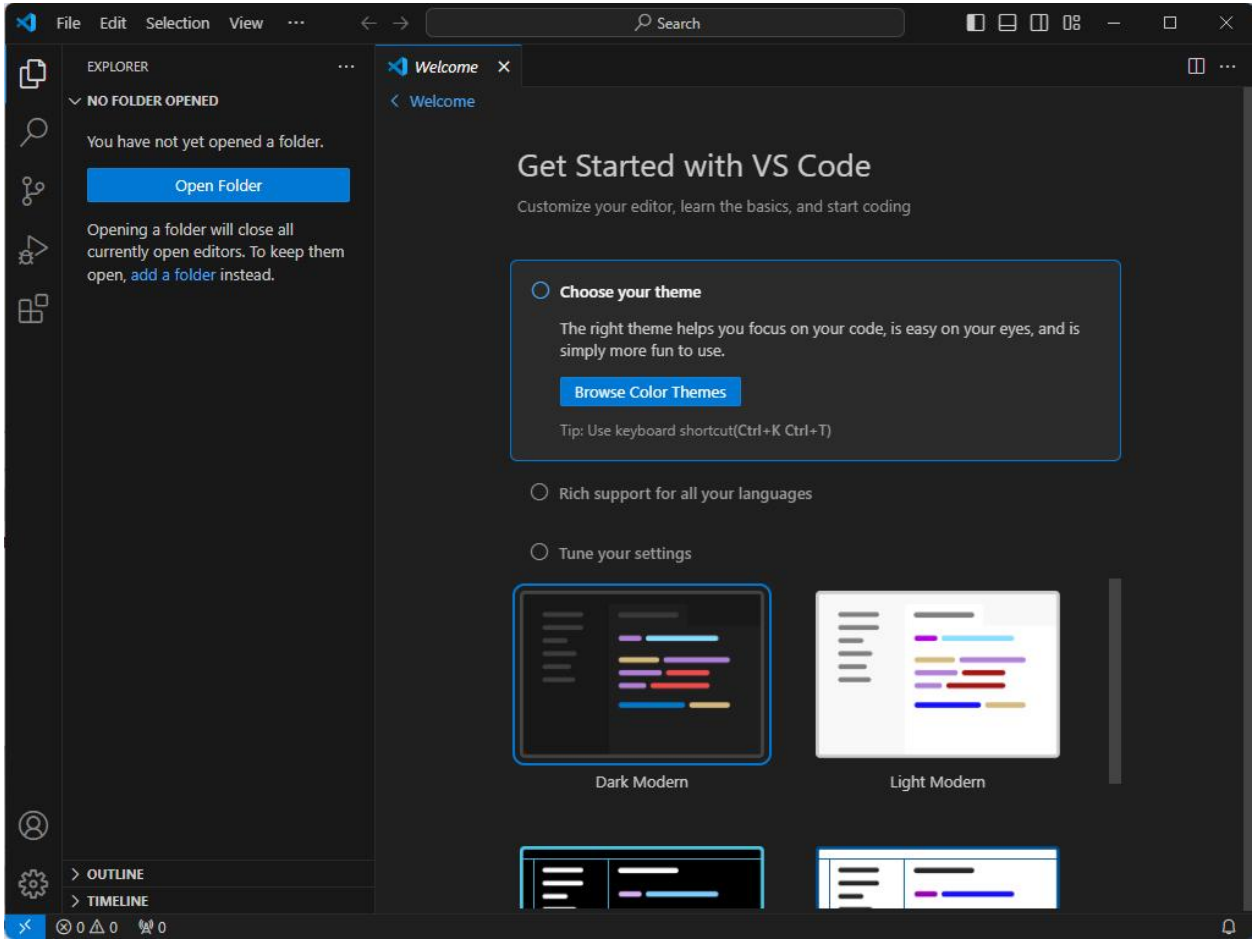
- FTP client
- upload files





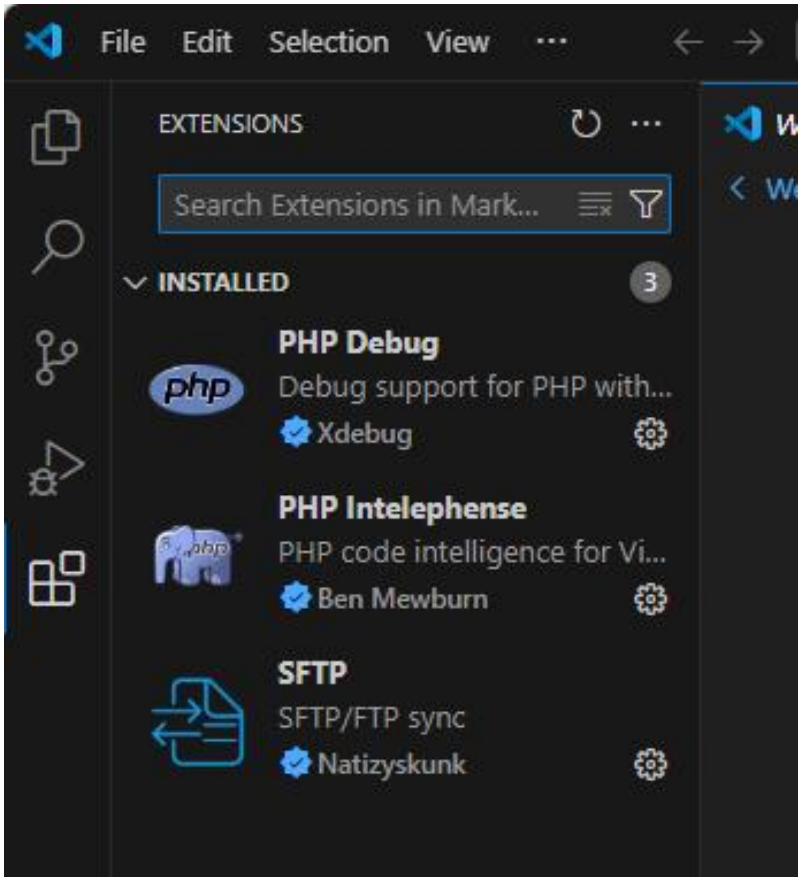
Visual Studio Code

- 1.87 Portable

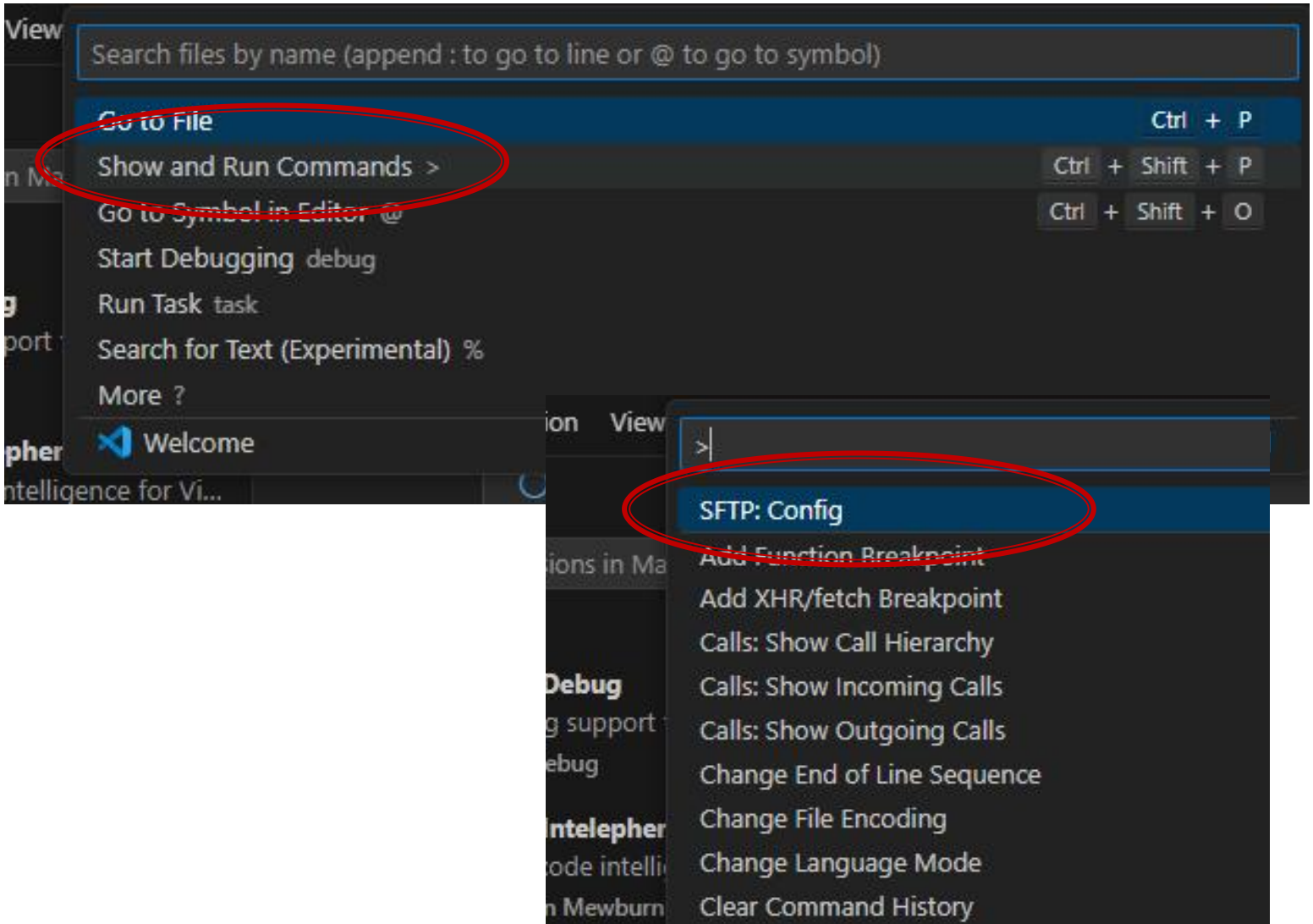


Visual Studio Code

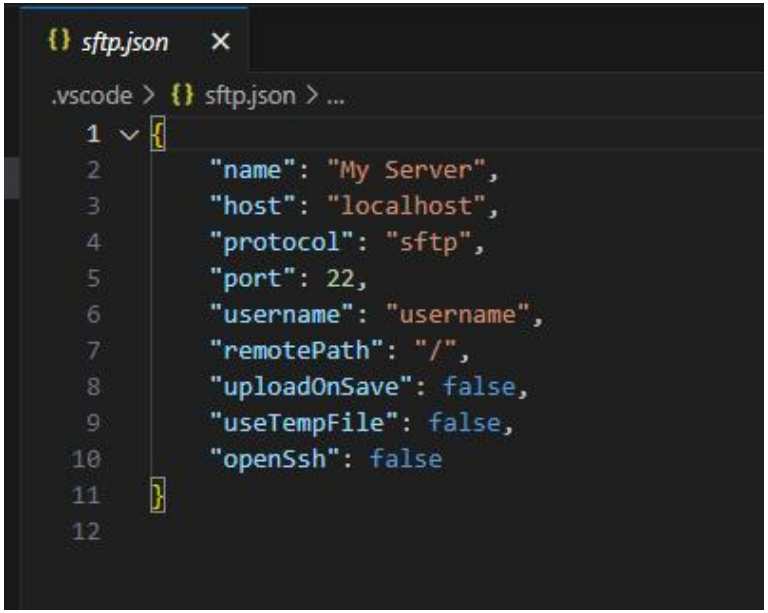
- Installed extensions
  - PHP Intelephense
    - PHP 8 -> Debian
  - PHP Debug (inactive for now)
  - SFTP – automatic save on server

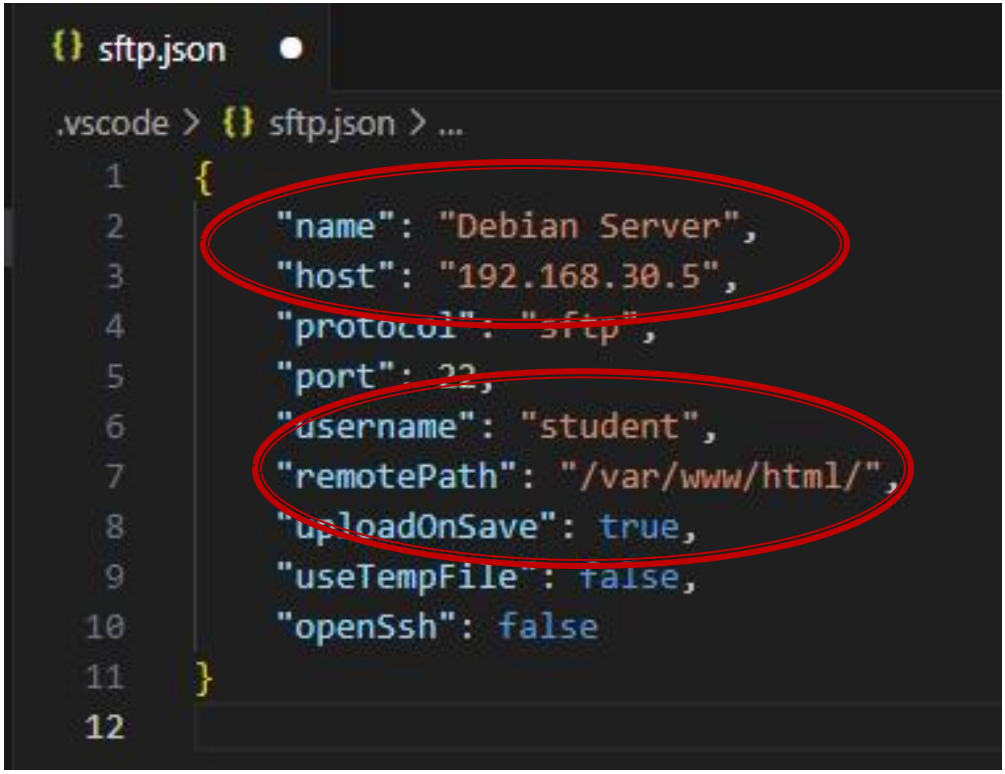


Visual Studio Code

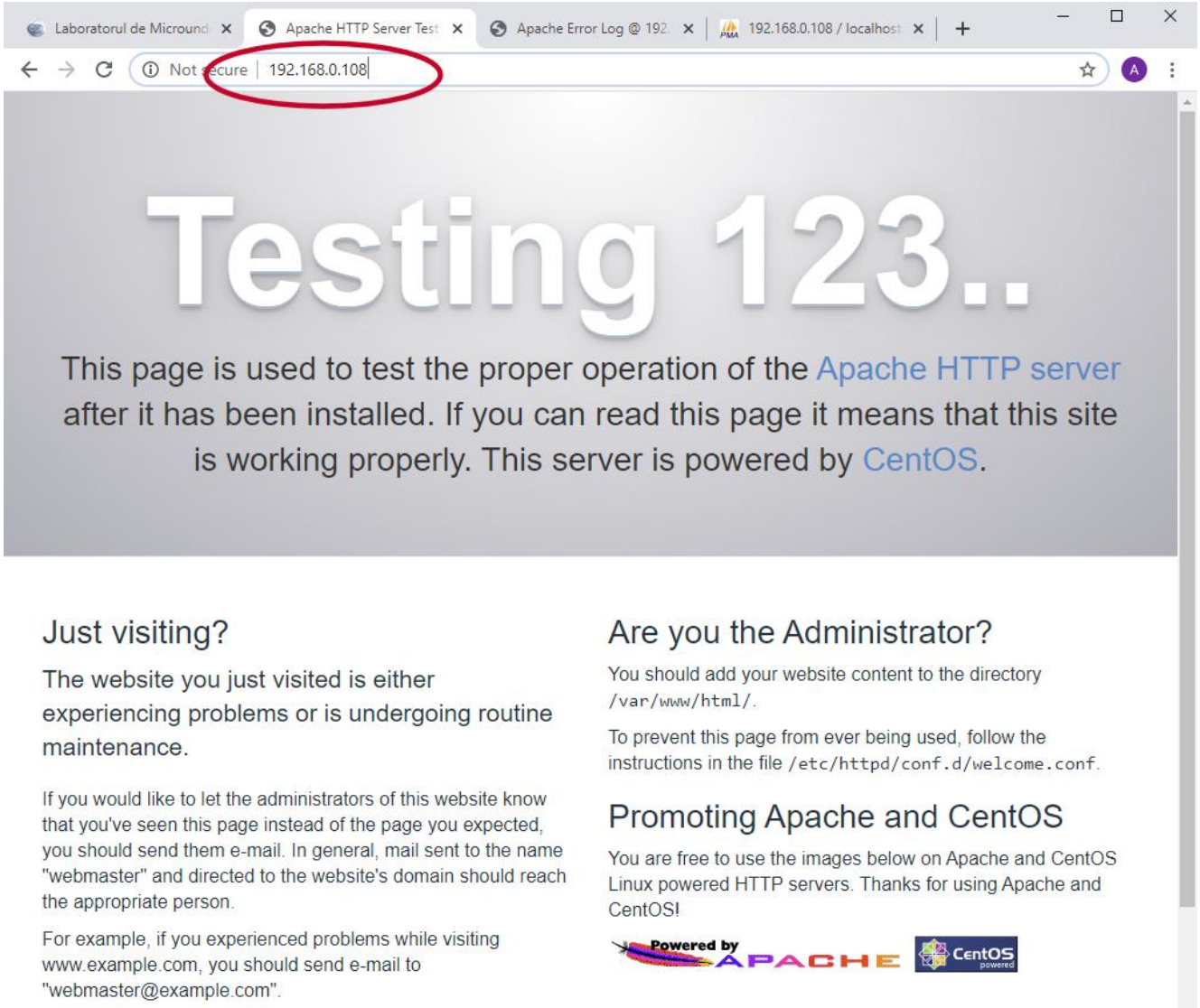


Visual Studio Code

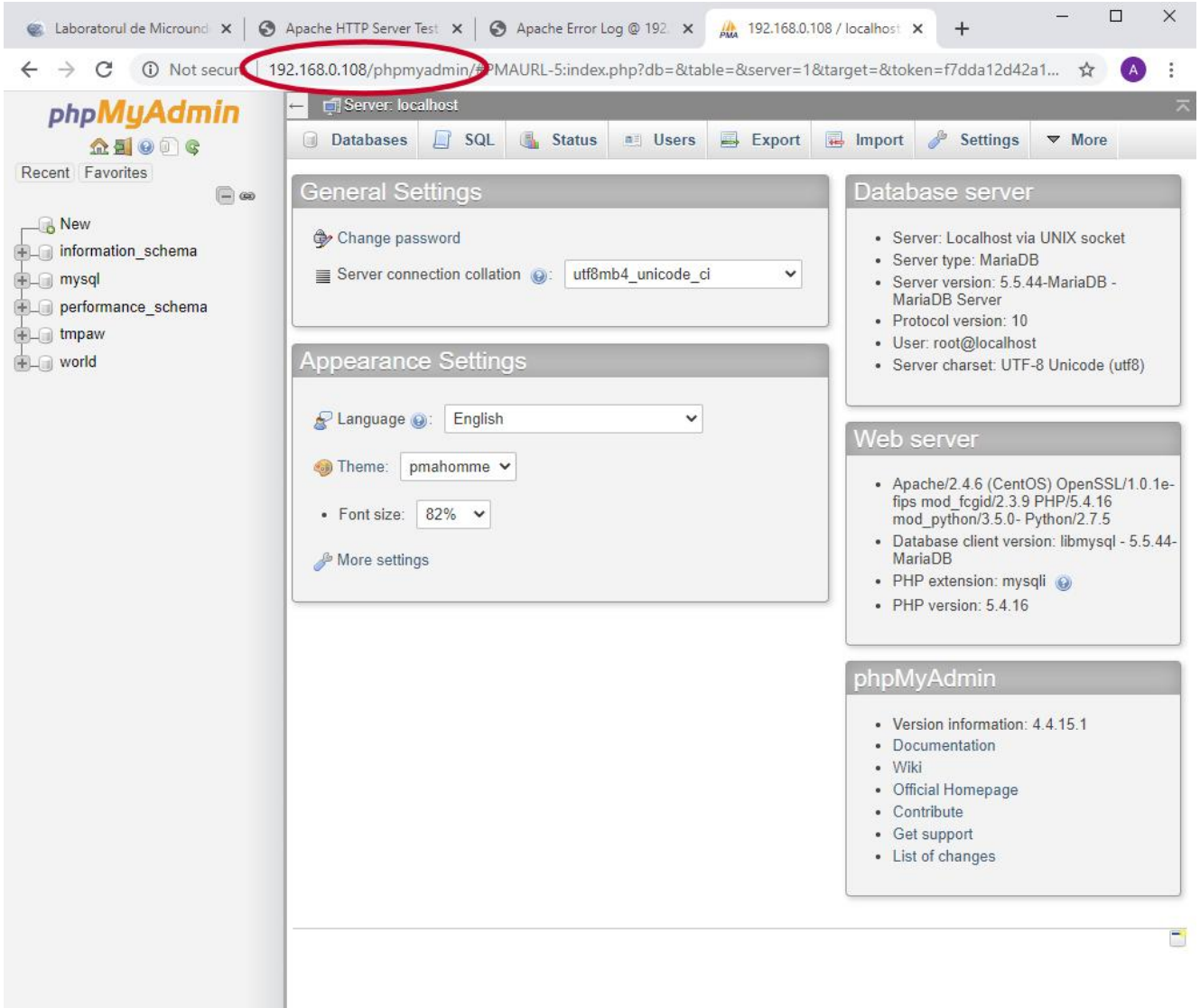




Browser


Laboratorul de Microunde
Apache HTTP Server Test
Apache Error Log @ 192.
192.168.0.108 / localhost
Not secure
192.168.0.108
Testing 123..
This page is used to test the proper operation of the Apache HTTP server after it has been installed. If you can read this page it means that this site is working properly. This server is powered by CentOS.
Just visiting?
The website you just visited is either experiencing problems or is undergoing routine maintenance.
If you would like to let the administrators of this website know that you've seen this page instead of the page you expected, you should send them e-mail. In general, mail sent to the name "webmaster" and directed to the website's domain should reach the appropriate person.
For example, if you experienced problems while visiting www.example.com, you should send e-mail to "webmaster@example.com".
Are you the Administrator?
You should add your website content to the directory /var/www/html/.
To prevent this page from ever being used, follow the instructions in the file /etc/httpd/conf.d/welcome.conf.
Promoting Apache and CentOS
You are free to use the images below on Apache and CentOS Linux powered HTTP servers. Thanks for using Apache and CentOS!
Powered by APACHE
CentOS powered

Server MySQL/MariaDB



MySql

SQL

MySql

- Baza de date – instrument pentru stocarea si manipularea informatiei eficient si efectiv
  - datele sunt protejate de corupere sau pierderi accidentale
  - nu se utilizeaza mai multe resurse decat minimul necesar
  - datele pot fi accesate cu performanta acceptabila
- Baze de date relationale
  - model relational (matematic eficient) – Codd ~1970

DBMS, RDBMS

- DBMS – database management system aplicatii incluse in baza de date pentru accesul la informatii
- RDBMS – Relational DBMS. Majoritatea sistemelor de baze de date tind la aceasta titulatura
  - ~300 de reguli trebuie respectate
  - nici un sistem actual nu implementeaza total aceste reguli

Relatii

- Toate sistemele de baze de date sunt caracterizate de:
  - toate informatiile sunt reprezentate intr-o aranjare ordonata **bidimensionala** numita **relatie**
  - toate valorile (atribute) stocate sunt scalare (in orice celula din tabel se stocheaza **o singura** valoare)
  - toate operatiile se aplica asupra unei intregi relatii si rezulta o intreaga relatie
- Terminologii (MySql)
  - tabel – table / recordset / result set
  - linie – record / row
  - coloana – field / column

Relatii, chei

- toate informatiile sunt reprezentate intr-o aranjare bidimensionala numita relatie
  - aranjarile bidimensionale nu sunt ordonate implicit
  - datele trebuie stocate pentru a implementa o relatie in asa fel incat fiecare linie sa fie unica
- cheie candidata
  - exista cel putin o combinatie de atribute (coloane) care pot identifica in mod unic o linie
  - aceste combinatii de atribute se numesc chei candidate

Chei

- Din toate combinatiile de coloane care pot fi utilizate pentru identificarea unica a unei linii se alege **macar** una utilizata intern de RDBMS pentru ordonarea datelor – **cheie primara**
  - Celelelte chei candidate devin **chei alternative** si pot fi folosite pentru eficientizarea prelucrarilor (crearea de "index" dupa aceste chei)
- In cazul in care nu exista o combinatie de coloane utilizabila ca si cheie cu utilitate practica se introduce artificial o cheie, cu numere intregi incrementate automat de DBMS (autoincrement)
  - de multe ori este recomandata o astfel de actiune, numerele intregi fiind mult mai usor de controlat, ordonat, cautat decat alte tipuri de date
  - cheile de tip autoincrement nu e **nevoie** sa contina informatie

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

Eliminarea datelor redundante

Order ID	SalesPerson	Hire Date	Phone	Company Name	Product Name	Quantity
10871	Dodsworth, Anne	15-Nov-1994	452	Bon app'	Alice Mutton	16
10747	Suyama, Michael	17-Oct-1993	428	Piccolo und mehr	Gorgonzola Telino	8
10258	Davolio, Nancy	01-May-1992	5467	Ernst Handel	Chef Anton's Gumbo Mix	65
11007	Callahan, Laura	05-Mar-1994	2344	Princesa Isabel Vinhos	Thüringer Rostbratwurst	10
10421	Callahan, Laura	05-Mar-1994	2344	Que Delicia	Perth Pasties	15
10558	Davolio, Nancy	01-May-1992	5467	Around the Horn	Perth Pasties	18
10431	Peacock, Margaret	03-May-1993	5176	Bottom-Dollar Markets	Alice Mutton	50
10659	King, Robert	02-Jan-1994	465	Queen Cozinha	Gorgonzola Telino	20
10273	Leverling, Janet	01-Apr-1992	3355	QUICK-Stop	Gorgonzola Telino	15
10382	Peacock, Margaret	03-May-1993	5176	Ernst Handel	Chef Anton's Gumbo Mix	32
10949	Fuller, Andrew	14-Aug-1992	3457	Bottom-Dollar Markets	Alice Mutton	6
10285	Davolio, Nancy	01-May-1992	5467	QUICK-Stop	Perth Pasties	36
10867	Suyama, Michael	17-Oct-1993	428	Lonesome Pine Restaur	Perth Pasties	3
10691	Fuller, Andrew	14-Aug-1992	3457	QUICK-Stop	Thüringer Rostbratwurst	40
10354	Callahan, Laura	05-Mar-1994	2344	Pericles Comidas clásic	Thüringer Rostbratwurst	4
10698	Peacock, Margaret	03-May-1993	5176	Ernst Handel	Thüringer Rostbratwurst	12
10962	Callahan, Laura	05-Mar-1994	2344	QUICK-Stop	Perth Pasties	20
10465	Davolio, Nancy	01-May-1992	5467	Vaffeljernet	Thüringer Rostbratwurst	18
10549	Buchanan, Steven	17-Oct-1993	3453	QUICK-Stop	Gorgonzola Telino	55

Eliminarea datelor redundante

Customers Relation

Customer ID	Company Name	Phone
ALFKI	Alfreds Futterkiste	030-0074321
ANATR	Ana Trujillo Emparedados y helados	(5) 555-4729
ANTON	Antonio Moreno Taquería	(5) 555-3932
AROUT	Around the Horn	(171) 555-7788
BERGS	Berglunds snabbköp	0921-12 34 65
BLAUS	Blauer See Delikatessen	0621-08460
BLONP	Blondel père et fils	88.60.15.31
BOLID	Bólido Comidas preparadas	(91) 555 22 82
BONAP	Bon app'	91.24.45.40
BOTTM	Bottom-Dollar Markets	(604) 555-4729
BSBEV	B's Beverages	(171) 555-1212
CACTU	Cactus Comidas para llevar	(1) 135-5555
CENTC	Centro comercial Moctezuma	(5) 555-3392

Invoices Relation

Order ID	Company Name	Phone
10643	Alfreds Futterkiste	030-0074321
10692	Alfreds Futterkiste	030-0074321
10702	Alfreds Futterkiste	030-0074321
10835	Alfreds Futterkiste	030-0074321
10952	Alfreds Futterkiste	030-0074321
11011	Alfreds Futterkiste	030-0074321
10308	Ana Trujillo Emparedados y helados	(5) 555-4729
10625	Ana Trujillo Emparedados y helados	(5) 555-4729
10759	Ana Trujillo Emparedados y helados	(5) 555-4729
10926	Ana Trujillo Emparedados y helados	(5) 555-4729
10365	Antonio Moreno Taquería	(5) 555-3932
10507	Antonio Moreno Taquería	(5) 555-3932
10535	Antonio Moreno Taquería	(5) 555-3932
10573	Antonio Moreno Taquería	(5) 555-3932
10677	Antonio Moreno Taquería	(5) 555-3932

When was she hired?

Order ID	SalesPerson	Hire Date	Phone	Company Name	Product Name
10871	Dodsworth, Anne	15-Nov-1994	452	Bon app'	Alice Mutton
10747	Suyama, Michael	17-Oct-1993	428	Piccolo und mehr	Gorgonzola Telino
10258	Davolio, Nancy	01-May-1992	5467	Ernst Handel	Chef Anton's Gumbo Mix
11007	Callahan, Laura	05-Mar-1994	2344	Princesa Isabel Vinhos	Thüringer Rostbratwurst
10421	Callahan, Laura	05-Mar-1994	2344	Que Delícia	Perth Pasties
10558	Davolio, Nancy	01-May-1992	5467	Around the Horn	Perth Pasties
10431	Peacock, Margaret	03-May-1993	5176	Bottom-Dollar Markets	Alice Mutton

Product ID	Product Name	Unit Price
1	Chai	$18.00
2	Chang	$19.00
3	Aniseed Syrup	$10.00
4	Chef Anton's Cajun Seasoning	$22.00
5	Chef Anton's Gumbo Mix	$21.35
6	Grandma's Boysenberry Spread	$25.00
7	Uncle Bob's Organic Dried Pears	$30.00
8	Northwoods Cranberry Sauce	$40.00
9	Mishi Kobe Niku	$97.00
10	Ikura	$31.00
11	Queso Cabrales	$21.00
12	Queso Manchego La Pastora	$38.00
13	Konbu	$6.00
14	Tofu	$23.25

These are not the same value

Order ID	Product Name	Unit Price	Quantity	Unit Price
10248	Mozzarella di Giovanni	$34.80	5	$174.00
10248	Queso Cabrales	$21.00	12	$168.00
10248	Singaporean Hokkien Fried Mee	$14.00	10	$98.00
10249	Manjimup Dried Apples	$53.00	40	$1,696.00
10249	Tofu	$23.25	9	$167.40

Prima forma normala

- toate valorile sunt scalare

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

- nu toate rezolvarile sunt eficiente

OrderID	CustomerID	Item1	Qty1	Item2	Qty2	Item3	Qty3
1	ANTON	Queso Cabrales	4	Tofu	3	Ravioli Angelo	1
2	BLAUS	Chai	2		0		

Product	Year	TargetJan	ActualJan	TargetFeb	ActualFeb
Aniseed Syrup	2004	$1,000.00	$1,300.00	$0.00	$0.00
Chai	2004	$4,000.00	$2,000.00	$0.00	$0.00
Chang	2004	$3,000.00	$8,022.00	$0.00	$0.00

A doua forma normala

- O relatie este in a **doua** forma normala cand este in **prima** forma normala si suplimentar atributele (valorile de pe coloana) depind de **intreaga cheie** candidata aleasa

Product Name	SupplierName	Category Name	SupplierPhoneNumber
Chai	Exotic Liquids	Beverages	(171) 555-2222
Chang	Exotic Liquids	Beverages	(171) 555-2222
Guaraná Fantástica	Refrescos Americanas LTDA	Beverages	(11) 555 4640
Sasquatch Ale	Bigfoot Breweries	Beverages	(503) 555-9931
Steeleye Stout	Bigfoot Breweries	Beverages	(503) 555-9931
Côte de Blaye	Aux joyeux ecclésiastiques	Beverages	(1) 03.83.00.68
Chartreuse verte	Aux joyeux ecclésiastiques	Beverages	(1) 03.83.00.68
Ipoh Coffee	Leka Trading	Beverages	555-8787
Laughing Lumberjack Lager	Bigfoot Breweries	Beverages	(503) 555-9931
Outback Lager	Pavlova, Ltd.	Beverages	(03) 444-2343

A doua forma normala

Product ID	Product Name	Category
1	Chai	Beverages
2	Chang	Beverages
3	Aniseed Syrup	Condiments
4	Chef Anton's Cajun Seasoning	Condiments
5	Chef Anton's Gumbo Mix	Condiments
6	Grandma's Boysenberry Spread	Condiments
7	Uncle Bob's Organic Dried Pears	Produce

Supplier ID	SupplierName	SupplierPhoneNumber
1	Exotic Liquids	(171) 555-2222
2	New Orleans Cajun Delights	(100) 555-4822
3	Grandma Kelly's Homestead	(313) 555-5735
4	Tokyo Traders	(03) 3555-5011
5	Cooperativa de Quesos 'Las Cabras'	(98) 598 76 54
6	Mayumi's	(06) 431-7877
7	Pavlova, Ltd.	(03) 444-2343
8	Specialty Biscuits, Ltd.	(161) 555-4448
9	PB Knäckebröd AB	031-987 65 43

A treia forma normala

- O relatie este in a **treia** forma normala cand este in a **doua** forma normala si suplimentar atributele (valorile de pe coloana) care nu fac parte din cheie sunt **mutual independente**

	Company Name	Address	City	Region	Postal Code
	Exotic Liquids	49 Gilbert St.	London		EC1 4SD
	New Orleans Cajun Delights	P.O. Box 78934	New Orleans	LA	70117
	Grandma Kelly's Homestead	707 Oxford Rd.	Ann Arbor	MI	48104
	Tokyo Traders	9-8 Sekimai	Tokyo		100
	Cooperativa de Quesos 'Las Cabras'	Calle del Rosal 4	Oviedo	Asturias	33007
	Mayumi's	92 Setsuko	Osaka		545
	Pavlova, Ltd.	74 Rose St.	Melbourne	Victoria	3058

A treia forma normala

	Company Name	Address	City
	Exotic Liquids	49 Gilbert St.	London
	New Orleans Cajun Delights	P.O. Box 78934	New Orleans
	Grandma Kelly's Homestead	707 Oxford Rd.	Ann Arbor
	Tokyo Traders	9-8 Sekimai	Tokyo
	Cooperativa de Quesos 'Las Cabras'	Calle del Rosal 4	Oviedo
	Mayumi's	92 Setsuko	Osaka
	Pavlova, Ltd.	74 Rose St.	Melbourne

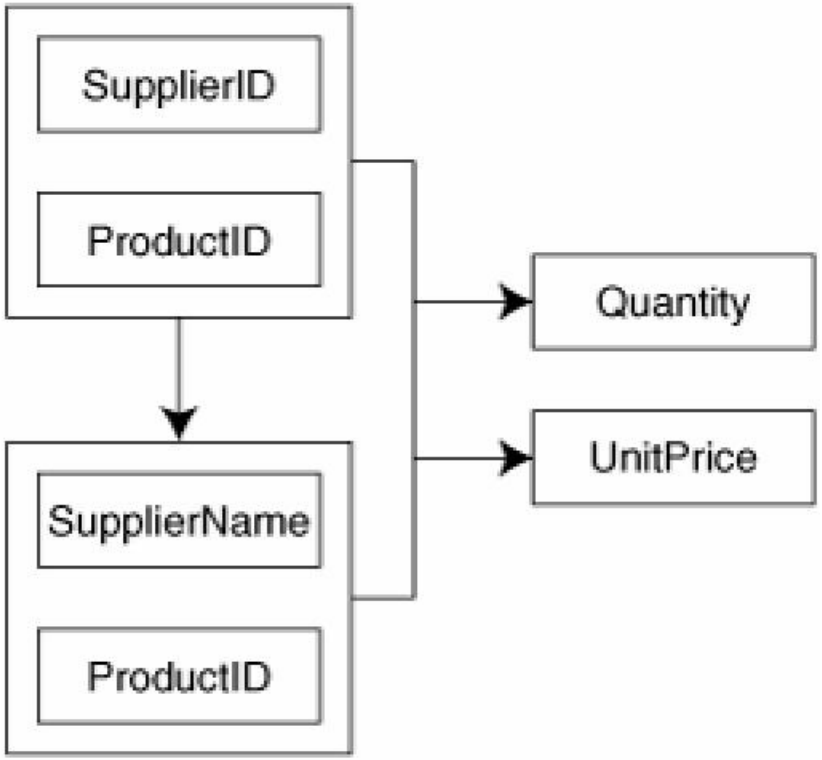
	City	Region	Postal Code
	Melbourne	Victoria	3058
	Ste-Hyacinthe	Québec	J2S 7S8
	Montréal	Québec	H1J 1C3
	Bend	OR	97101
	Sydney	NSW	2042
	Ann Arbor	MI	48104
	Boston	MA	02134
	New Orleans	LA	70117
	Oviedo	Asturias	33007

Normalizare suplimentara

- Se tine cont si de eliminarea datelor redundante. Anumite redundante pot fi eliminate prin introducerea de relatii suplimentare
- Forma normala Boyce/Codd cere sa nu existe dependenta functionala intre cheile candidate

Supplier ID	SupplierName	Product	Quantity	Unit Price
5	Cooperativa de Quesos 'Las Cabras'	Queso Cabrales	12	$14.00
20	Leka Trading	Singaporean Hokkien Fried Mee	10	$9.80
14	Formaggi Fortini s.r.l.	Mozzarella di Giovanni	5	$34.80
24	G'day, Mate	Manjimup Dried Apples	40	$42.40
6	Mayumi's	Tofu	9	$18.60
24	G'day, Mate	Manjimup Dried Apples	35	$42.40
19	New England Seafood Cannery	Jack's New England Clam Chowder	10	$7.70
2	New Orleans Cajun Delights	Louisiana Fiery Hot Pepper Sauce	15	$16.80

Normalizare suplimentara



Supplier ID	SupplierName
1	Exotic Liquids
2	New Orleans Cajun Delights
3	Grandma Kelly's Homestead
4	Tokyo Traders
5	Cooperativa de Quesos 'Las Cabras'
6	Mayumi's

SupplierID	ProductID	Quantity	UnitPrice
2	65	15	$21.05
24	53	15	$32.80
8	20	40	$81.00
22	47	16	$9.50
6	14	9	$23.25
28	59	30	$55.00
28	60	40	$34.00
21	46	15	$12.00

MySql

Relatii in Bazele de date

Relatii in Bazele de date

- Legaturile intre tabele pot fi
  - One to One
  - One to Many
  - Many to Many
    - Unare (auto referinta)

One to One

- Fiecare tabel poate avea corespondenta **o singura linie (row) sau nici una** de cealalta parte a relatiei
- echivalent cu o relatie "bijectiva"
- analogie cu casatorie:
  - o persoana poate fi casatorita sau nu
  - daca este casatorita va fi casatorita cu o singura persoana din tabelul cu persoane de sex opus
  - persoana respectiva va fi caracterizata de aceeasi relatie "one to one" – primeste simultan un singur corespondent in tabelul initial

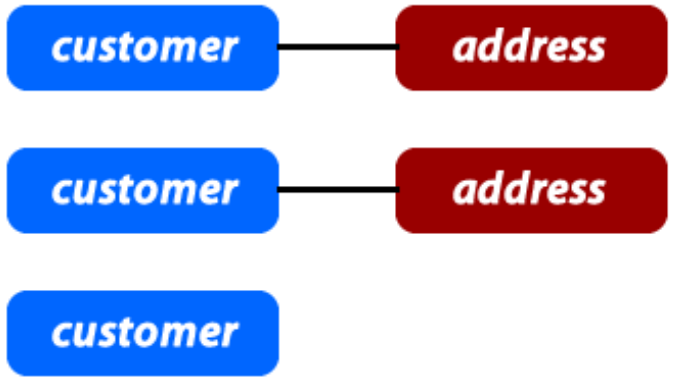
One to One

- de multe ori legaturile "one to one" se bazeaza pe reguli externe
- de obicei se poate realiza usor si eficient gruparea ambelor tabele in unul singur

CUSTOMERS		
customer_id	customer_name	address_id
101	John Doe	301
102	Bruce Wayne	302

ADDRESSES	
address_id	address
301	12 Main St., Houston TX 77001
302	1007 Mountain Dr., Gotham NY 10286

CUSTOMERS		
customer_id	customer_name	customer_address
101	John Doe	12 Main St., Houston TX 77001
102	Bruce Wayne	1007 Mountain Dr., Gotham NY 10286



One to Many

- O linie dintr-un tabel (row), identificata prin cheia primara, poate avea: **nici una, una sau mai multe linii corespondente** in celalalt tabel. In acesta o linie poate fi legata cu o **singura** linie din tabelul primar.
- Analogie cu relatii parinte/copil:
  - fiecare om are o singura mama
  - fiecare femeie poate avea nici unul, unul sau mai multi copii

One to Many, Many to One

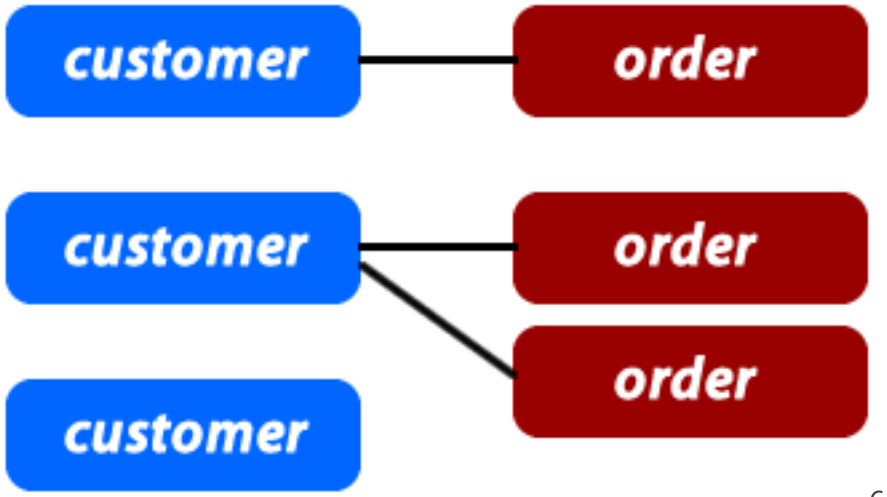
- de obicei aceste legaturi se implementeaza prin introducerea cheii primare din tabelul **One** in calitate de coloana in tabelul **Many** (cheie externa – foreign key)

CUSTOMERS

customer_id	customer_name
101	John Doe
102	Bruce Wayne

ORDERS

order_id	customer_id	order_date	amount
555	101	12/24/09	$156.78
556	102	12/25/09	$99.99
557	101	12/26/09	$75.00



Many to Many

- Fiecare linie (row) din **ambele tabele** implicate in legatura poate fi legat cu **oricate (niciuna, una sau mai multe) linii** din tabelul corespondent.
- Analogie cu relatii de rudenie (veri de exemplu), tabel 1 – barbati, tabel 2 – femei :
  - fiecare barbat poate fi ruda cu una sau mai multe femei
  - la randul ei fiecare femeie poate fi ruda cu unul sau mai multi barbati

Many to Many

- de obicei aceste legaturi se implementeaza prin introducerea unui tabel **suplimentar** (numit tabel **asociat** sau de **legatura**) care sa memoreze legaturile

ORDERS

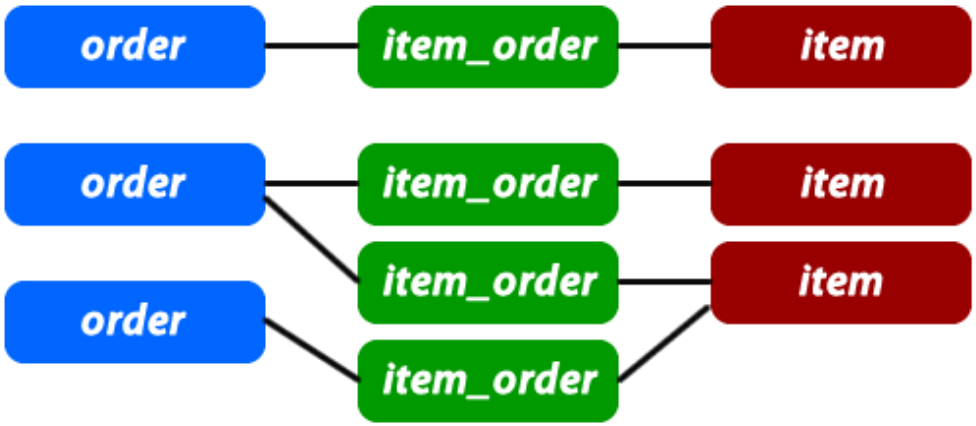
order_id	customer_id	order_date	amount
555	101	12/24/09	$156.78
556	102	12/25/09	$99.99

ITEMS

item_id	item_name	item_description
201	Tickle Me Elmo	It wants to be tickled
202	District 9 DVD	Awesome sci-fi movie
203	Batarang	It is very sharp

ITEMS_ORDERS

order_id	item_id
555	201
555	202
556	202
556	203

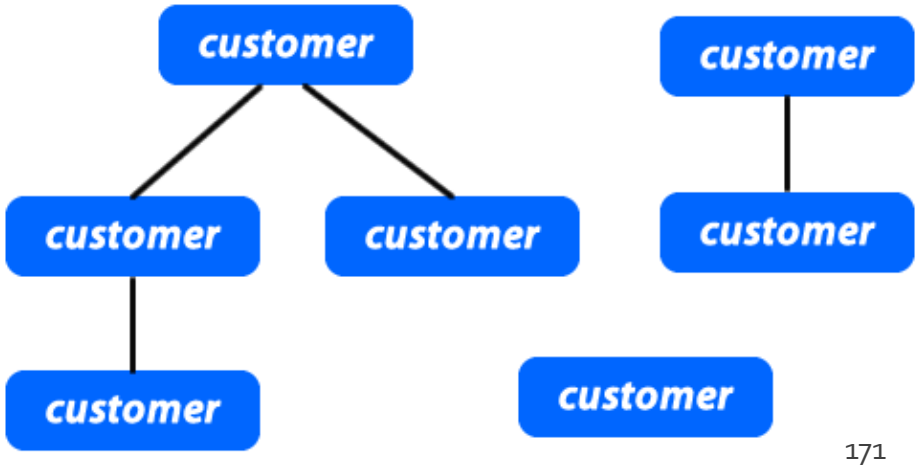


Self Referencing (unare)

- Un caz particular de legatura "one to many" in care legatura e in interiorul aceluiasi tabel
- rezolvarea este similara, introducerea unei coloane suplimentara, cu referinta la cheia primara din tabel
- analogie cu relatii parinte copil cand ambele persoane se regasesc in acelasi tabel

CUSTOMERS

customer_id	customer_name	referrer_customer_id
101	John Doe	0
102	Bruce Wayne	101
103	James Smith	101



Relatii in Bazele de date

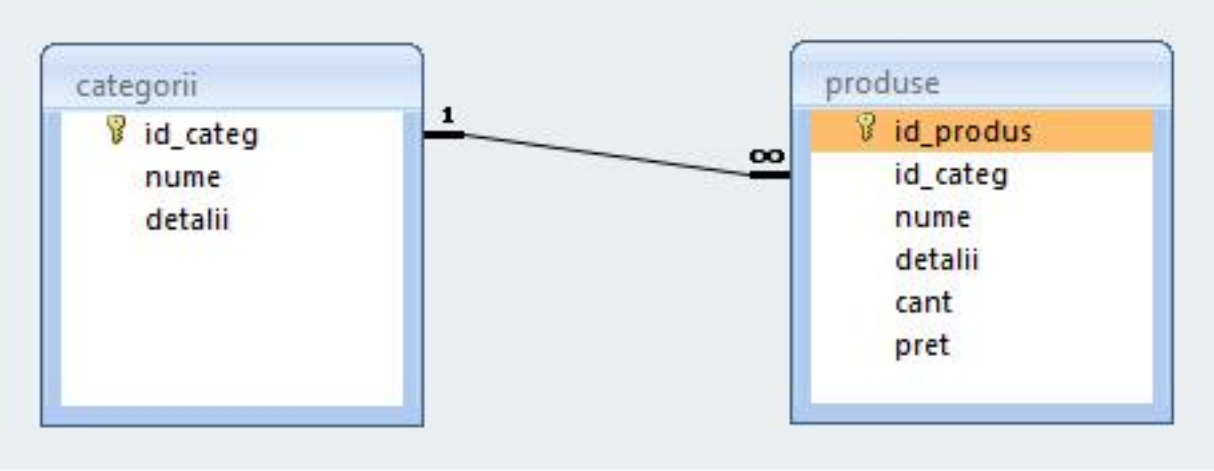
- Respectarea formelor normale ale bazelor de date aduce nenumarate avantaje
- Efectul secundar este dat de necesitatea separarii datelor intre mai multe tabele
- In exemplul utilizat avem doua concepte diferite din punct de vedere logic
  - produs
  - categorie de produs

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

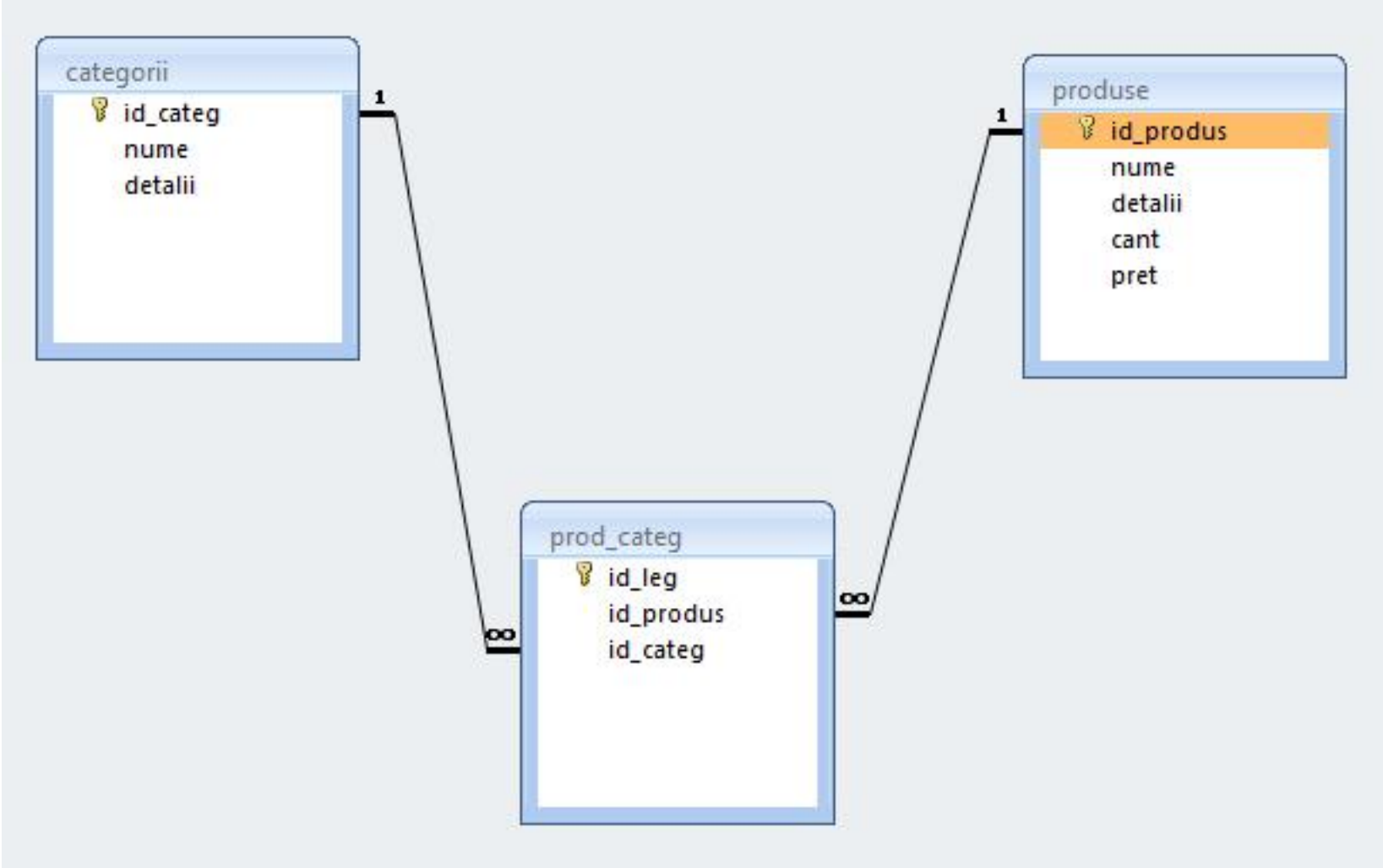
- Legaturile implementata
  - One to Many
  - in tabelul “produse” apare cheia externa (foreign key): “id_categ”



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 (**R**DB**MS**)
  - 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**

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 = 4GB

MySql – tipuri de date

- enumerare
  - ENUM('val1','val2',...)
    - una singura din cele maxim 65535 valori distincte posibile
  - SET('val1','val2',...)
    - niciuna sau mai multe din cele maxim 64 valori distincte
    - echivalent cu "setare de biti" intr-un intreg pe 64 biti cu tabela asociata

Metode de stocare

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 pierd**, 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

Limbaj SQL

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 cuvantului 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(nume," ",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)

Interogari SQL

Interogari

- Interogarile 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**

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 aspra 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
- Fisierele 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
- Fisierele SQL “backup” contin succesiunea DROP..., CREATE... pentru a inlocui datele in intregime

Interogari SQL

Interogari

- Interogarile 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**

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 acepta functii aggregate, dar se aplica la sfarsit, exact inainte de a fi trimise datele clintului, **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 aggregate (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 cuvantului 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 SELECT * FROM clienti INNER JOIN comenzi USING (id_client);
```

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 SELECT * FROM clienti INNER JOIN comenzi ON (clienti.id_client=comenzi.id_client);
```

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 SELECT * FROM clienti NATURAL JOIN comenzi;
```

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 tabel1 SET ... SELECT ... FROM tabel1 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)

Contact

- Laboratorul de microunde si optoelectronica
- https://rf-opto.etti.tuiasi.ro
- rdamian@etti.tuiasi.ro