Curs 8

2022/2023

Programarea aplicațiilor web

PAW

- Programarea aplicațiilor web
  - An V RC
    - 1.5C/1L/1P

Program

- An V
  - Saptamana 1
    - Luni 17-20 Curs (Intro/HTML/CSS)
  - Saptamanile 2-8
    - Luni 16(17)-18 Curs
    - Luni 18-20 Laborator
  - Saptamanile 9-14
    - Luni 16(17)-18 Curs
    - Luni 18-20 Proiect

Orar

- https://orar.etti.tuiasi.ro/ : C->16(17)-18, L/P -> 18

FACULTATEA DE ELECTRONICA, TELECOMUNICATII SI TEHNOLOGIA INFORMATIEI

55RC

ETTI_

	1 8:00 - 8:50	2 9:00 - 9:50	3 10:00 - 10:50	4 11:00 - 11:50	5 12:00 - 12:50	6 13:00 - 13:50	7 14:00 - 14:50	8 15:00 - 15:50	9 16:00 - 16:50	10 17:00 - 17:50	11 18:00 - 18:50	12 19:00 - 19:50
L										Damian R. PAW (C) 2.13 TC (R)		
									Damian R. PAW (P) 2.13 TC (R)		Damian R. PAW (L) 2.13 TC (R)	
Ma								Scripcariu L. RCALSC (C) C1			Scripcariu L. RCALSC (L) 2.13 TC (R)	
Mi								Casian-Botez I. Etic (C) P6 (Amf.)	Casian-Botez I. Etic (S) P6 (Amf.)		Trifina L. TEFO (L) 3.25 TTI (L)	
										Alecsandrescu I. POO (L) CI5(Corp C)		
J									Sirbu A. POO (C) P2 (Amf.)		Trifina L. TEFO (L) 3.25 TTI (L)	
V										Trifina L. TEFO (C) 3.25 TTI (L)		

Nota

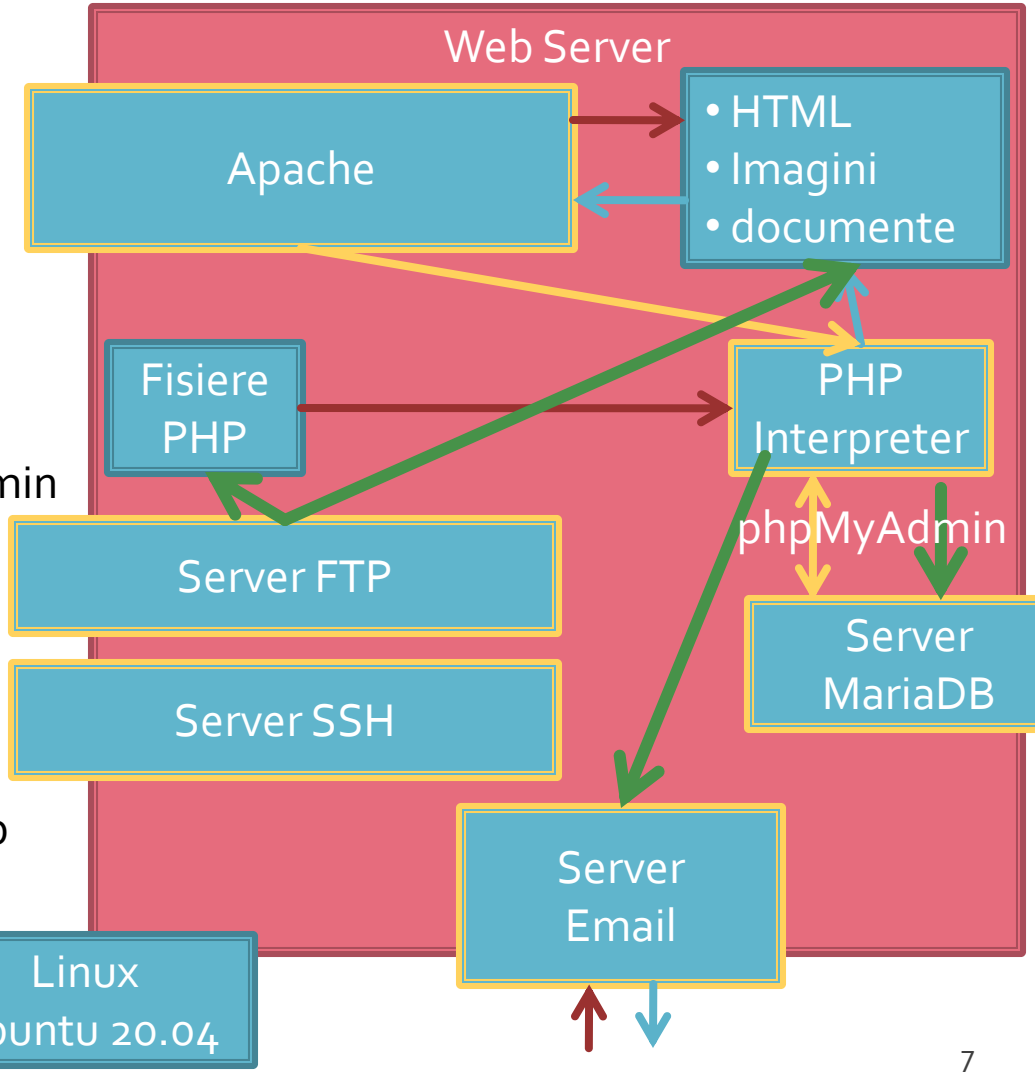
- An V
  - 33% E
  - 66% Aplicatii
    - 33% L
    - 33% P

Nota

- An V
  - 33% E
  - 33% L
  - 33% P
- Laborator - Prezenta
  - 1pz = 1p (p>5 ←→ Curs)
- Examen
  - Prezenta la curs: 3pz = 0.5p
  - Asemanator cu materia de **proiect**
- Activitate suplimentara
  - Dupa terminarea activitatii la laborator
  - +~~2~~ **1**p la E/L

Utilizare LAMP

1. login → **paw**:masteretti
2. ifconfig → 192.168.30.5
3. putty.exe → 192.168.30.5 → SSH → **paw**:masteretti (remote login)
4. [alte comenzi linux dorite]
5. FTP → Winscp → SFTP → student:masterrc@192.168.30.5
6. MySql → http://192.168.30.5/phpmyadmin → **root**: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



Tema bonus

- logfile.php
  - Afiseaza log Apache (erori php majore)
- **1p ~~2p~~ suplimentar** la laborator/examen
- Modificare logfile.php pentru a afisa **toate** erorile PHP
  - php.ini – activare erori √
  - php.ini – locatie erori √
  - logfile.php – afisare log PHP √
  - drepturi de acces la fisiere (Linux)

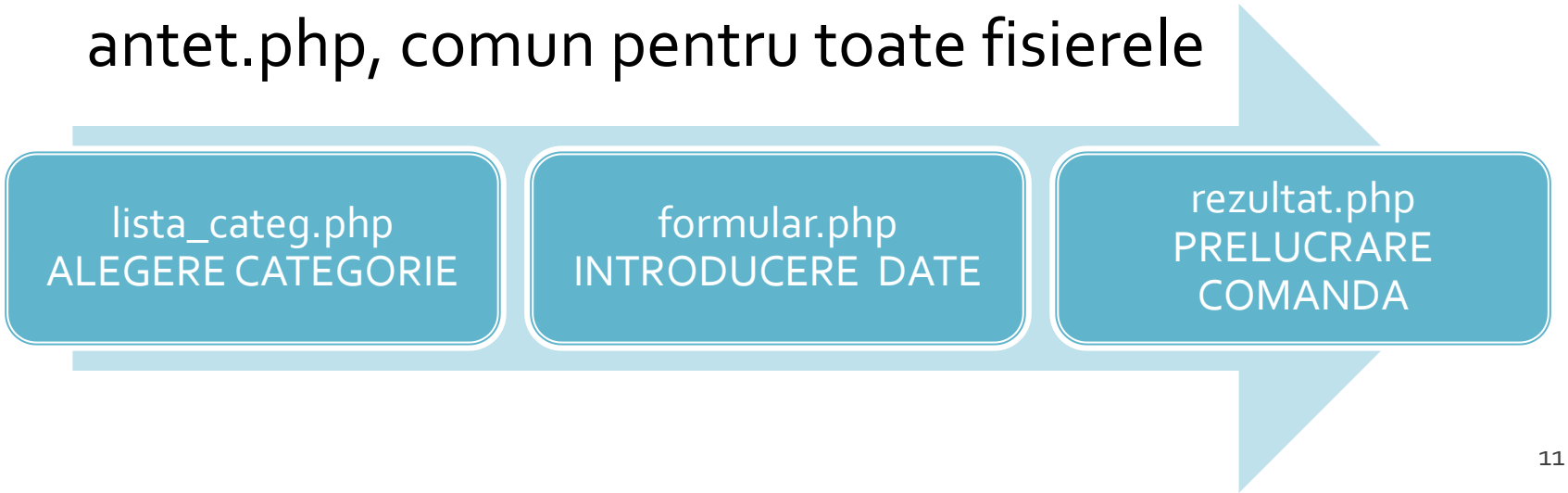
Server referinta LAMP

- Linux, doua variante
  - Centos 7.1
    - PHP 5.4.16
    - MariaDB 5.5.44 / root:masterrc
    - Apache 2.4.6
    - **root**/student:masterrc
  - **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 paw**:
      - **sudo usermod -a -G upload paw**
      - **sudo chmod -R 775 /var/www**

Laborator 6

Plan aplicatie – Cumparator

- Pe masura ce aplicatia paraseste un fir liniar de executie este necesara introducerea unui plan (graf) al aplicatiei
- Cumparator
  - citirea fisierului text (XML) se realizeaza in antet.php, comun pentru toate fisierele



Rezultat (cumparator)

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

Magazin online Firma X SRL

alizati 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

Magazin online Firma X SRL

Rezultate comanda

Pret total (fara TVA): 350

Pret total (cu TVA): 416.5

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

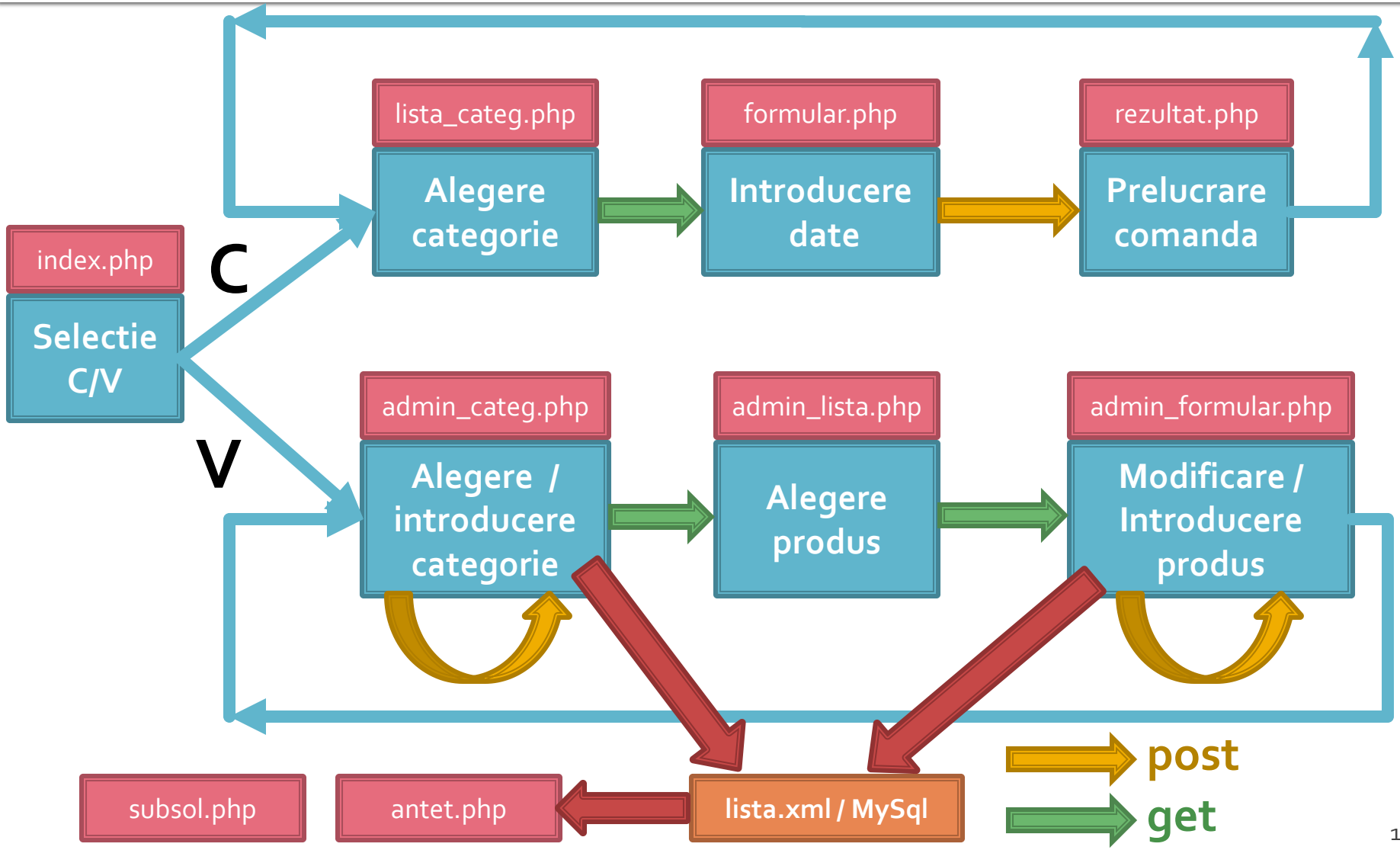
post

get

Plan aplicatie – Vanzator

- Aparitia aplicatiei pentru vanzator
  - introduce un fir paralel de executie cu necesitatea alegerii initiale: cumparator/vanzator
  - aduce posibilitatea scrierii fisierului XML
  - diverse operatii de scriere
    - introducere categorie de produse
    - introducere produs nou intr-o categorie existenta
    - modificare produs existent
  - modificarea fisierului implica 2 actiuni:
    - colectare date
    - prelucrare

Plan aplicatie (Proiect !!)



Rezultat (vanzator)

Magazin Firma X

Inceput | Inapoi

Magazin online Firma X SRL

Alegeti:

- Cumparator
- Vanzator

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

Categorie noua de produse:

Trimite

Lista produse in categoria Calculatoare

Nr.	Produs	Descriere	Pret	Cantitate	Actiuni
1	Laptop	calculator mic	2000	2	modifica
2	Desktop	calculator mare	1000	5	modifica
3	Imprimanta	prn	200	2	modifica
-	**Produs nou**				adauga

Produs in categoria Calculatoare

Produs	laptop
Descriere	calculator mic
Pret	2000
Cantitate	2

Trimite

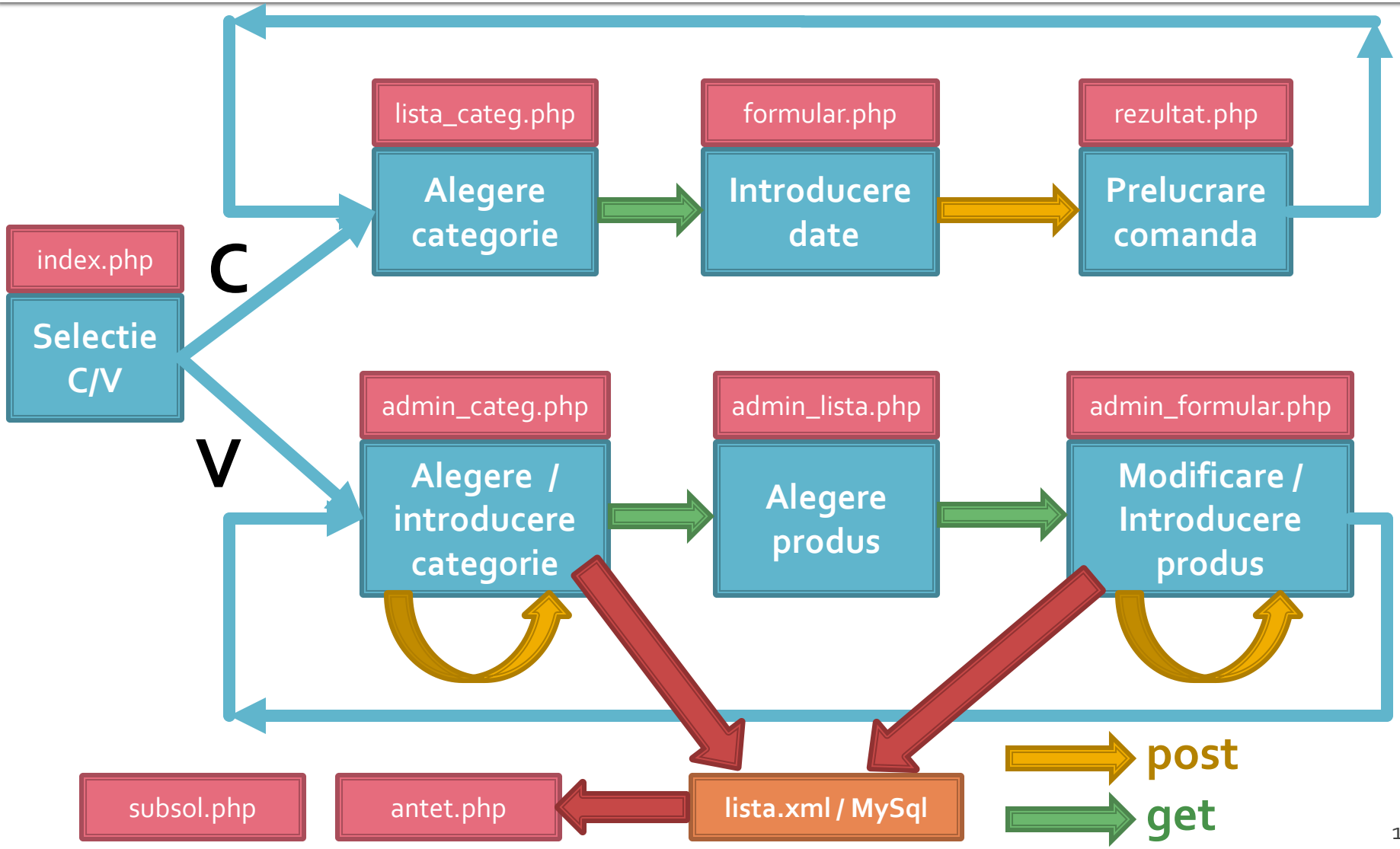
post

get

Laborator 6(7)

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

Plan aplicatie



Stocare eficienta a datelor

MySql

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

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

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

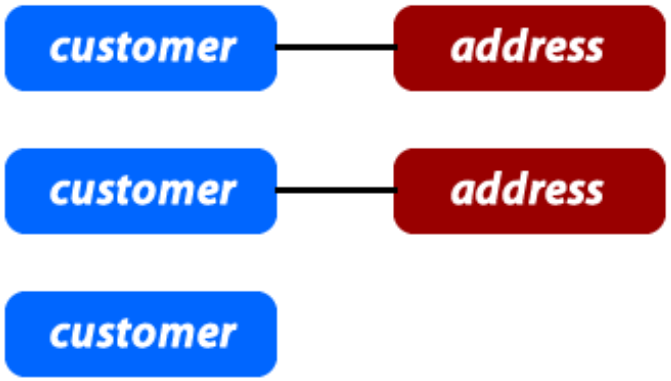
One to One

- 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

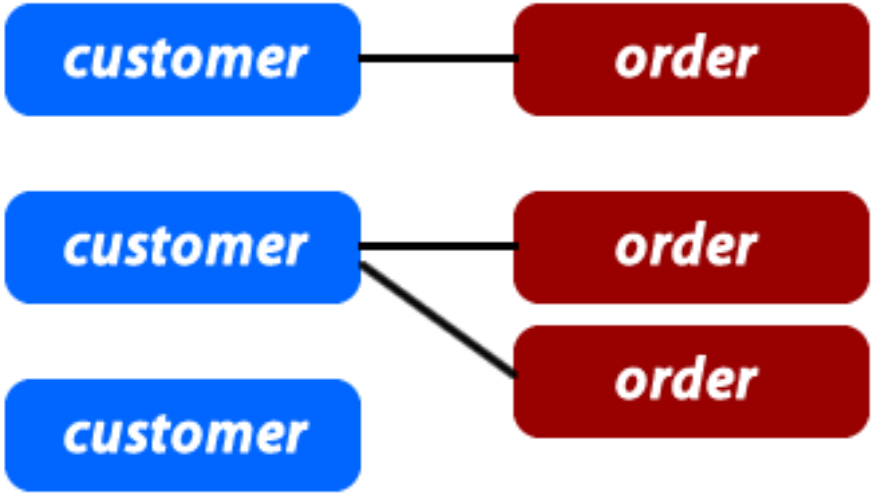
- de obicei aceste legaturi se implementeaza prin introducerea cheii primare din tabelul **One** in calitate de coloana in tabelul **Many** (cheie externa – foreign key)

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

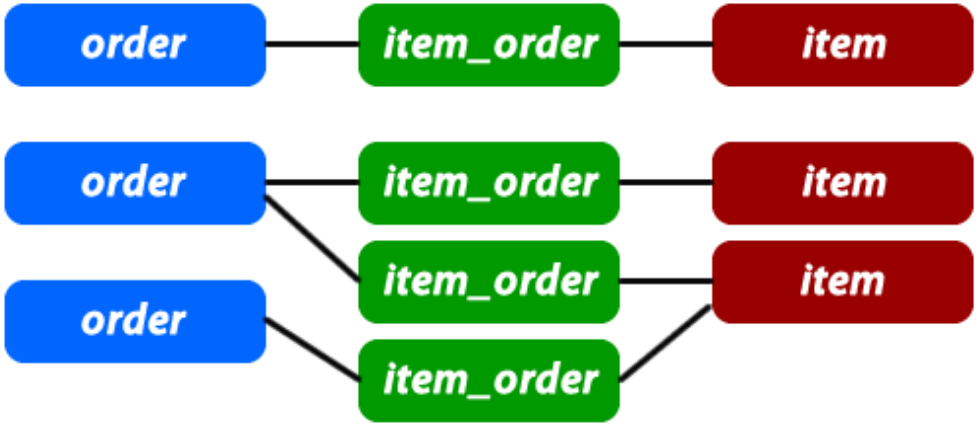
order_id	customer_id	order_date	amount
555	101	12/24/09	$156.78
556	102	12/25/09	$99.99

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

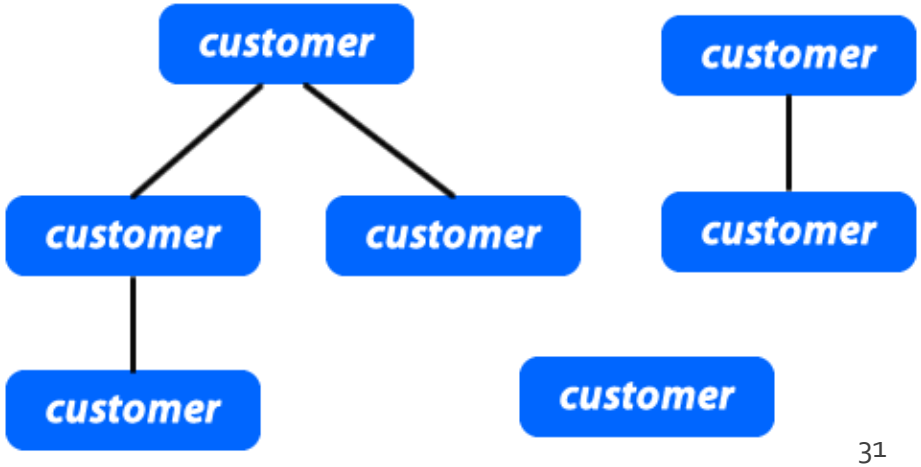
order_id	item_id
555	201
555	202
556	202
556	203



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

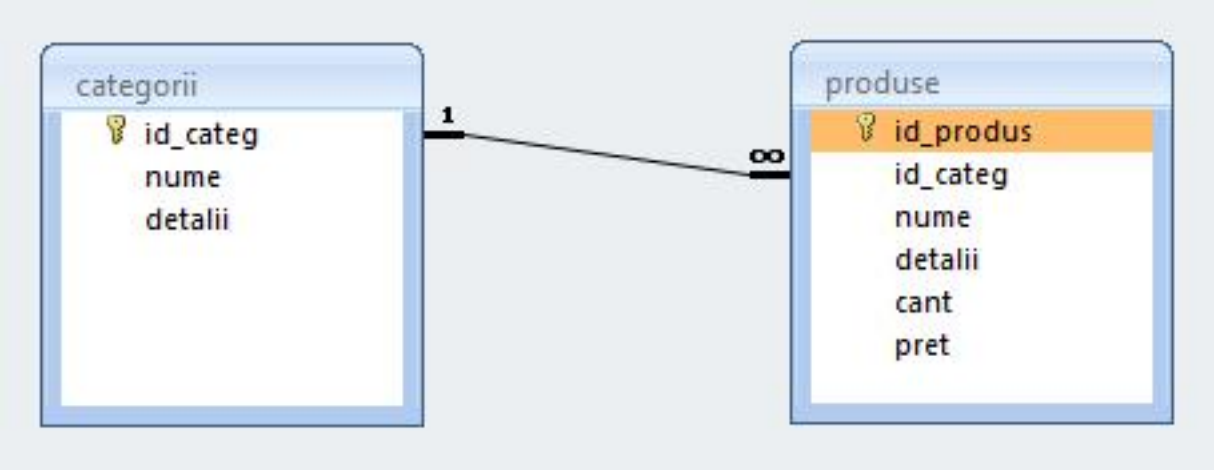
- Respectarea formelor normale ale bazelor de date aduce nenumarate avantaje
- Efectul secundar este dat de necesitatea separarii datelor intre mai multe tabele
- In exemplul utilizat avem doua concepte diferite din punct de vedere logic
  - produs
  - categorie de produs

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

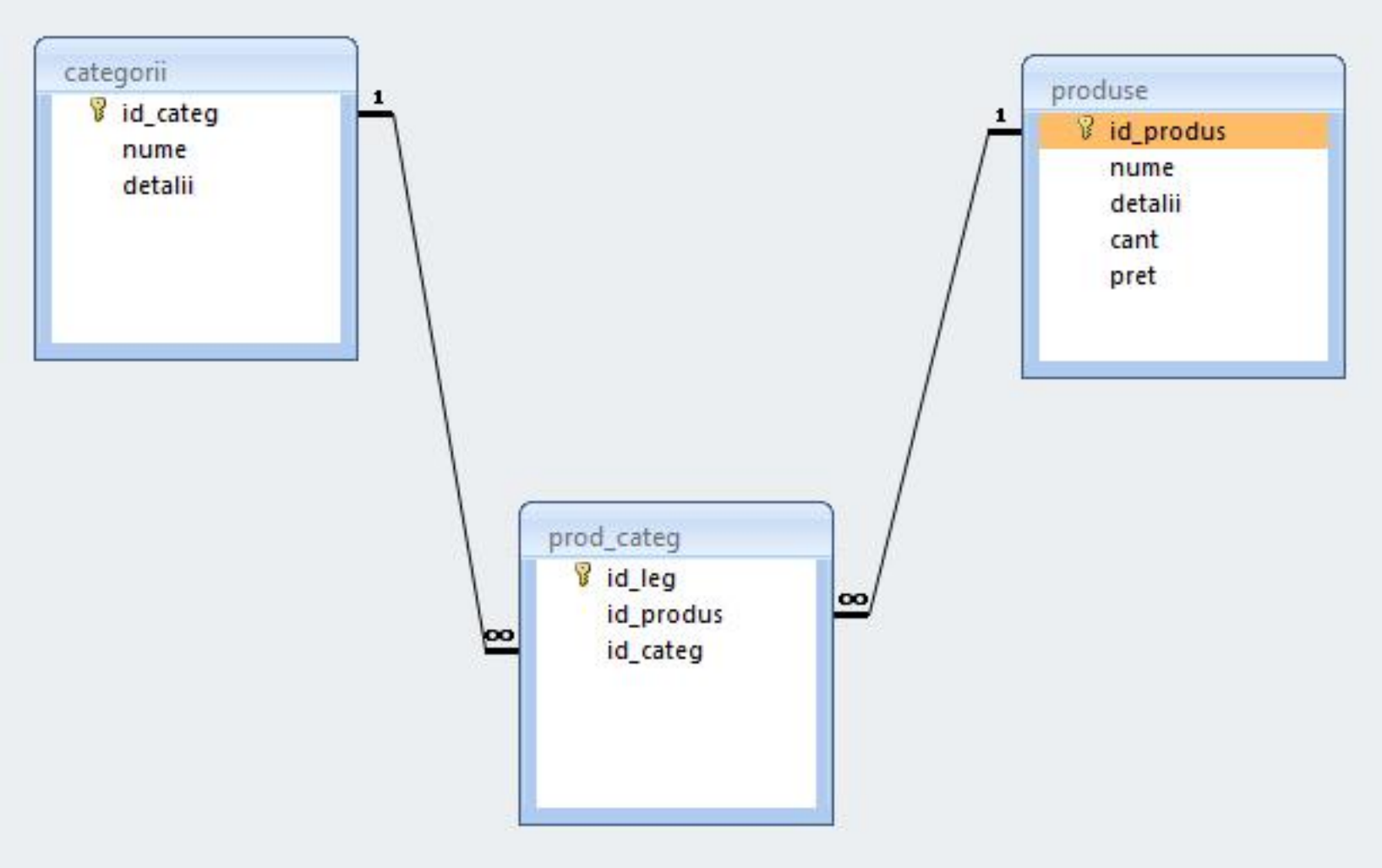
- Legaturile implementata
  - One to Many
  - in tabelul "produse" apare cheia externa (foreign key): "id_categ"



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

Acces la server-ul MySql din PHP

Functii PHP de acces MySql

- mysql_connect
  - realizarea unei conexiuni cu server-ul MySQL
  - resource **mysql_connect** (string server, string user, string password)
  - rezultatul
    - succes – resursa (conexiune, link_identifier)
    - esec – false
- mysql_select_db
  - selecteaza **baza de date** de pe server cu care se va lucra in continuare
  - bool **mysql_select_db** (string database [, resource link_identifier])

PHP 5.5

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

Extensia mysqli

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

mysqli – Procedural

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

$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'];
?>
```

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

mysqli – Programare orientata obiect

```
<?php
$var = new mysqli("example.com", "user", "password", "database");
$res = $var->query ($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'];
?>
```

Resurse MySQL – mysqli

Structura

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

Date

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

Metode

Constructor	query	fetch_assoc	....

Functii de acces la structura

Functii de acces la date

~~Acces direct~~

Metode atasate resursei

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 7-8):
  - 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 clientului, **fara nici o optimizare** – utilizarea este recomandata doar cand nu exista echivalent WHERE

SELECT

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

Laborator 2 / 2011-2012

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

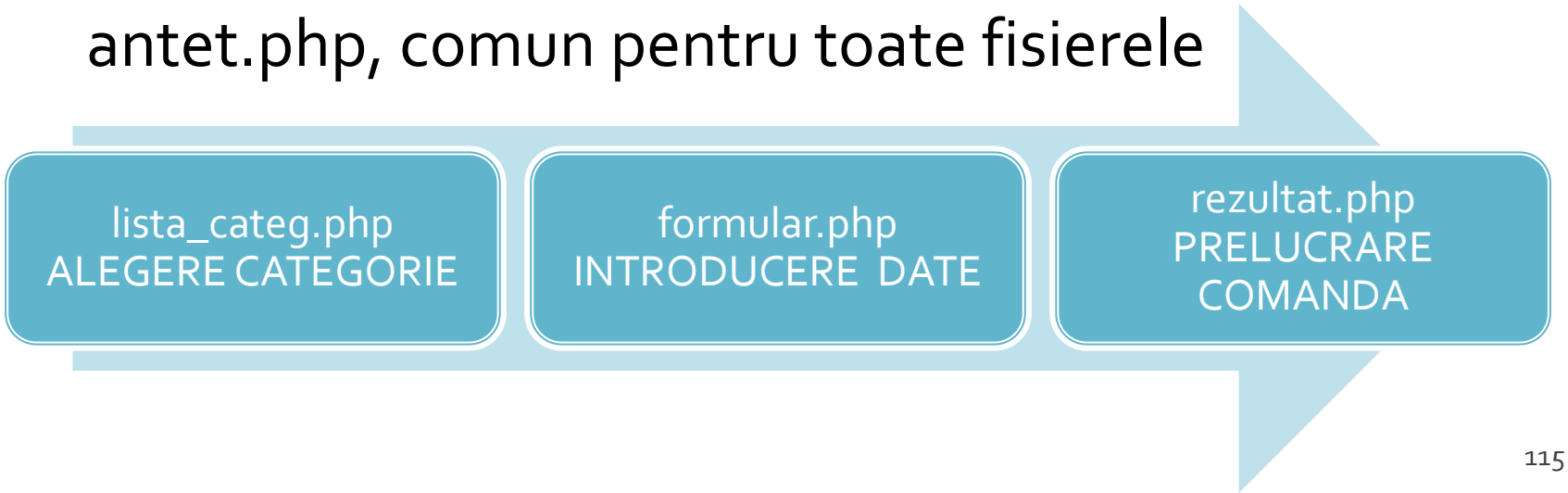
MySql – eficienta

- 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**

Laborator 7

Plan aplicatie – Cumparator

- Pe masura ce aplicatia paraseste un fir liniar de executie este necesara introducerea unui plan (graf) al aplicatiei
- Cumparator
  - citirea fisierului text (XML) se realizeaza in antet.php, comun pentru toate fisierele



Rezultat (cumparator)

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

Magazin online Firma X SRL

alizati 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

Magazin online Firma X SRL

Rezultate comanda

Pret total (fara TVA): 350

Pret total (cu TVA): 416.5

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

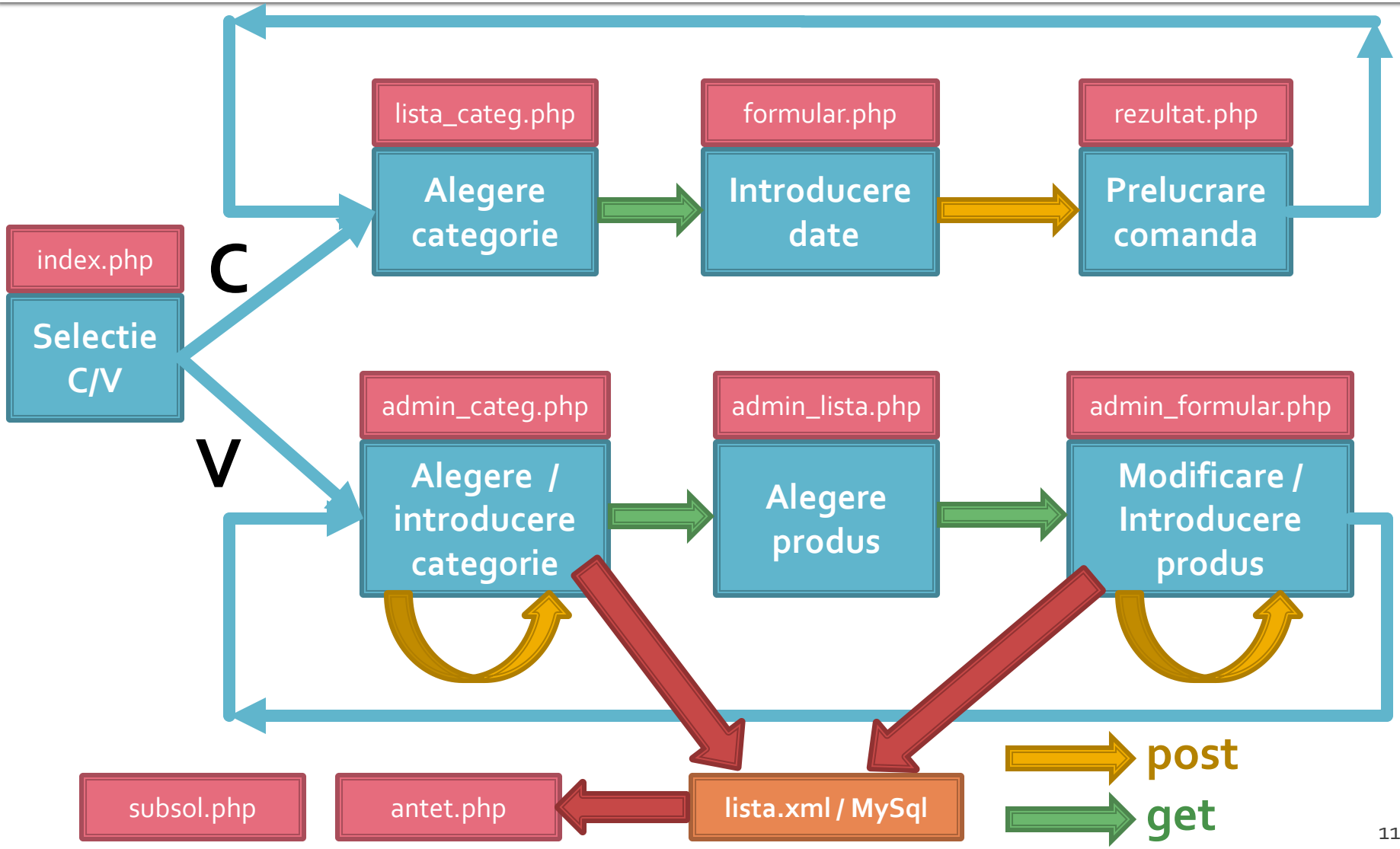
post

get

Plan aplicatie – Vanzator

- Aparitia aplicatiei pentru vanzator
  - introduce un fir paralel de executie cu necesitatea alegerii initiale: cumparator/vanzator
  - aduce posibilitatea scrierii fisierului XML
  - diverse operatii de scriere
    - introducere categorie de produse
    - introducere produs nou intr-o categorie existenta
    - modificare produs existent
  - modificarea fisierului implica 2 actiuni:
    - colectare date
    - prelucrare

Plan aplicatie (Proiect !!)



Rezultat (vanzator)

Magazin Firma X

Inceput | Inapoi

Magazin online Firma X SRL

Alegeti:

- Cumparator
- Vanzator

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

Categorie noua de produse: Trimite

Lista produse in categoria Calculatoare

Nr.	Produs	Descriere	Pret	Cantitate	Actiuni
1	Laptop	calculator mic	2000	2	modifica
2	Desktop	calculator mare	1000	5	modifica
3	Imprimanta	prn	200	2	modifica
-	**Produs nou**				adauga

Produs in categoria Calculatoare

Produs	laptop
Descriere	calculator mic
Pret	2000
Cantitate	2

Trimite

post

get

Laborator 6/7

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

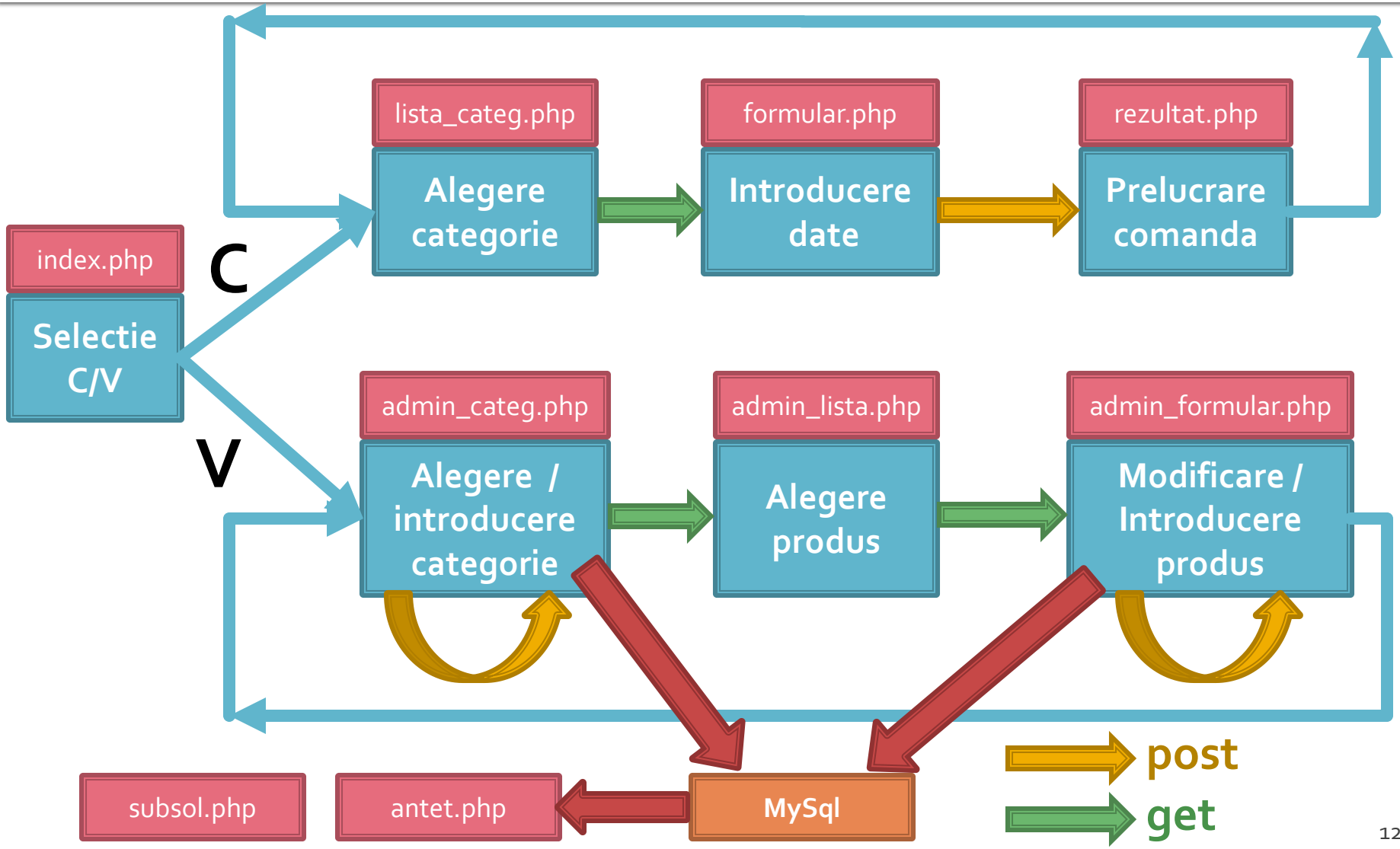
Laborator 6/7 – Mod de lucru

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

Laborator 6/7 – Mod de lucru

- Se ia o decizie relativ la relatia dintre produse si categorii (S34-S36)
  - One to Many
  - Many to Many
- Se creaza cele 2(3) tabele corespunzatoare
- Se populeaza cu date de start
- Se actualizeaza planul aplicatiei pentru a corespunde cu aplicatia proprie
  - nume de fisiere, tipuri de transfer a datelor

Plan aplicatie



Rezultat (cumparator)

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

Magazin online Firma X SRL

alizati 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

Magazin online Firma X SRL

Rezultate comanda

Pret total (fara TVA): 350

Pret total (cu TVA): 416.5

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

post

get

Rezultat (vanzator)

Magazin Firma X

Inceput | Inapoi

Magazin online Firma X SRL

Alegeti:

- Cumparator
- Vanzator

Categorii Produse

Alegeti categoria:

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

Total produse: 14

Categorie noua de produse: Trimite

Lista produse in categoria Calculatoare

Nr.	Produs	Descriere	Pret	Cantitate	Actiuni
1	Laptop	calculator mic	2000	2	modifica
2	Desktop	calculator mare	1000	5	modifica
3	Imprimanta	prn	200	2	modifica
-	**Produs nou**				adauga

Produs in categoria Calculatoare

Produs	laptop
Descriere	calculator mic
Pret	2000
Cantitate	2

Trimite

post

get

Laborator 6/7 – Mod de lucru

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

Plan aplicatie – vanzator

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

admin_categ.php

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

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

admin_categ.php

Magazin Firma X SRL

Inceput | Inapoi

Magazin online Firma X SRL

Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	Papetarie	3
2	Instrumente	3
3	Audio-video	3

Total produse: 9

Categorie noua de produse:
Nume: jucarii
Descriere: pentru copii
Trimite

Magazin Firma X SRL

Inceput | Inapoi

Magazin online Firma X SRL

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

Categoria jucarii adaugata! Are id = 4

Categorii Produse

Alegeti categoria:

Nr.	Categorie	Total Produse
1	Papetarie	3
2	Instrumente	3
3	Audio-video	3
4	Jucarii	0

Total produse: 9

Categorie noua de produse:
Nume:
Descriere:
Trimite

Magazin online Firma X SRL

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

Categoria jucarii adaugata! Are id = 4

admin_formular.php

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

admin_formular.php

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

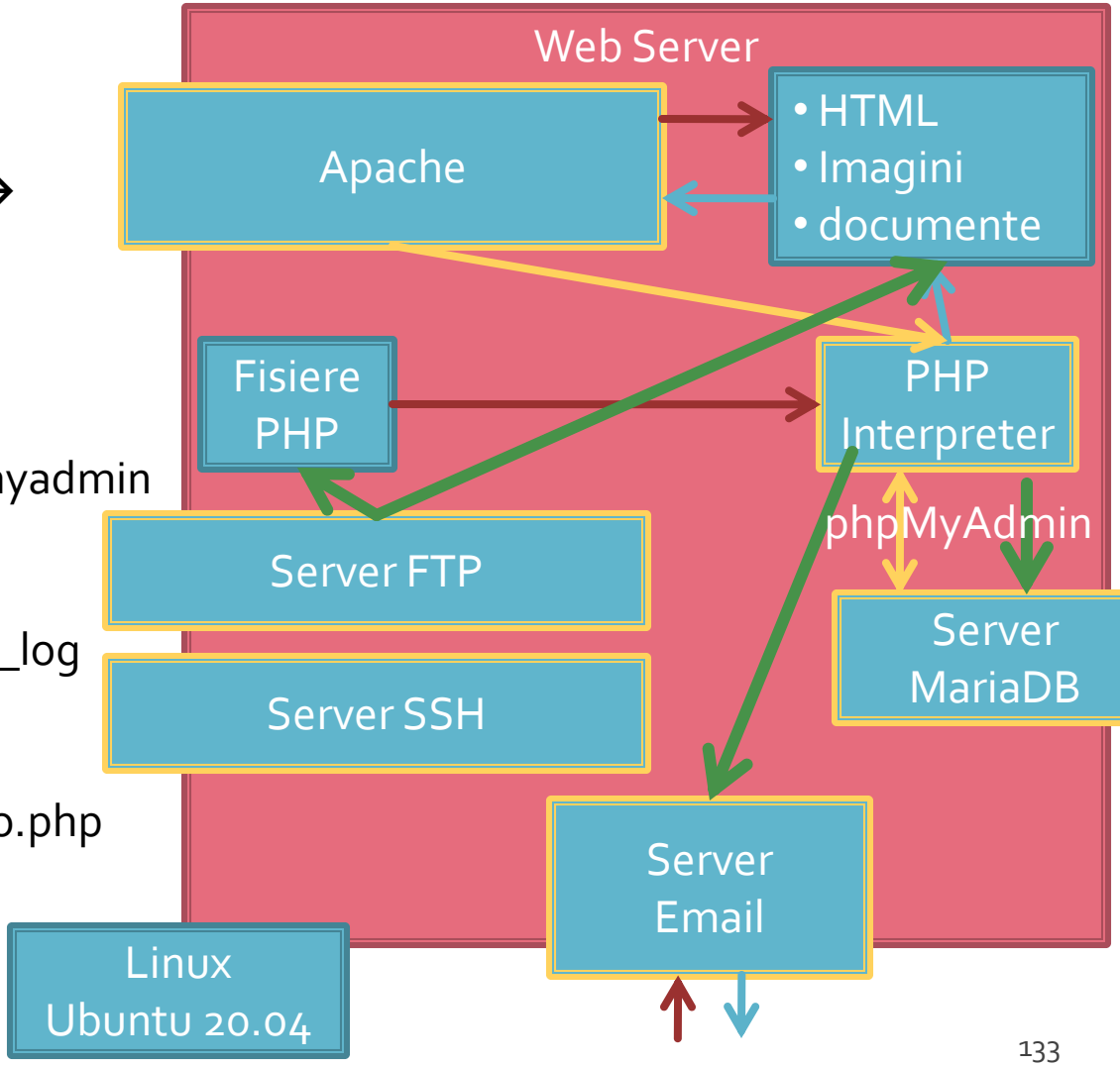
Faza de verificare/depanare

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

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

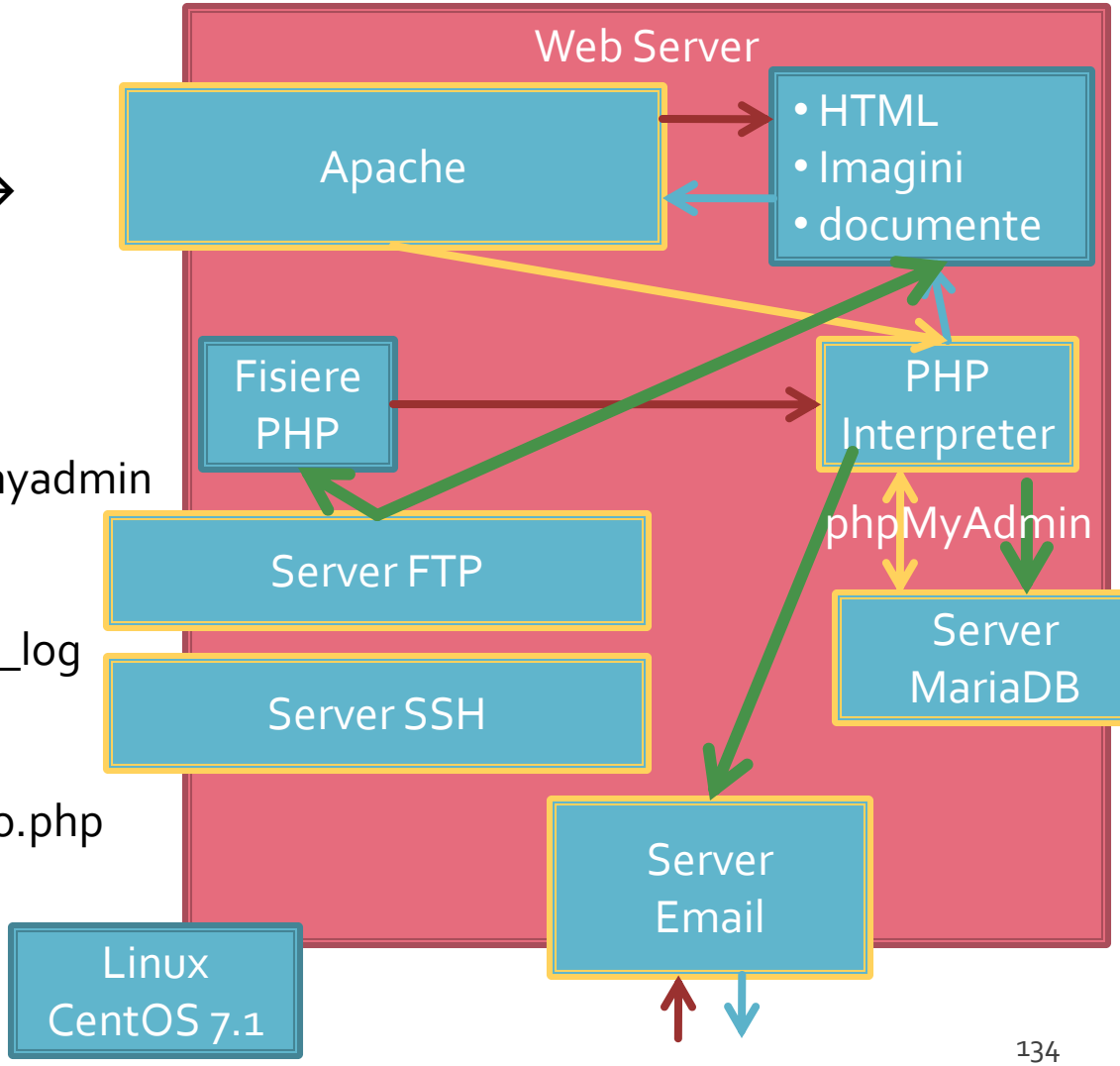
Utilizare LAMP

1. login → **paw**:masteretti
2. ifconfig → 192.168.30.5
3. putty.exe → 192.168.30.5 → SSH → **paw**:masteretti (remote login)
4. [alte comenzi linux dorite]
5. FTP → Winscp → SFTP → student:masterrc@192.168.30.5
6. MySql → http://192.168.30.5/phpmyadmin → **root**: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

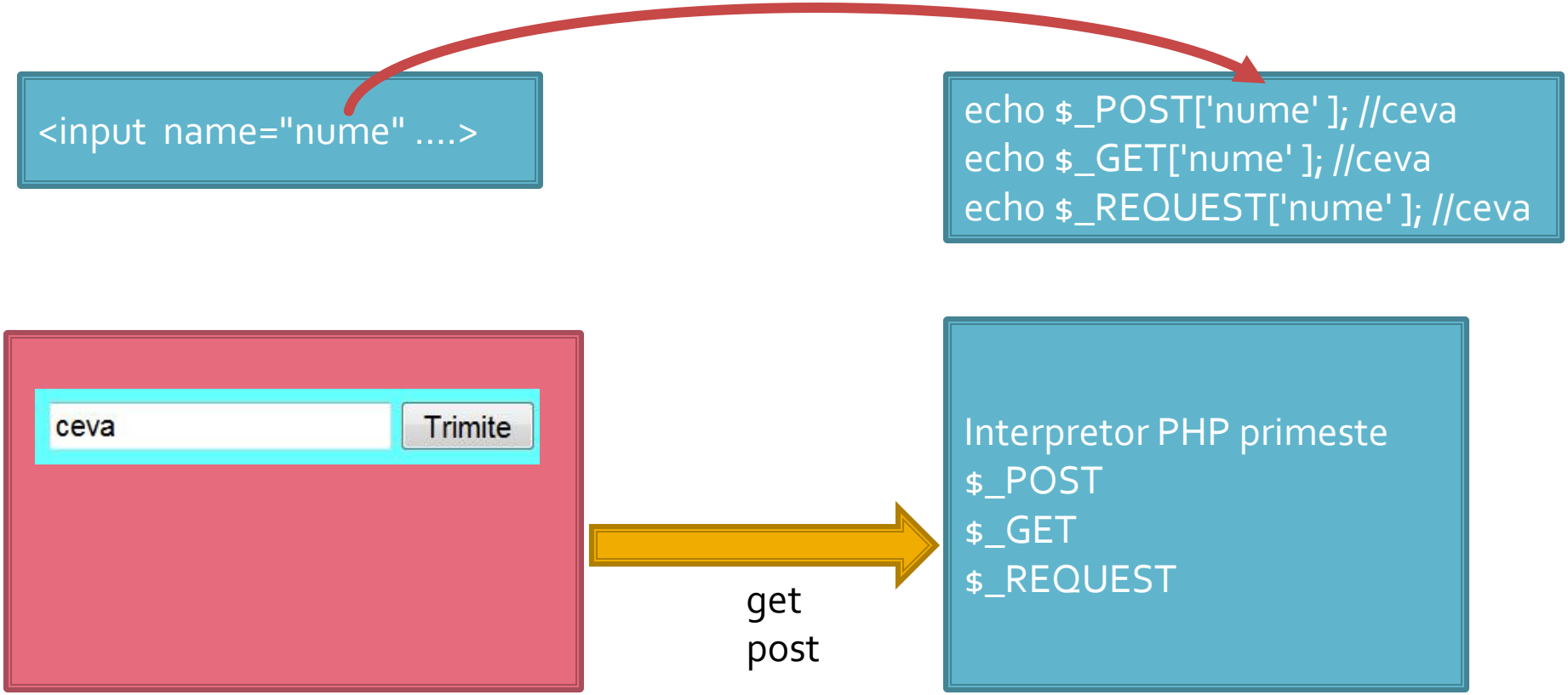


Utilizare LAMP

1. login → root:masterrc
2. ifconfig → 192.168.30.5
3. putty.exe → 192.168.30.5 → SSH → root:masterrc (remote login)
4. [alte comenzi linux dorite]
5. FTP → Winscp → SFTP → student:masterrc@192.168.30.5
6. MySql → http://192.168.30.5/phpmyadmin → root:masterrc
7. Apache Error Log →
7a. putty → nano /var/log/httpd/error_log
7b. http://192.168.30.5/logfile.php (nonstandard)
8. PHP info → http://192.168.30.5/info.php



Client / Server



Depanare

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

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

Contact

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