Curs 11

2013/2014

Tehnici moderne de proiectare a aplicatiilor web

Activitate suplimentara

Activitate suplimentara

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

Recompensa activitate suplimentara

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

Regulament recompensa

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

Indicatii examinare

Teme de proiect

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

PROIECT (final)

- Tema de nota 7 (>6)
  - Tema unica pentru fiecare student
- Tema de nota 8 (>6)
  - Conditiile de la tema de nota 8 **si in plus**
  - Necesitatea conlucrarii intre **2 studenti** cu doua teme "pereche"

PROIECT (final)

- Tema de nota 9 (>5)
  - Conditiile de la tema de nota 8 **si in plus**
  - Necesitatea conlucrarii intre **3 studenti** cu trei teme "pereche"
  - Tema se preda/trimite cu macar **1 zi** inaintea sustinerii ei
  - Baza de date cu care se lucreaza sa contina minim **50** de inregistrari in tabelul cel mai "voluminos".
- Tema de nota 10 (>5)
  - Conditiile de la tema de nota 9 **si in plus**
  - Baza de date cu care se lucreaza contine minim **300** de inregistrari in tabelul cel mai "voluminos"
  - Necesitatea investigarii posibilitatilor de **imbunatatire** a aplicatiei si adaugarii de functionalitate
  - nota individuala la proiect va depinde intr-o mica masura (in limita a 1p) de nota medie a colegilor din echipa

PROIECT (final)

- proiectul se sustine individual (oral si practic)
- grila de notare la proiect schimbata fata de anii precedenti
- fiecare membru al unei echipe (la temele de nota 9 si 10) trebuie sa sustina in aceeasi zi proiectul
- nota individuala la proiect va depinde intr-o mica masura (in limita a 1p) de nota medie a colegilor din echipa (numai la temele de 10 si 10+)
  - N-min(E)=1 -> -0 p
  - N-min(E)=2 -> -0.5 p
  - N-min(E)=3 -> -1 p

PROIECT (final)

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

PROIECT (final)

- Tema de nota 10+ (>5, in general offline)
  - Conditiile de la tema de nota 10 **si in plus**
  - Baza de date cu care se lucreaza contine minim **400** de inregistrari in tabelul cel mai "voluminos"
  - Tema care face apel la controlul **sesiunii** client/server
  - Necesitatea utilizarii **Javascript** in **aplicatie** (aplicatie libera dar cu efect tehnic nu estetic)
  - Forma paginii trebuie sa respecte cerintele "F shape pattern"
  - Facilitati in ceea ce priveste prezenta la laborator (**DACA** toate celelalte conditii sunt indeplinite – P = 66%, L = 0%, E = 33%)

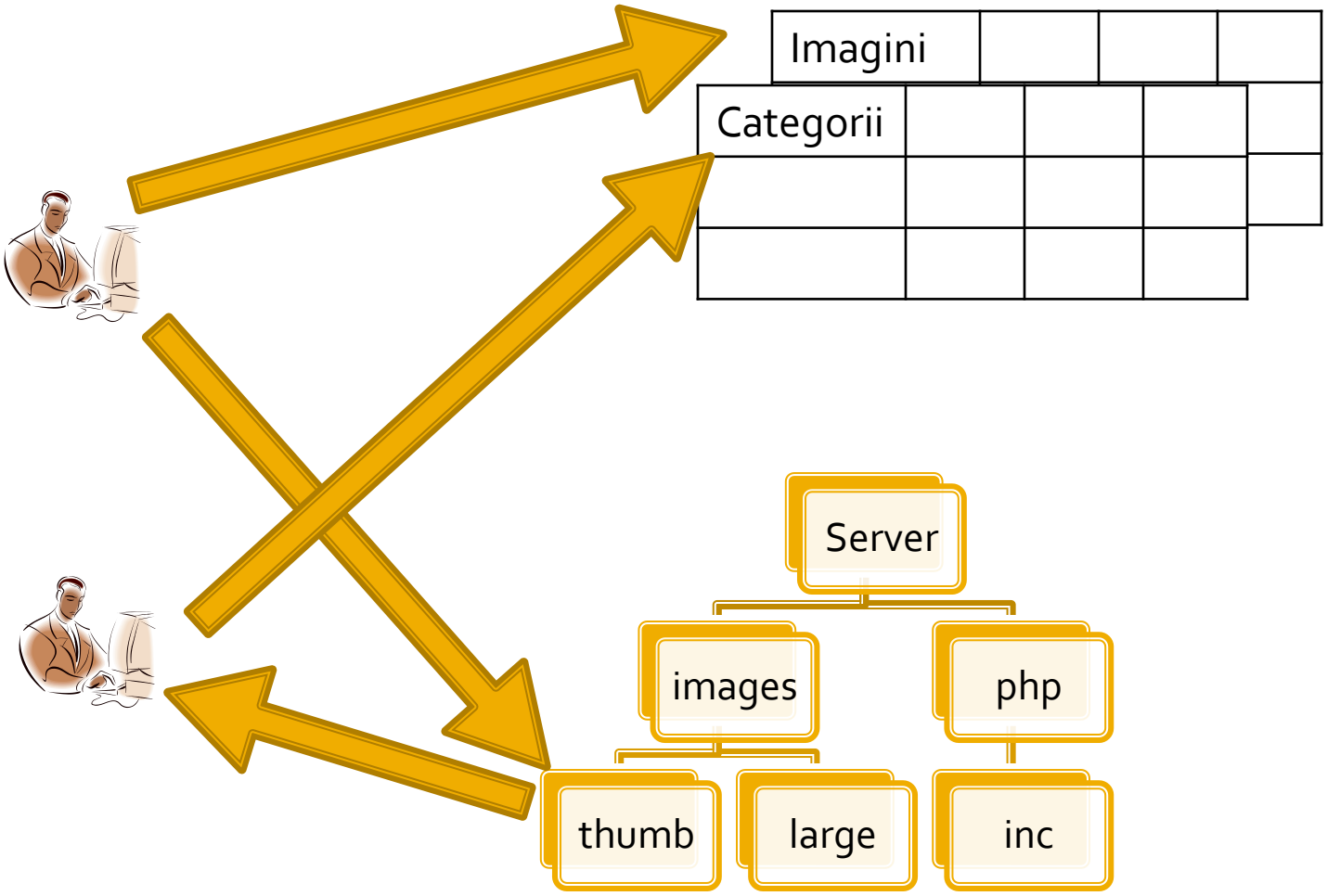
Exemplu

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

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

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

Exemplu



Teme de proiect

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

Notare

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

Examen

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

Examen

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

Limbaj SQL

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 (posibilitatea cautarii in intregul text, index FULLTEXT apare doar MySql 5.6 ->)
  - 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

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**

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
- Util pentru a vedea:
  - efectul normalizarii
  - efectul indecsilor
  - efectul utilizarii, peste tot unde e posibil, a valorilor numerice intregi
  - exemple de "traducere" din limbaj natural (uman) in SQL, a interogarilor

Laborator 2, BD nenormalizata

```
210
211 /*!40000 ALTER TABLE `inregistrare` DISABLE KEYS */;
212 INSERT INTO `inregistrare` (`ID_INREGISTRARE`,`Data`,`Data_Lucru`,`Lot`,`Brut`,`Net`,`Amb`,`Tip`,`Prod`,`Dest`) VALUES
213  (1,'2008-02-29 11:01:13','2008-02-29',40,95.2,85.2,'Lada','Refrigerat','Cap+Ghiare','Distributie SAFIR'),
214  (2,'2008-02-29 11:09:28','2008-02-29',40,110.8,103,'Lada','Refrigerat','Oase Pulpe Dezosate','AGRICOLA INTERNATIONAL'),
215  (3,'2008-02-29 11:15:45','2008-02-29',40,830.2,770.1,'Lada','Refrigerat','Pulpa cu OS','AGRICOLA INTERNATIONAL'),
216  (4,'2008-02-29 11:21:39','2008-02-29',40,34,30.1,'Punga','Refrigerat','Pulpa cu OS','Distributie SAFIR'),
217  (5,'2008-02-29 11:22:49','2008-02-29',40,8.4,5.8,'Punga','Refrigerat','Pulpa cu OS','Distributie SAFIR'),
218  (6,'2008-02-29 11:24:42','2008-02-29',40,829.8,772.6,'Lada','Congelat','MDM spinari','Distributie SAFIR'),
219  (7,'2008-02-29 11:29:32','2008-02-29',40,624.6,554.1,'Punga','Refrigerat','Pui GRILL','AGRICOLA INTERNATIONAL'),
220  (8,'2008-02-29 11:37:17','2008-02-29',40,1.6,1.1,'Punga','Refrigerat','Pulpe INFERIOARE','Distributie SAFIR'),
221  (9,'2008-02-29 11:46:19','2008-02-29',40,826.6,785,'Lada','Refrigerat','Piept cu OS','AGRICOLA INTERNATIONAL'),
222  (10,'2008-02-29 11:48:41','2008-02-29',40,463.8,406.3,'Lada','Refrigerat','Pui GRILL','AGRICOLA INTERNATIONAL'),
223  (11,'2008-02-29 12:02:09','2008-02-29',40,614,547.5,'Lada','Refrigerat','Pui GRILL','AUCHAN PITESTI GASTRO'),
224  (12,'2008-02-29 12:06:05','2008-02-29',40,26.8,22,'Tava','Refrigerat','Pulpa cu OS','AUSHAN TG. MURES CARMANGERIE'),
225  (13,'2008-02-29 12:07:54','2008-02-29',40,0,0,'Tava','Refrigerat','Pui CGP','AUSHAN TG. MURES CARMANGERIE'),
226  (14,'2008-02-29 12:11:06','2008-02-29',40,112,101.6,'Punga','Refrigerat','Pui CGP','AUSHAN TG. MURES CARMANGERIE'),
227  (15,'2008-02-29 12:12:03','2008-02-29',40,38.2,32.2,'Tava','Refrigerat','FICAT','AUCHAN TG. MURES GASTRO'),
228  (16,'2008-02-29 12:12:38','2008-02-29',40,32.2,27.4,'Punga','Refrigerat','PIPOTA si INIMI','AUSHAN TG. MURES CARMANGERIE'),
229  (17,'2008-02-29 12:17:01','2008-02-29',1,21.8,19.2,'Tava','Refrigerat','Pulpe SUPERIOARE','Distributie SAFIR'),
230  (18,'2008-02-29 12:17:02','2008-02-29',1,21.8,19.2,'Tava','Refrigerat','Pulpe SUPERIOARE','Distributie SAFIR'),
231  (19,'2008-02-29 12:18:04','2008-02-29',1,18,15.6,'Tava','Refrigerat','FICAT','Distributie SAFIR'),
232  (20,'2008-02-29 12:19:21','2008-02-29',1,24.4,20.8,'Tava','Refrigerat','Pulpa cu OS','Distributie SAFIR'),
233  (21,'2008-02-29 12:21:07','2008-02-29',1,14.4,10.9,'Tava','Refrigerat','ARIPI','Distributie SAFIR'),
234  (22,'2008-02-29 12:22:04','2008-02-29',1,24.8,21.3,'Tava','Refrigerat','Piept cu OS','Distributie SAFIR'),
235  (23,'2008-02-29 12:22:56','2008-02-29',1,34,30.1,'Tava','Refrigerat','Piept cu OS','Distributie SAFIR'),
236  (24,'2008-02-29 12:23:58','2008-02-29',1,35.2,30.4,'Tava','Refrigerat','Pulpe SUPERIOARE','Distributie SAFIR'),
237  (25,'2008-02-29 12:24:39','2008-02-29',1,20.4,16.7,'Tava','Refrigerat','FICAT','Distributie SAFIR'),
238  (26,'2008-02-29 12:25:46','2008-02-29',1,42.2,37,'Punga','Refrigerat','Pui CGP','Distributie SAFIR'),
239  (27,'2008-02-29 12:25:46','2008-02-29',1,42.2,37,'Punga','Refrigerat','Pui CGP','Distributie SAFIR'),
240  (28,'2008-02-29 12:25:46','2008-02-29',1,42.2,37,'Punga','Refrigerat','Pui CGP','Distributie SAFIR'),
241  (29,'2008-02-29 12:26:43','2008-02-29',1,31.2,27.3,'Punga','Refrigerat','Pui CGP','Distributie SAFIR'),
242  (30,'2008-02-29 12:27:55','2008-02-29',1,272.2,253.1,'Lada','Congelat','CT Aripi','Distributie SAFIR'),
243  (31,'2008-02-29 12:29:14','2008-02-29',1,17.6,13.7,'Lada','Congelat','MDM piept','Distributie SAFIR'),
```

Laborator 2, BD normalizata

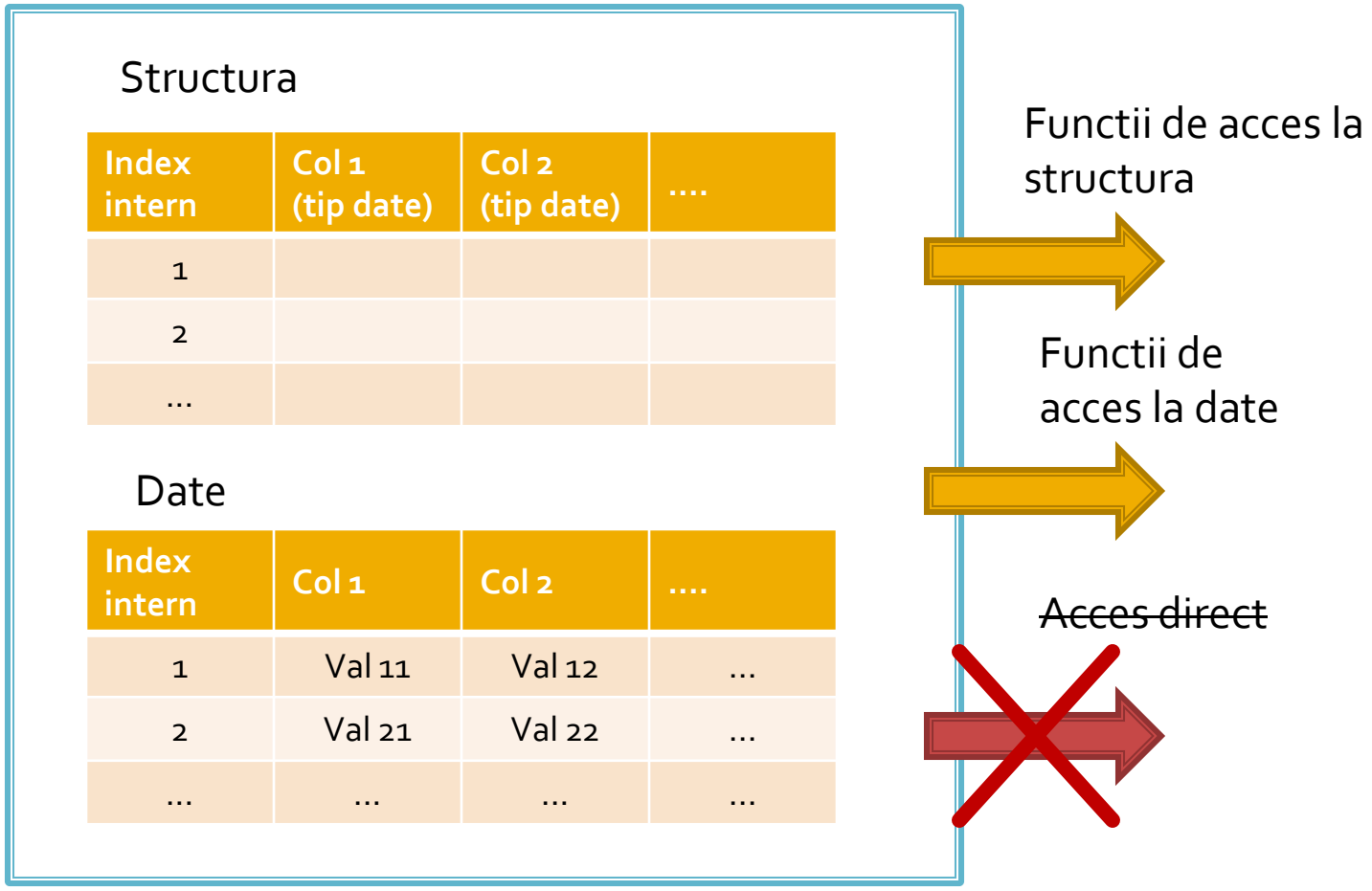
```
--

/*!40000 ALTER TABLE `inregistrare` DISABLE KEYS */;
INSERT INTO `inregistrare` (`ID_INREGISTRARE`,`Data`,`Data_Lucru`,`Lot`,`Ambalaj`,`Tip_Produs`,`Produs`,`Destinatie`,`Brut`,`Net`,
 (1,'2008-02-29 11:01:13','2008-02-29',40,3,1,29,34,95.2,85.2,6,0),
 (2,'2008-02-29 11:09:28','2008-02-29',40,3,1,53,23,110.8,103,6,43),
 (3,'2008-02-29 11:15:45','2008-02-29',40,3,1,8,23,830.2,770.1,6,0),
 (4,'2008-02-29 11:21:39','2008-02-29',40,1,1,8,34,34,30.1,6,0),
 (5,'2008-02-29 11:22:49','2008-02-29',40,1,1,8,34,8.4,5.8,6,0),
 (6,'2008-02-29 11:24:42','2008-02-29',40,3,2,35,34,829.8,772.6,6,0),
 (7,'2008-02-29 11:29:32','2008-02-29',40,1,1,2,23,624.6,554.1,6,0),
 (8,'2008-02-29 11:37:17','2008-02-29',40,1,1,9,34,1.6,1.1,6,0),
 (9,'2008-02-29 11:46:19','2008-02-29',40,3,1,15,23,826.6,785,6,0),
 (10,'2008-02-29 11:48:41','2008-02-29',40,3,1,2,23,463.8,406.3,6,0),
 (11,'2008-02-29 12:02:09','2008-02-29',40,3,1,2,11,614,547.5,6,0),
 (12,'2008-02-29 12:06:05','2008-02-29',40,2,1,8,12,26.8,22,6,0),
 (13,'2008-02-29 12:07:54','2008-02-29',40,2,1,1,12,0,0,6,0),
 (14,'2008-02-29 12:11:06','2008-02-29',40,1,1,1,12,112,101.6,6,0),
 (15,'2008-02-29 12:12:03','2008-02-29',40,2,1,6,13,38.2,32.2,6,0),
 (16,'2008-02-29 12:12:38','2008-02-29',40,1,1,7,12,32.2,27.4,6,0),
 (17,'2008-02-29 12:17:01','2008-02-29',1,2,1,10,34,21.8,19.2,6,0),
 (18,'2008-02-29 12:17:02','2008-02-29',1,2,1,10,34,21.8,19.2,6,0),
 (19,'2008-02-29 12:18:04','2008-02-29',1,2,1,6,34,18,15.6,6,0),
 (20,'2008-02-29 12:19:21','2008-02-29',1,2,1,8,34,24.4,20.8,6,0),
 (21,'2008-02-29 12:21:07','2008-02-29',1,2,1,20,34,14.4,10.9,6,0),
 (22,'2008-02-29 12:22:04','2008-02-29',1,2,1,15,34,24.8,21.3,6,0),
 (23,'2008-02-29 12:22:56','2008-02-29',1,2,1,15,34,34,30.1,6,0),
 (24,'2008-02-29 12:23:58','2008-02-29',1,2,1,10,34,35.2,30.4,6,0),
 (25,'2008-02-29 12:24:39','2008-02-29',1,2,1,6,34,20.4,16.7,6,0),
 (26,'2008-02-29 12:25:46','2008-02-29',1,1,1,1,34,42.2,37,6,0),
 (27,'2008-02-29 12:25:46','2008-02-29',1,1,1,1,34,42.2,37,6,0),
 (28,'2008-02-29 12:25:46','2008-02-29',1,1,1,1,34,42.2,37,6,0),
 (29,'2008-02-29 12:26:43','2008-02-29',1,1,1,1,34,31.2,27.3,6,0),
 (30,'2008-02-29 12:27:55','2008-02-29',1,3,2,42,34,272.2,253.1,6,0),
```

Laborator 2, interogari

```
    FROM `inregistrare` AS inr
    LEFT JOIN `destinatie` AS d ON (inr.`Destinatie`=d.`ID_DESTINATIE`)
    WHERE inr.`Data_Lucru` BETWEEN '2008-03-10' AND '2008-03-18'
    GROUP BY inr.`Ambalaj` ASC,d.`Beneficiar` ASC,inr.`Tip_Produs` ASC WITH ROLLUP

SELECT inr.`Ambalaj`,d.`Beneficiar`,inr.`Tip_Produs`,ROUND(SUM(inr.`Net`),2) AS Val
    FROM `inregistrare` AS inr
    LEFT JOIN `destinatie` AS d ON (inr.`Destinatie`=d.`ID_DESTINATIE`)
    WHERE inr.`Lot` = 52
    GROUP BY inr.`Ambalaj` ASC,d.`Beneficiar` ASC,inr.`Tip_Produs` ASC WITH ROLLUP

SELECT d.`Beneficiar`,inr.`Tip_Produs`,ROUND(SUM(inr.`Net`),2) AS Val
    FROM `inregistrare` AS inr
    LEFT JOIN `destinatie` AS d ON (inr.`Destinatie`=d.`ID_DESTINATIE`)
    WHERE inr.`Lot` = 52
    GROUP BY d.`Beneficiar` ASC,inr.`Tip_Produs` ASC WITH ROLLUP

SELECT inr.`Ambalaj`,inr.`Produs`,p.`Nume`,d.`Beneficiar`,inr.`Tip_Produs`,ROUND(SUM(inr.`Net`),2) AS Val
    FROM `inregistrare` AS inr
    LEFT JOIN `produs` AS p ON (inr.`Produs`=p.`ID_PRODUS`)
    LEFT JOIN `destinatie` AS d ON (inr.`Destinatie`=d.`ID_DESTINATIE`)
    WHERE inr.`Lot` = 52
    GROUP BY inr.`Ambalaj` ASC,inr.`Produs` ASC,d.`Beneficiar` ASC,inr.`Tip_Produs` ASC

SELECT `Ambalaj`,`Tip_Produs`,ROUND(SUM(`Net`),2) AS Val
    FROM `inregistrare` WHERE `Destinatie` = 100 AND `Data_Lucru` = '2008-03-11'
    GROUP BY `Ambalaj` ASC,`Tip_Produs` ASC WITH ROLLUP;


SELECT inr.`Ambalaj`,inr.`Produs`,p.`Nume`,inr.`Tip_Produs`,ROUND(SUM(inr.`Net`),2) AS Val
    FROM `inregistrare` AS inr
LEFT JOIN `produs` AS p ON (inr.`Produs`=p.`ID_PRODUS`)
    WHERE inr.`Destinatie` = 100 AND inr.`Data_Lucru` = '2008-03-11'
    GROUP BY inr.`Ambalaj` ASC,inr.`Produs` ASC,inr.`Tip_Produs` ASC;
```

Resurse MySQL

Resurse MySQL



Resurse MySQL

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

Resurse MySQL

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

Optimizare PHP/SQL

- Se tine cont de faptul ca accesul la date se va face secvential
- Ca urmare algoritmul PHP trebuie adaptat pentru a lucra secvential cu datele ("on the fly")
- Este **Nerecomandata** metoda de a utiliza un cod PHP pentru a culege **toate** datele urmata de o sectiune de prelucrare a datelor integrale
- Interogarile SQL se modifica/optimizeaza pentru a oferi datele dorite in **EXACT** ordinea necesara pentru prelucrare (in general doar afisare)

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 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/">
- 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 (slide-ul urmator)

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 ( )

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

- 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

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

```
<?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
// 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>' ;
?>
```

MySql

Mini – Indrumar practic
Lucru cu bazele de date

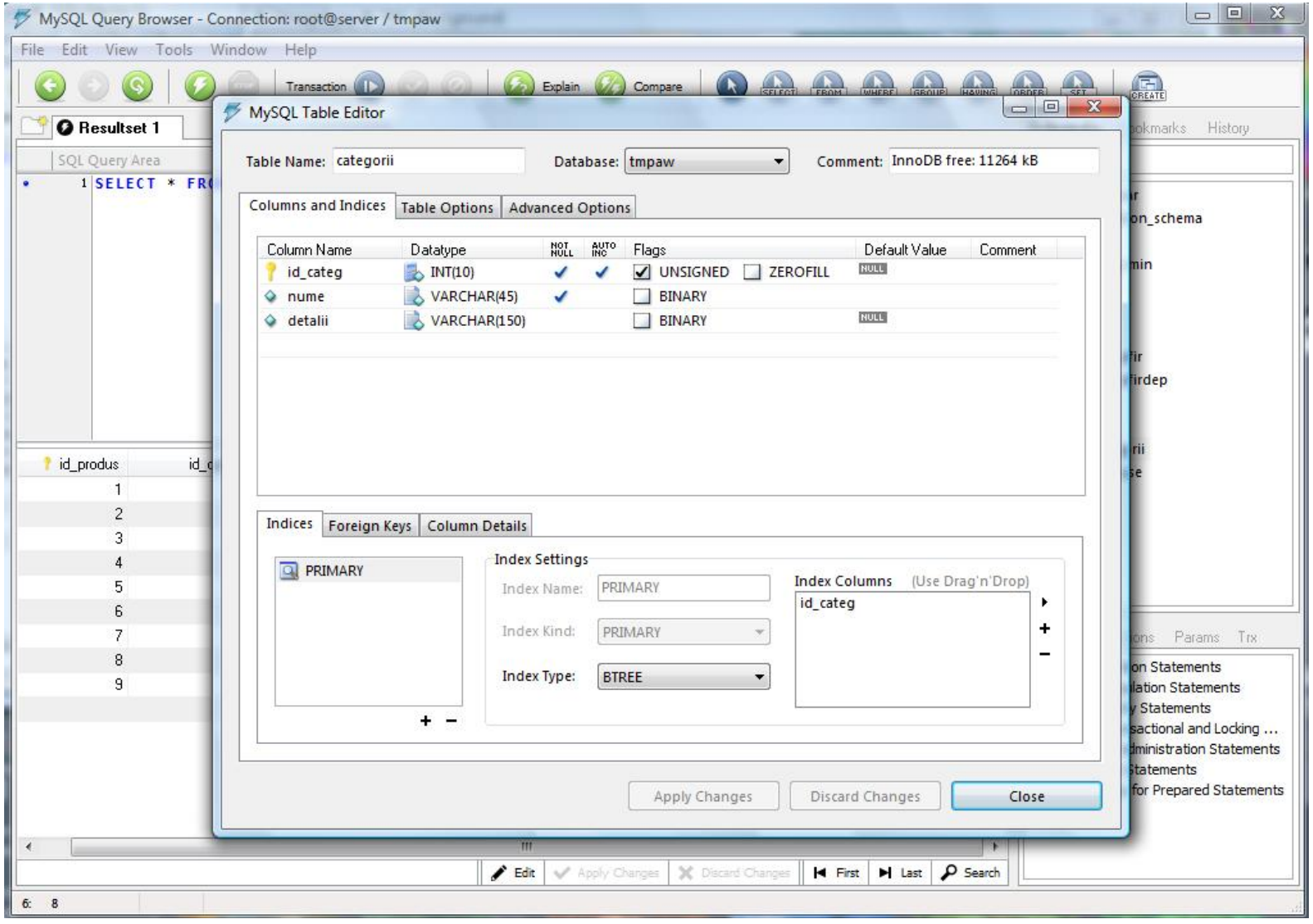
Realizarea bazei de date

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

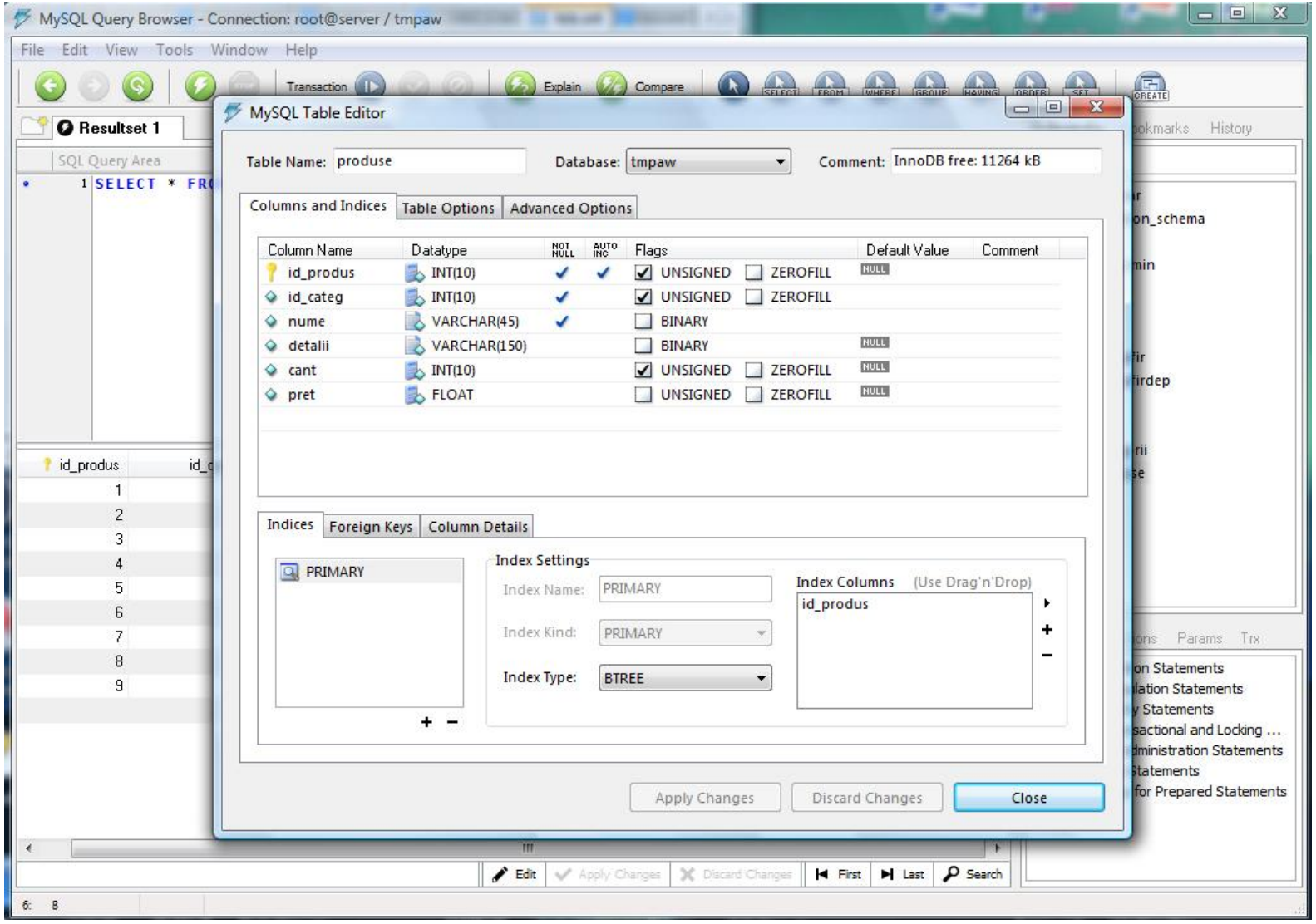
Introducere tabele

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

Tabel Categorii



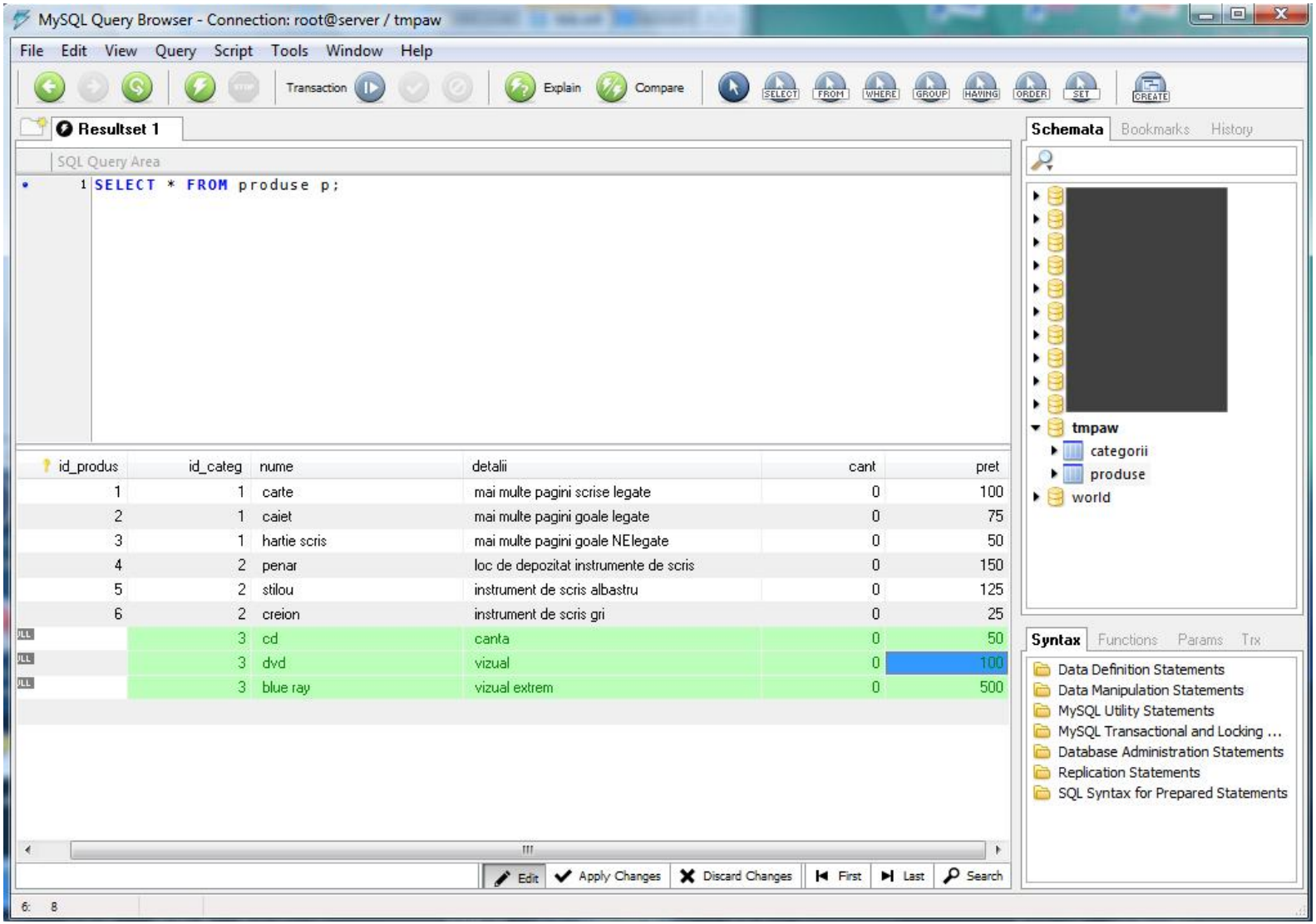
Tabel Produse



Introducere date initiale

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

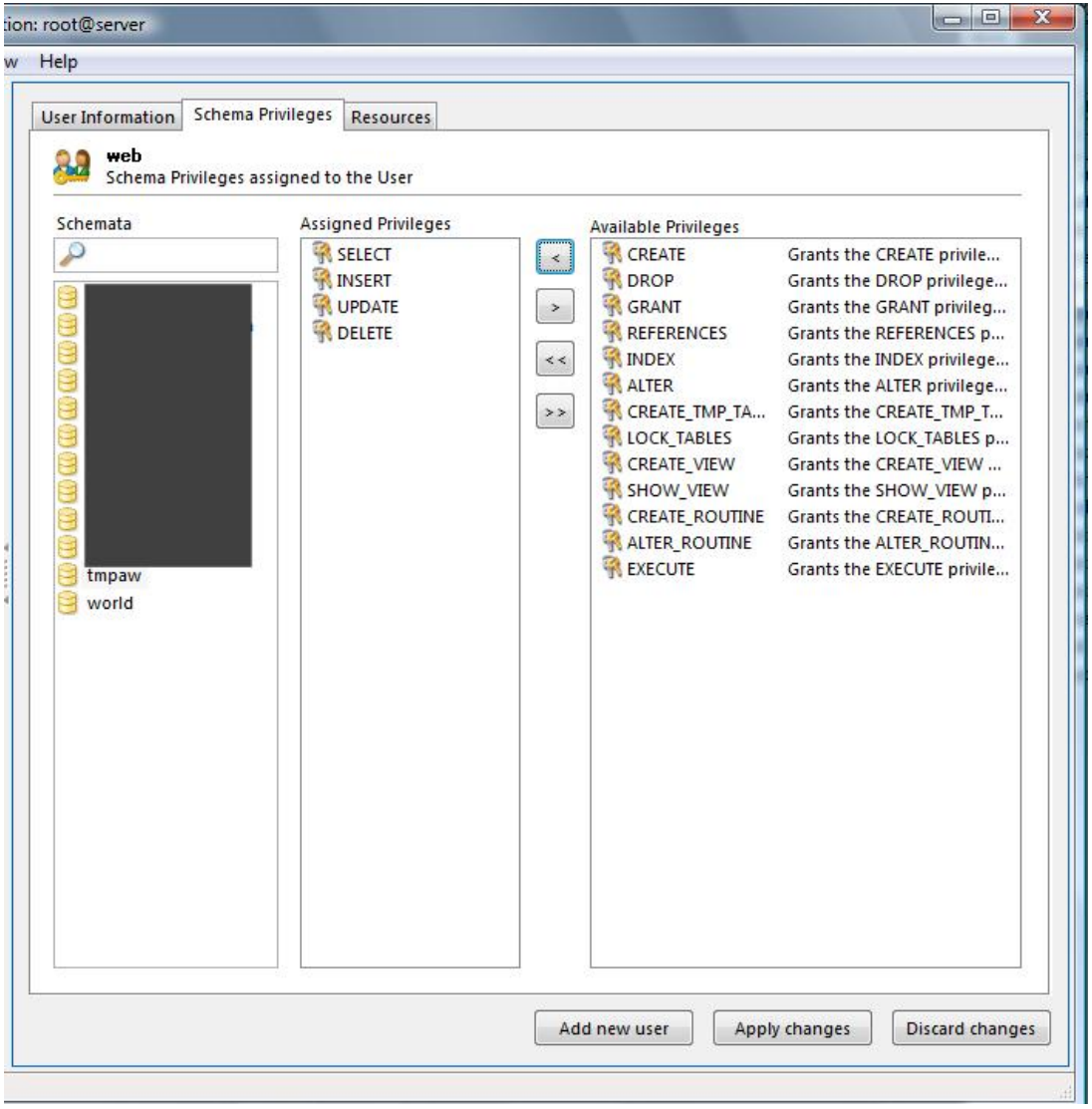
Introducere date initiale



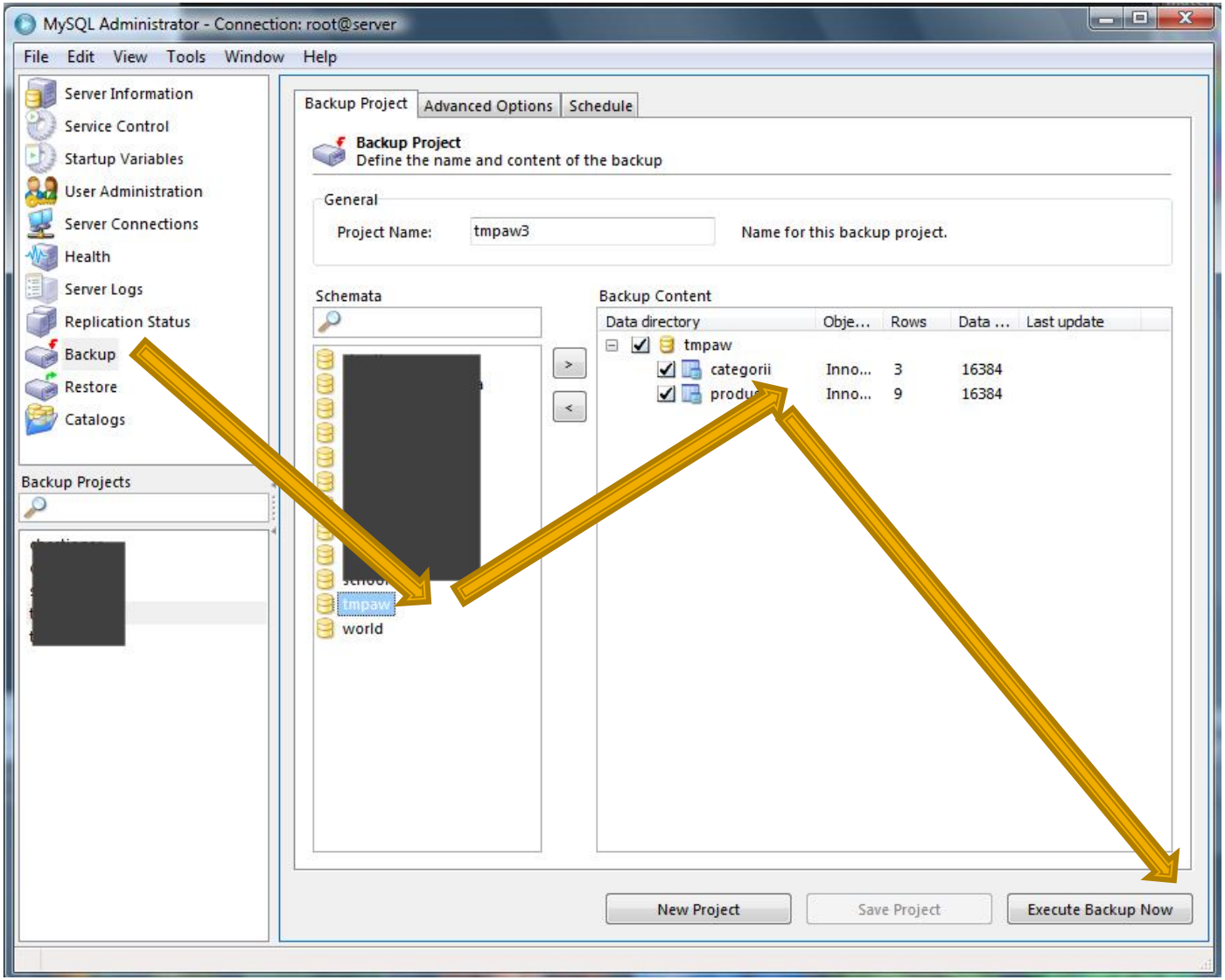
Backup, Restore, drepturi de acces

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

Drepturi de acces



Backup


MySQL Administrator - Connection: root@server
File Edit View Tools Window Help
Server Information
Service Control
Startup Variables
User Administration
Server Connections
Health
Server Logs
Replication Status
Backup
Restore
Catalogs
Backup Projects
Backup Project
Advanced Options
Schedule
Backup Project
Define the name and content of the backup
General
Project Name:
tmpaw3
Name for this backup project.
Schemata
Backup Content
Data directory
Obje...
Rows
Data ...
Last update
tmpaw
categorii
Inno...
3
16384
Inno...
9
16384
tmpaw
world
New Project
Save Project
Execute Backup Now

Restaurarea bazei de date

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

Script SQL Backup - utilitate

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

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

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

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

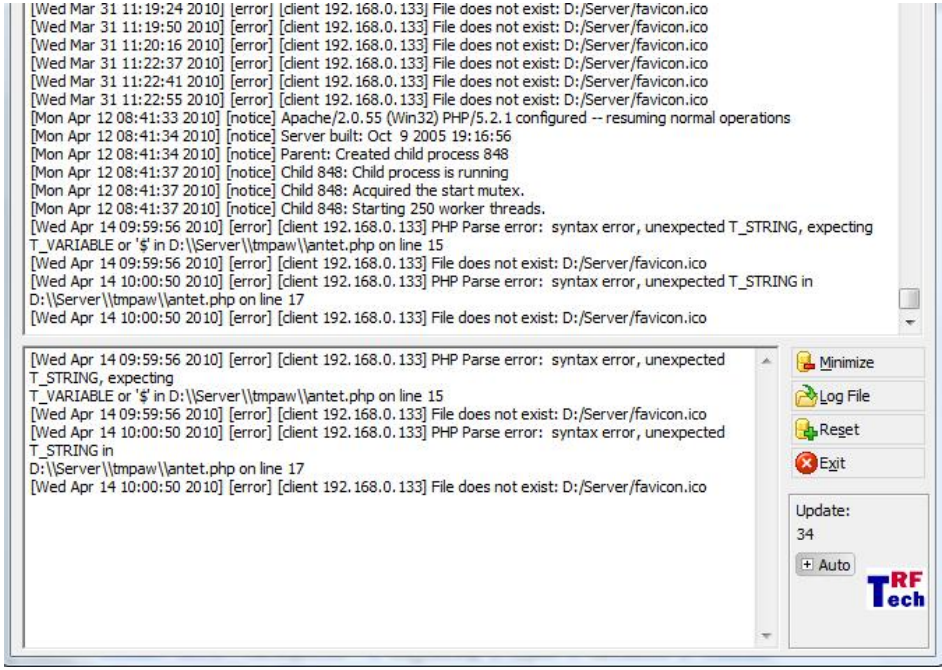
Aspecte practice recomandate in realizarea aplicatiilor web

Metode de lucru recomandate 1

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

Metode de lucru recomandate 2

- Verificarea "log-ului" de erori al server-ului Apache ramane principala metoda de depanare a codului PHP. Utilizarea aplicatiei prezentata la laborator este mai comoda datorita automatizarii dar orice alta varianta este utila



Metode de lucru recomandate 3

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

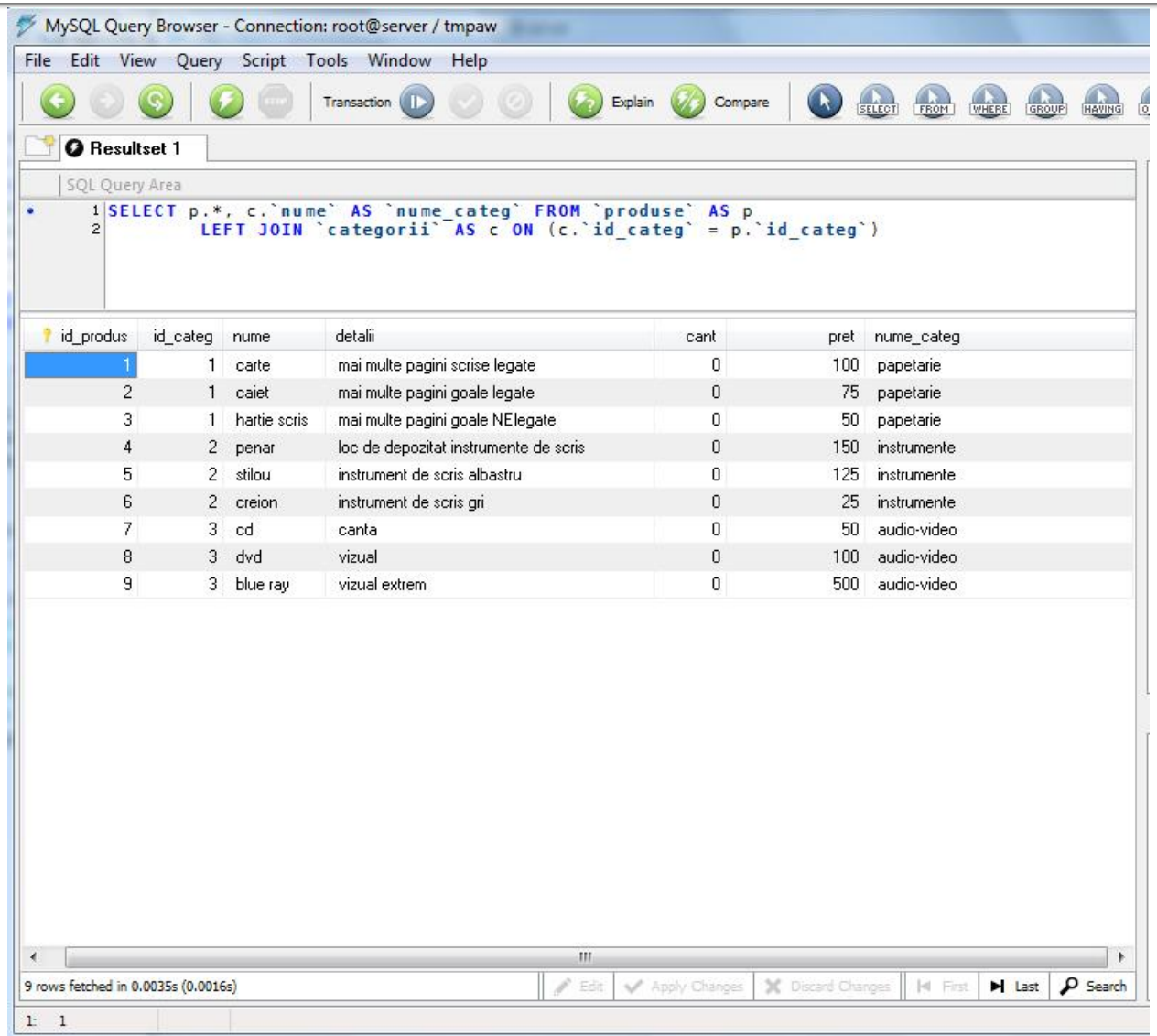
Resultset 1

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

Metode de lucru recomandate 3



Metode de lucru recomandate 4

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

Metode de lucru recomandate 5

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

Metode de lucru recomandate 5

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

Metode de lucru recomandate 5

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

MySql – eficienta

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

Contact

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