Programare Web

Acces la baze de date din PHP

Ciprian Dobre
ciprian.dobre@cs.pub.ro

Obiectiv

- Prezentarea modului în care se poate face accesul la baze de date din PHP.
- Vor fi prezentate funcțiile de acces pentru MySQL și Oracle.
  - Funcțiile ODBC sunt similare celor din MySQL și Oracle.

Caracteristici

- Pachetul PHP pune la dispoziție funcții pentru accesul la o multitudine de SGBD-uri și pachete de gestiune a datelor.
- Din păcate aceste funcții nu sunt standardizate ci de obicei copiază funcții existente în API-urile pachetelor respective.
- Din această cauză pentru fiecare tip de SGBD există un set diferit de funcții care permit accesul la datele din baza de date.

Caracteristici

- Pașii în cazul regăsirii datelor sunt în general următorii:

1. Stabilirea unei conexiuni cu serverul de baze de date (cine se conectează?)
2. Trimiterea spre execuție a cererii de tip SELECT
3. Preluarea linie cu linie a datelor din tabela rezultat și prelucrarea acestora
4. Eliberarea resurselor ocupate de rezultat
5. Închiderea conexiunii, după execuția repetata a pașilor 2-4

Caracteristici

- În cazul execuției altor tipuri de cereri (insert, update, delete, cereri DDL, etc) se execută pașii 1, 2 și 5, neexistând o tabelă rezultat.
- În toate cazurile, la apariția unei erori se pot obține informații despre ea folosind funcții care întorc textul mesajului de eroare.

Funcții MySQL - conectare

1. Conectare: Pentru conectarea la SGBD există mai multe funcții care pot fi folosite alternativ:

resource mysql_connect(string [hostname] [:port] , string [username] , string [password], bool [new_link], int [client_flags]);

- Întoarce un identificator de conexiune (numeric, pozitiv) în caz de succes și *false* în caz de eșec.
- Toate argumentele sunt opționale, valorile implicite fiind *localhost:3306*, numele utilizatorului care deține procesul server și un șir vid pentru parolă.

Funcții MySQL - conectare

- *new_link* forțează returnarea unui nou descriptor de conexiune chiar dacă există deja o conexiune (implicit nu) cu acel server iar client_flags specifică o serie de alte setări (implicit e ignorat)
- În cazul unui apel cu aceleași argumente cu ale unei conexiuni deja deschise va întoarce identificatorul acesteia.
- Pot exista simultan mai multe conexiuni deschise cu unul sau mai multe servere. Conexiunile se închid cu mysql_close() sau automat la terminarea scriptului.
- Exemplu:

```
$conexiune = mysql_connect("localhost");
```

Funcții MySQL - conectare

resource mysql_pconnect(string [hostname] [:port], string [username] , string [password]], int [client_flags]);

- Întoarce un identificator de conexiune persistentă sau *false* în caz de eșec.
- Se aseamănă cu mysql_connect() cu deosebirea că o astfel de conexiune nu se închide la terminarea scriptului și nici la execuția mysql_close().
- Conexiunile persistente rămân deschise urmând să fie returnate ca rezultat când se încearcă deschiderea de noi conexiuni cu aceeași parametri.

Functii MySQL- ce BD

2. Specificarea bazei de date:

```
bool mysql_select_db(string database_name, resource
   [id_conexiune] );
```

- Specifică numele bazei de date care va fi exploatată prin acea conexiune (serverul MySQL poate gestiona mai multe baze de date).
- Întoarce *true* pentru succes și *false* pentru eșec.
- Toate cererile care vor fi trimise în viitor cu mysql_query() se vor executa în baza de date specificată.
- Exemplu:

```
mysql_select_db("test_studenti", $conexiune );
```

Funcții MySQL - cereri

1. Execuția unei cereri SQL:

   resource mysql_query(string cerere, resource [id_conexiune]);

- Efectul este trimiterea cererii către server pentru a fi procesată.
- Întoarce: *true*/*resursa* în caz de reușită, *false* în caz de eșec.
- În cazul în care cererea a fost un SELECT rezultatul întors este o resursa = identificator de rezultat.
- Spațiul ocupat de un rezultat poate fi eliberat ulterior cu mysql_free_result().
- Exemplu:

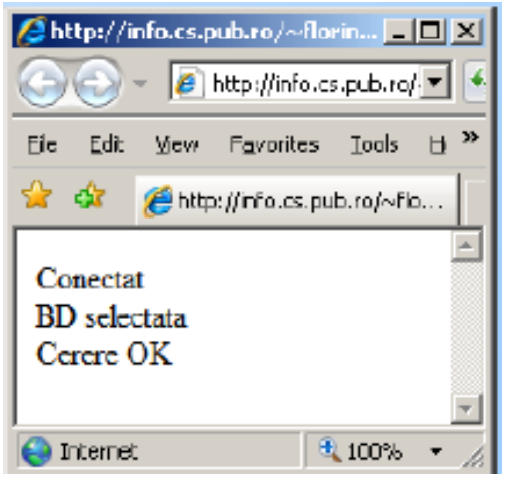
```
$rezultat = mysql_query("select * from studenti", $conexiune );
```

Funcții MySQL - cereri

- Observație: în afară de funcția mysql_connect toate celelalte funcții mysql care utilizează un descriptor de conexiune folosesc în cazul lipsei acestui parametru ultima conexiune deschisă.
- Deci dacă într-o pagină nu se folosește decât o singură conexiune MySQL atunci nu este necesară stocarea și folosirea ei explicită în funcțiile utilizate.
- Iată un exemplu:

Funcții MySQL - cereri

```
<?php
mysql_connect("localhost", "stud", "studpw") or die('Conexiune
    esuata');
echo "Conectat<br>";
// nu folosim descriptor de conexiune:
mysql_select_db("stud") or die ('Selectie BD esuata');
echo " BD selectata<br>";
// si nici aici:
mysql_query("select * from produse") or
die('cerere esuata');
echo " Cerere OK<br>";
?>
```



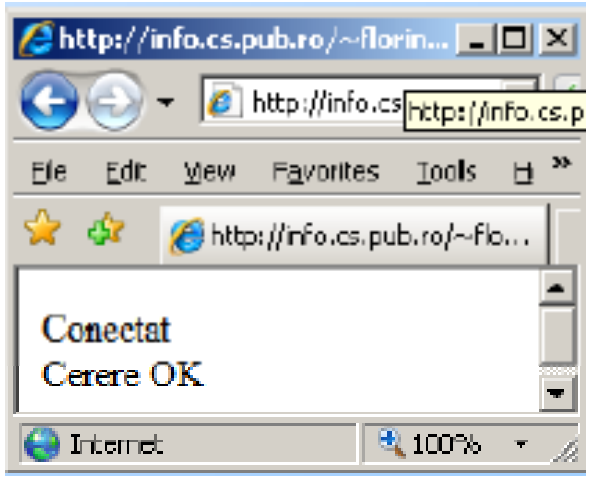
Funcții MySQL - cereri

- Există și funcția mysql_db_query:

resource mysql_db_query(string database, string cerere, resource [id_conexiune]);

- Folosirea acestei funcții nu este recomandată de situl php (*deprecated*).
- Efectul ei este cel cumulat al mysql_select_db și mysql_query.
- Exemplu:

```
<?php
mysql_connect("localhost", "stud", "studpw") or die('Conexiune esuata');
echo "Conectat<br>";
mysql_db_query("stud",
"select * from produse")
or die('Cerere esuata');
echo " Cerere OK<br>";
?>
```



Funcții MySQL - erori

- Detectarea erorii se face cu funcțiile:

```
int mysql_errno(resource [id_conexiune] );
```

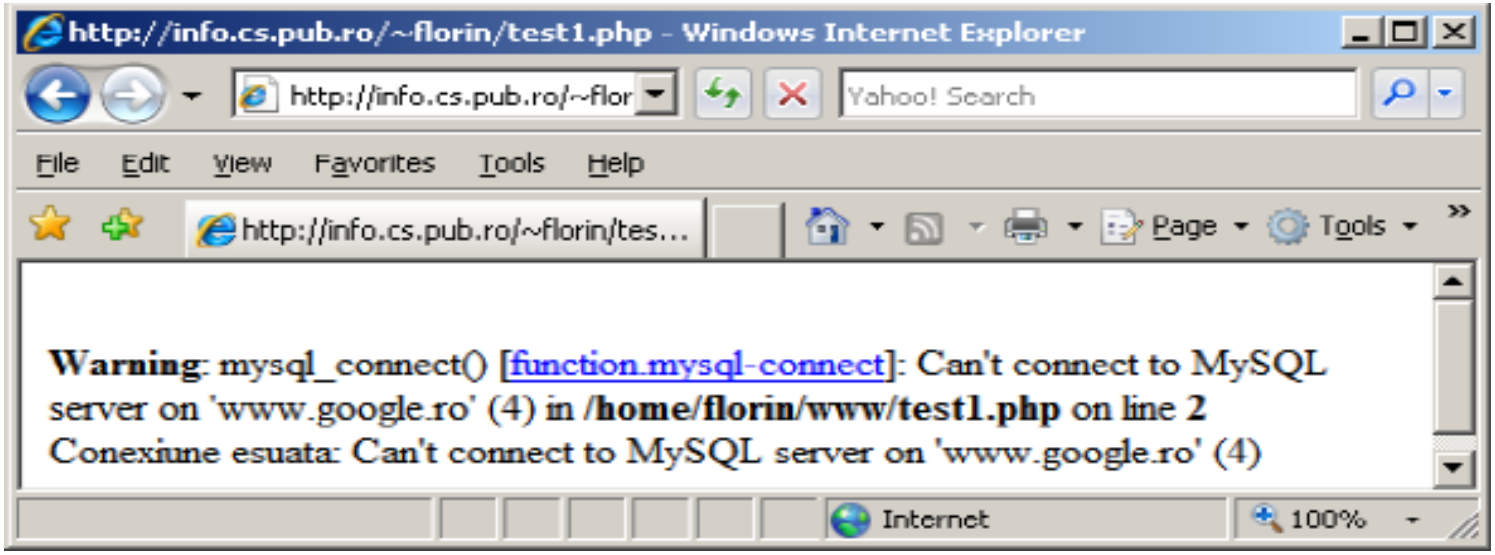
- Întoarce codul de eroare pentru operația precedentă pe acea conexiune (implicit ultima deschisă, ca mai înainte).

```
string mysql_error(resource [id_conexiune] );
```

- Întoarce textul mesajului de eroare pentru operația precedentă pe acea conexiune.

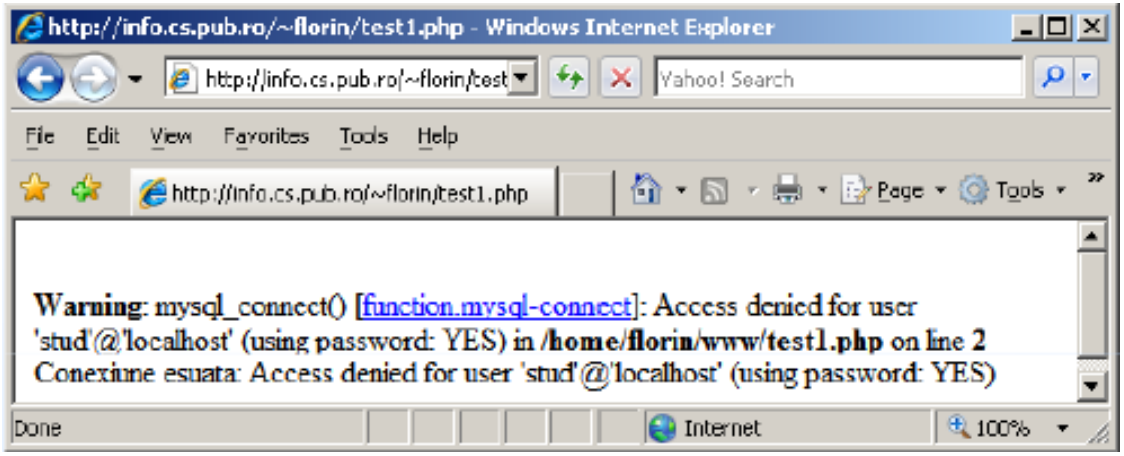
Funcții MySQL - erori

- Aceste erori sunt returnate de serverul MySQL.
- Excepție face încercarea unei conexiuni cu un host unde nu exista server MySQL.
- În acest caz mesajul va fi ca mai jos:

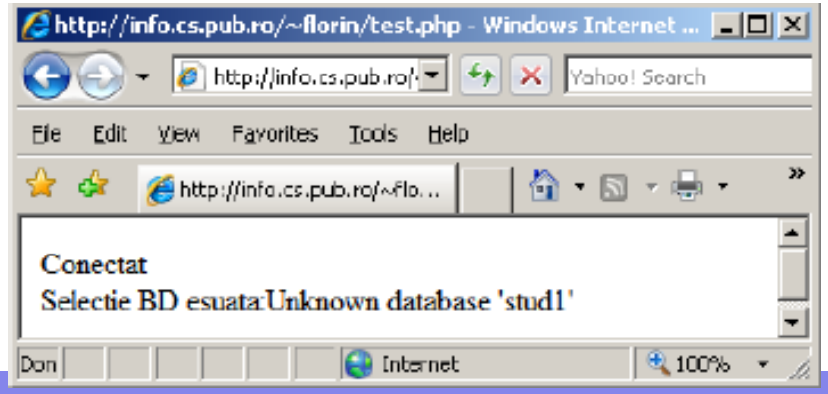


Exemple

- Parola greșită la conectare:



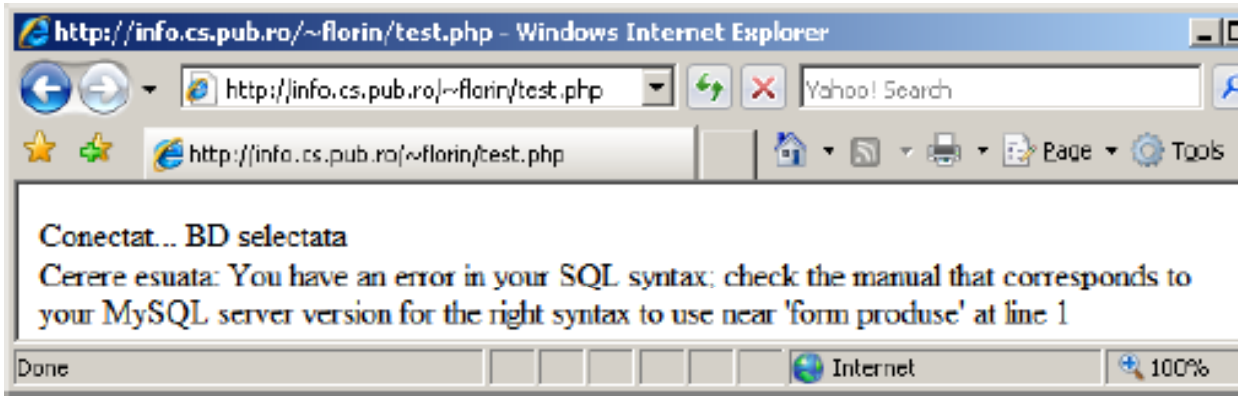
- Nume bază de date eronat:



Exemple

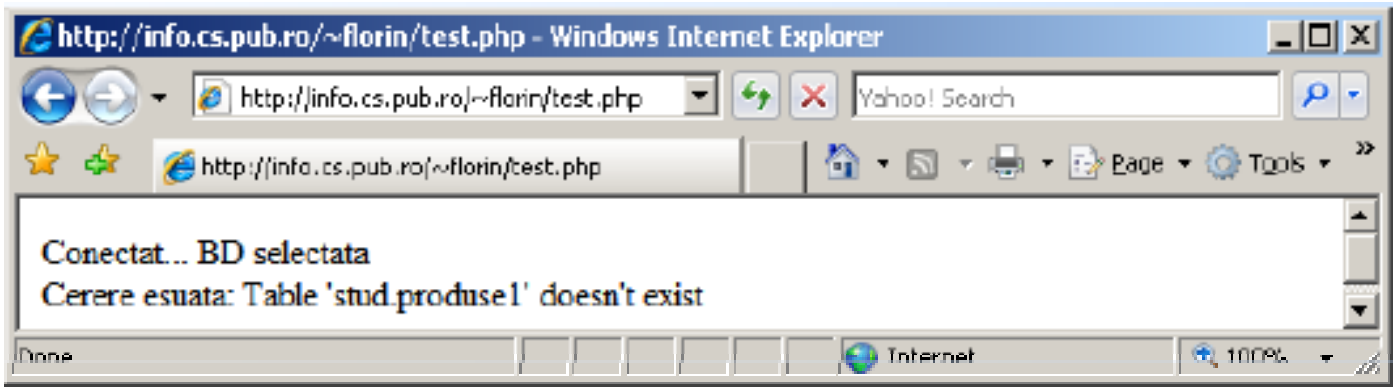
- Eroare de sintaxa in cerere:

```
<?php
mysql_connect("localhost", "stud", "studpw") or die('Conexiune
    esuata');
echo "Conectat...";
mysql_select_db("stud") or die ('Selectie BD esuata:
    '.mysql_error());
echo " BD selectata<br>";
mysql_query("select * form produse") or die('Cerere esuata:
    '.mysql_error());
echo " Cerere OK<br>";
?>
```

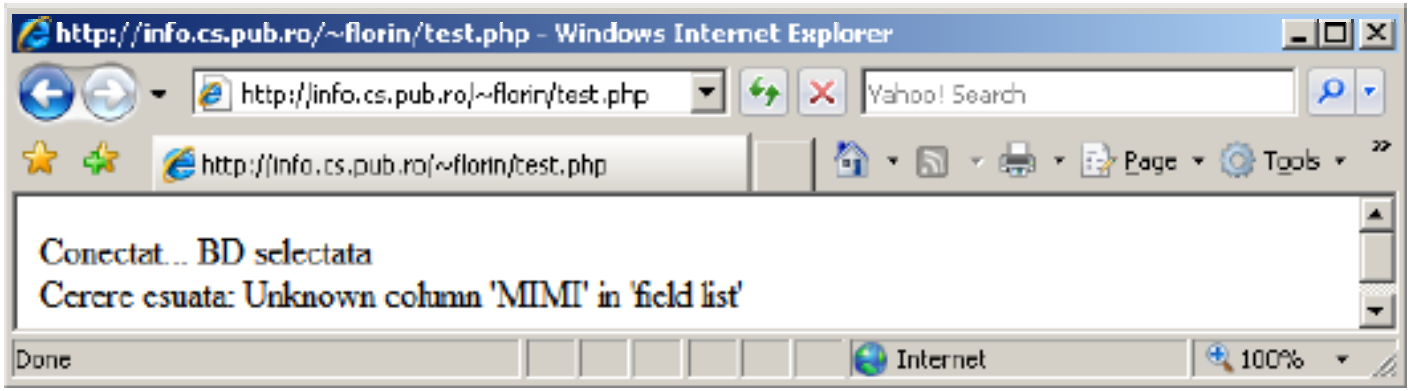


Exemple

- Tabelă inexistentă:



- Coloană inexistentă:



Funcții MySQL - Fetch

- Încărcarea unei linii din rezultatul întors de o cerere SELECT se poate face folosind una dintre funcțiile următoare:

1. mysql_fetch_row() – Întoarce o linie de rezultat ca tablou cu indici numerici.
2. mysql_fetch_array() - Întoarce o linie de rezultat ca tablou ce poate fi folosit atât cu indici numerici cât și ca tablou asociativ.
3. mysql_fetch_assoc() - Întoarce o linie de rezultat ca tablou asociativ.
4. mysql_fetch_object() - Întoarce o linie de rezultat ca obiect.

Funcții MySQL - Fetch

array mysql_fetch_row(resource rezultat);

- Întoarce un array (neasociativ, accesat prin indici) cu valorile următoarei linii din rezultat sau *false* dacă nu mai sunt linii.
- Exemplu:

```
while($linie = mysql_fetch_array($rezultat)) {
    echo $linie[0];
    echo $linie[1];
    echo $linie[2];
}
```

Funcții MySQL - Fetch

array mysql_fetch_array(resource rezultat, int [tip_rezultat]);

- Încarcă următoarea linie din rezultat într-un array asociativ.
- În caz de eroare întoarce *false*.
- Este asemănătoare cu mysql_fetch_row() dar elementele liniei pot fi accesate și după numele coloanei din rezultat.
- În cazul în care mai multe coloane au același nume (pentru un join de exemplu) ultima coloana cu acel nume va avea prioritate, celelalte putând fi accesate prin indici.

Funcții MySQL - Fetch

- Tipul rezultatului (parametru opțional) poate fi MYSQL_ASSOC, MYSQL_NUM sau valoarea implicită MYSQL_BOTH. Se poate astfel ca tabloul să fie accesat doar ca asociativ sau doar cu indici numerici.
- Exemplu:

```
while($linie = mysql_fetch_array($rezultat))
{ echo $linie["nume"]; }
```

Funcții MySQL - Fetch

array mysql_fetch_asoc(resource rezultat);

- Încarcă următoarea linie din rezultat într-un array asociativ.
- În caz de eroare întoarce *false*.
- Este identic cu mysql_fetch_array având parametrul tip_rezultat egal cu MYSQL_ASSOC.
- Exemplu:

```
while($linie = mysql_fetch_array($rezultat))
{  echo $linie["nume"]; }
```

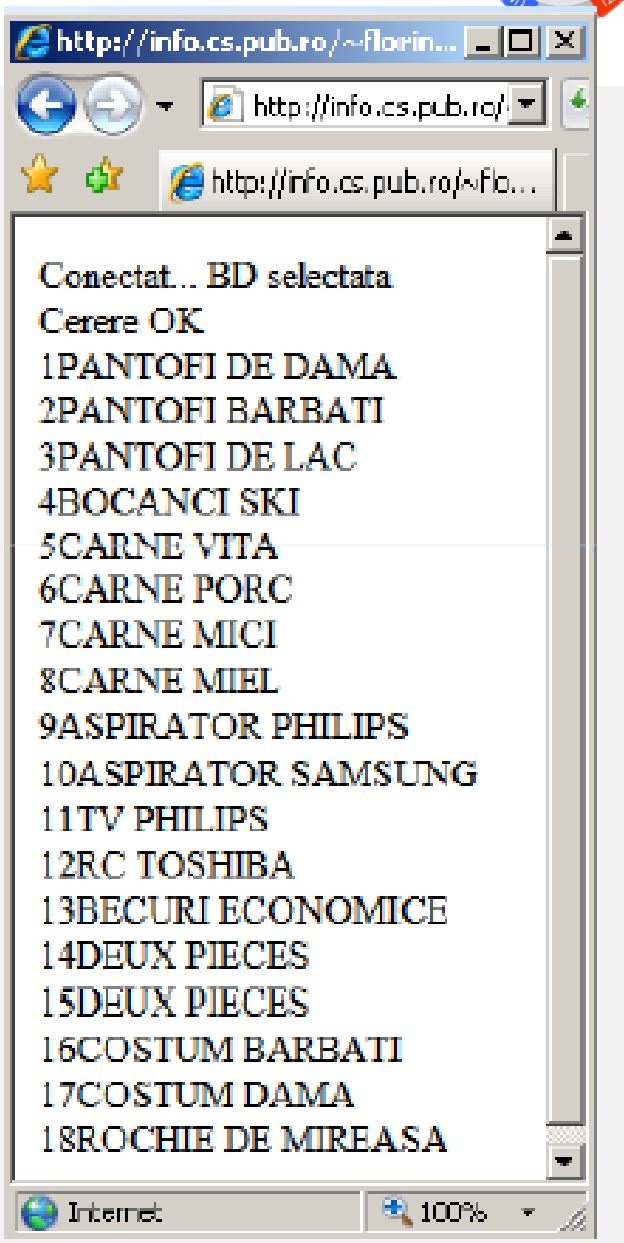
Funcții MySQL - Fetch

array mysql_fetch_object(resource rezultat[, string nume_clasa [, array $parametri]]);

- Încarcă următoarea linie din rezultat într-un obiect.
- În caz de eroare întoarce *false*.
- nume_clasa specifică numele clasei (implicit stdClass).
- Parametrii formează un array care va fi transmis constructorului pentru clasa nume_clasa.
- Nu se mai pot în acest caz accesa elementele înregistrării încărcate prin indici numerici ci doar prin numele lor.

Exemplu

```
<?php
mysql_connect("localhost", "stud", "studpw")
or die('Conexiune esuata');
echo "Conectat...";
// nu folosim descriptor de conexiune la select db
mysql_select_db("stud")
or die ('Selectie BD esuata: '.mysql_error());
echo " BD selectata<br>";
// si nici la query
$rezultat = mysql_query("select * from produse")
or die('Cerere esuata: '.mysql_error());
echo " Cerere OK<br>";
while($linie=mysql_fetch_object($rezultat))
{ echo $linie->codp;
echo $linie->nume, "<br>";
}
?>
```



Funcții MySQL - Fetch

- La încărcarea unei linii de rezultat, pointerul intern în rezultat avansează cu 1. El se poate 'mișca' și altfel folosind funcția

```
int mysql_data_seek(resource id_rezultat, int numar_linie);
```

- Linia specificată devine linia curentă în cadrul rezultatului: următorul apel mysql_fetch_???() va întoarce această linie.
- Întoarce: *true* în caz de reușită, *false* în caz de eșec.
- Notă: Pentru prima linie folosim numărul 0!
- Exemplu:

```
int mysql_data_seek($rezultat, $j);
```

Funcții MySQL – Informații

int mysql_num_fields(resource rezultat);

- Întoarce numărul de câmpuri (coloane) dintr-un rezultat de cerere SELECT.

int mysql_num_rows(resource rezultat);

- Întoarce numărul de linii dintr-un rezultat de cerere SELECT.

object mysql_fetch_field(resource rezultat, int [offset_camp]);

- Întoarce un obiect conținând informații despre un câmp (coloana) al rezultatului. Dacă offsetul de câmp nu este specificat este considerat următorul câmp al rezultatului (se poate seta cu mysql_field_seek()). Pornește de la 0!

Funcții MySQL – Informații

- Proprietățile obiectului întors de mysql_fetch_field sunt:
- *name* - nume coloană
- *table* - numele tabelei din care provine acea coloană
- *def* – valoarea implicită a coloanei
- *max_length* - lungimea maximă pentru acea coloană
- *not_null* - 1 dacă acea coloană nu poate conține valori nule.
- *primary_key* - 1 dacă acea coloană este o cheie primară.
- *unique_key* - 1 dacă acea coloană este cheie unică
- *multiple_key* - 1 dacă acea coloană este o cheie neunică
- *numeric* - 1 pentru coloana numerică
- *blob* - 1 pentru coloana de tip BLOB
- *type* - tipul coloanei
- *unsigned* - 1 pentru coloane unsigned
- *zerofill* - 1 pentru coloane zero-filled

Funcții MySQL – Informații

string mysql_field_name(resource rezultat, int numar_camp);

- Întoarce numele câmpului cu numărul specificat dintr-un rezultat (pornește de la 0!).
- Exemplu:

```
$numecamp = mysql_field_name($rezultat, $j);
```

string mysql_field_type(resource rezultat, int offset_camp);

- Este similar cu mysql_field_name() întorcând însă tipul câmpului specificat (pornește de la 0!).

int mysql_field_len(resource rezultat, int offset_camp);

- Întoarce lungimea câmpului specificat (pornește de la 0!).

Funcții MySQL - Modificare

int mysql_affected_rows(resource [id_conexiune]);

- Întoarce numărul de linii afectate de ultima cerere INSERT, UPDATE sau DELETE pe conexiunea specificată sau în lipsa acestui parametru pe ultima conexiune deschisă.
- Exemplu:

```
mysql_affected_rows($conexiune);
```

int mysql_insert_id(resource [id_conexiune]);

- Întoarce valoarea generată de ultimul INSERT executat pentru o coloană definită cu AUTO_INCREMENT.

Funcții MySQL - Terminare

int mysql_free_rezultat(resource rezultat);

- Apelul acestei funcții eliberează spațiul ocupat de rezultatul unei cereri.

int mysql_close(resource [id_conexiune]);

- Închide conexiunea specificată sau în lipsă ultima conexiune deschisă.
- Întoarce *false* în caz de eșec, altfel *true*.
- Nu închide conexiunile deschise cu mysql_pconnect().
- Exemplu:

```
int mysql_close($conexiune);
```

Funcții MySQL

- Pe lângă funcțiile prezentate există și altele care se pot consulta în documentația PHP.
- Unele dintre acestea vor fi prezentate în finalul lecției, în contextul rezolvării problemelor de securitate (cum este mysql_real_escape_string)

Funcții Oracle - conectare

resource oci_connect (string user , string passw [, string bd [, string setcar [, int mod_sesiune]]])

- Se realizează conectarea la serverul Oracle local cu userul și parola specificate.
- Dacă lipsește numele instanței (bd) PHP îl ia din variabila env. ORACLE_SID.
- Setul de caractere se poate indica începând cu versiunea 9.2 de Oracle.
- mod_sesiune poate fi OCI_DEFAULT, OCI_SYSOPER și OCI_SYSDBA. Ultimele două stabilesc o sesiune privilegiată.
- Există și aliasul ocilogon(...) cu aceleași efecte.

Funcții Oracle - conectare

resource oci_pconnect (string user , string passw [, string bd [, string setcar [, int mod_sesiune]]])

- Are aceiași parametri ca precedenta.
- Funcția stabilește o conexiune persistentă, care nu se închide la terminarea scriptului ci e refolosită de scripturile următoare.
- În felul acesta se micșorează overhead-ul de sistem.

Funcții Oracle - conectare

resource oci_new_connect (string user , string passw [, string bd [, string setcar [, int mod_sesiune]]])

- Deschide o nouă conexiune, chiar daca există deja una (celelalte 2 funcții nu fac asta).
- Toate 3 returnează un descriptor de conexiune (echiv. *TRUE*) sau *FALSE* în caz de eroare.

Funcții Oracle - Parse

- Al doilea pas dupa conectare este compilarea (parsingul) cererii.
- Aceasta se face cu funcția:

resource oci_parse (resource conexiune , string cerere)

- Returnează un descriptor de cerere (statement handle) necesar execuției efective.
- Exemplu:

```
$c = oci_connect("scott","tiger");
$cerere = "select * from STUD where
   cods=".$_REQUEST['cods'];
echo "<h2>Cererea este: $cerere</h2>";
$stmt = oci_parse($c, $cerere);
```

Funcții Oracle – Execuție

- Execuția efectivă a cererii se face cu:

```
bool oci_execute ( resource $stmt [, int mod])
```

- Parametrii sunt: un descriptor de cerere returnat de oci_parse și opțional modul de tratare al tranzacțiilor.
- Modul poate fi:

  OCI_COMMIT_ON_SUCCESS (opțiunea implicită) și OCI_DEFAULT.

Funcții Oracle – Execuție

OCI_DEFAULT:

- În al doilea caz, execuția unei cereri DML duce la demararea unei tranzacții.
- Această tranzacție va fi revocată automat la terminarea scriptului sau la închiderea conexiunii.
- Pentru a controla această tranzacție se poate folosi oci_commit și oci_rollback.

Funcții Oracle – Execuție

bool oci_commit (resource conexiune);

bool oci_rollback (resource conexiune);

- Efectul lor este de a comite, respectiv revoca tranzacția în curs pe acea conexiune.
- Această tranzacție poate fi formată din execuția uneia sau mai multor cereri DML.
- Returnează *TRUE* / *FALSE* în caz de succes/eșec.

Funcții Oracle - Rezultat

- Pentru a afla dimensiunea rezultatului se pot folosi:

```
int oci_num_rows ( resource $stmt )
```

- Întoarce numărul de linii afectate de execuția cererii (câte linii returnează un select dar și câte linii șterge un *delete* de exemplu)

```
int oci_num_fields ( resource $stmt )
```

- Returnează numărul de coloane ale rezultatului execuției unei cereri SELECT.

```
string oci_field_name ( resource stmt , int coloana )
```

- Returnează numele coloanei cu numărul specificat (prima coloană are indicele 1)

Exemplu

```
. . . . . .
$stmt = oci_parse($c, $cerere);
// executie
oci_execute($stmt,OCI_DEFAULT);
// rezultatul va fi prezentat ca tabela
echo "<table border = 1>";
// luam in nc nr. de coloane in rezultat
$nc = oci_num_fields($stmt);
// afisam antet tabel
echo "<tr>";
for($i=1;$i<=$nc;$i++)
echo "<td>".oci_field_name($stmt, $i)."</td>";
echo "</tr>";
. . . . . .
```

Funcții Oracle - Fetch

- Pentru încărcarea liniilor există 4 funcții asemănătoare cu cele din MySQL:

array oci_fetch_row (resource stmt)

- Întoarce tablou cu indici numerici

array oci_fetch_assoc (resource stmt)

- Întoarce tablou asociativ

array oci_fetch_object (resource stmt)

- Întoarce obiect

Funcții Oracle - Fetch

array oci_fetch_array (resource stmt [, int mod])

- Modul poate fi:

1. **OCI_BOTH** - întoarce array asociativ și cu indici numerici - valoarea implicită.
2. **OCI_ASSOC** - întoarce array asociativ
3. **OCI_NUM** - întoarce array cu indici numerici
4. **OCI_RETURN_NULLS** - crează elemente goale pentru valorile nule
5. **OCI_RETURN_LOBS** - returnează valoarea LOB-ului unui descriptor

Exemplu

```
<?php
    $connection = oci_connect("stud", "studpw");
    $cerere = "SELECT nume, pret, coloana_lob FROM produse";

    $stmt = oci_parse ($connection, $cerere);
    oci_execute ($stmt);

    while ($lin = oci_fetch_array ($stmt, OCI_RETURN_LOBS)) {
            echo $lin[0]."<br>";
            echo $lin[1]."<br>";
            echo $lin['COLOANA_LOB']."<br>";
            // ultima linie afiseaza continutul col. LOB
    }
?>
```

Alt exemplu

- Continuarea exemplelor anterioare:

```
while ($linie=oci_fetch_row($stmt)) {
    // incepe linie date
    echo "<tr>";
    for($i=0;$i<$nc;$i++) {
                // daca e valoare nula o inlocuim cu sirul NULL
                if (!isset($linie[$i]))
                        $linie[$i] = "NULL";
                // afisam efectiv o celula
                echo "<td>".$linie[$i]."</td>";
    }
    // gata linie date
    echo "</tr>";
}
```

PL/SQL

- Prezentăm în continuare un exemplu de execuție a unui bloc PL/SQL în PHP.
- Se folosește funcția oci_bind_by_name pentru punerea în corespondență a variabilelor externe PL/SQL cu variabile PHP.
- Sintaxa funcției este (vezi și documentația):

```
bool oci_bind_by_name ( resource stmt , string
   nume_oracle , mixed &var_PHP [, int
   lung_maxima = -1 [, int tip = SQLT_CHR ]] )
```

Fetch all

int oci_fetch_all (resource stmt , array &output [, int skip= 0 [, int maxlinii = -1 [, int $flags= 0]]])

- Încarcă toate liniile unui rezultat SELECT (sau un număr specificat de linii) într-un array.
- Întoarce numărul de linii încărcate.
- Exemplu:

```
$conn = oci_connect(. . . );
$stmt = oci_parse($conn, "select * from emp");
oci_execute($stmt);
$nrows = oci_fetch_all($stmt, $results);
if ($nrows > 0) {
foreach ($results as $key => $val) {
echo "$key, $val\n";
}
```

PL/SQL

```
$c = oci_connect("stud","studpw");
// variabile: $cati (PHP) si :cati (externa Oracle)
$cerere = "begin
    select count(*) into :cati from STUD
    where cods=".$_REQUEST['cods']."; ".
    "exception
    when others then
    :cati := 0;
    end;"; // atentie: ; de dupa end!
$stmt = oci_parse($c, $cerere);
// bind by name:
oci_bind_by_name($stmt, ":cati", $cati, 32);
// executie
oci_execute($stmt,OCI_DEFAULT);
// rezultatul va fi in variabila PHP $cati
echo "Numar de studenti: ".$cati;
```

Alt exemplu

```
$c = oci_connect("scott","tiger");
$cerere = "begin
    select cods, nume into :cods, :nume from STUD
    where matr=".$_REQUEST['matr']."; ".
    "exception
    when others then
    :cods := 0;
    :nume := 'nu avem astfel de student';
    end;"; // nu trebuie uitat ; dupa end
$stmt = oci_parse($c, $cerere);
// bind by name: se asociaza 2 variabile externe cu var. PHP
oci_bind_by_name($stmt, ":cods", $cods, 32);
oci_bind_by_name($stmt, ":nume", $nume, 32);
// executie
oci_execute($stmt,OCI_DEFAULT);
echo $cods, " ", $nume;
```

Funcții Oracle - Erori

- Descrierea erorilor se poate obține cu:

```
array oci_error ([ resource $sursa ] )
```

- Pentru erorile de conexiune nu se indică sursa
- Pentru celelalte se pune descriptorul cel mai apropiat
- Array-ul conține elementele:
  - `code` (codul de eroare)
  - `message` (mesajul)
  - `offset` (pozitia erorii)
  - `sqltext` (textul cererii eronate)

Exemplu

```
if (!($r = oci_execute($stmt))) {
   // se intra aici in caz de eroare
   $e = oci_error($stmt);
   echo htmlentities($e['message']);
   echo "<pre>";
   echo htmlentities($e['sqltext']);
   printf("\n%".($e['offset']+1)."s", " ^"); // o
        sageata la pozitia  erorii
   echo "</pre>";
}
```

Funcții Oracle - Final

- Închiderea conexiunii se face cu:

bool oci_close (resource conexiune)

- Pe lângă funcțiile prezentate mai există și altele (vezi documentația)

Elemente de securitate

- În continuare prezentăm doar câteva elemente de securitate pentru aplicații scrise în PHP și care accesează o bază de date.
- Una dintre problemele principale este aceea că unui script care în mod obișnuit este asociat unui formular i se pot transmite parametri - inclusiv în bara de adresă (metoda GET) – care nevalidați pot duce la injecție SQL.

Injecția SQL

- Injecția SQL înseamnă execuția unei cereri nedorite inserată într-un șir de caractere executat de serverul de baze de date.
- Ea apare ca urmare a nevalidării datelor de intrare și a ignorării unor măsuri minime de securitate.

Comentarii

- Semnul de comentarii poate afecta cererea SQL executată cu un parametru primit.
- Exemplu: fie o cerere de tipul

```
SELECT * FROM members WHERE username = 'param1'
    AND password = 'param2'
```

- Dacă pentru param1 se introduce valoarea: admin'--
  cererea executată va fi:

```
SELECT * FROM members WHERE username = 'admin' --
    AND password = 'param2'
```

- Se observă că partea de verificare parolă din cerere este acum comentată - deci nu se mai verifică și parola ci doar username.

Comentarii

- Variante (SqlServer, MySQL, Oracle):
  - admin' –
  - admin' #
  - admin'/*
  - ' or 1=1—
  - ' or 1=1#
  - ' or 1=1/*
  - ') or '1'='1—
  - ') or ('1'='1--

Date fără '

- O recomandare general acceptată este aceea de a pune între apostrofe toate elementele primite ca intrare de la user.
- Iată un exemplu (bine și mai puțin bine)

```
$query = "SELECT * FROM tabela where
  col = '" . $_REQUEST['ceva'] . "'";
```

```
$query = "SELECT * FROM tabela where
  col = " . $_REQUEST['ceva'];
```

Date fără '

- Ce se poate întâmpla în al doilea caz:
- Input: 1 or (1=1). Cererea devine:

```
$query = "SELECT * FROM tabela where col = 1
   or (1=1)"
```

- Vor fi întoarse deci toate liniile tabelei și nu doar câteva.
- Input: 1 and (1=0); drop table tabela;
- Cererea devine:

```
$query = "SELECT * FROM tabela where col = 1
   and (1=0); drop table tabela; "
```

Cereri multiple

- În acest caz șirul de executat conține două cereri SQL, dintre care a 2-a este distructivă.
- Există sisteme de gestiune care execută astfel de succesiuni de cereri primite din PHP (postgresql, mssql)
- Soluția este fie folosirea apostrofelor, fie folosirea mecanismului de marcare a unor caractere speciale (inclusiv apostrof).

Escape string

- Dacă parametrul PHP (din PHP.ini) *magic_quotes_gpc* este trecut pe On, toate aceste caractere speciale sunt prefixate cu backslash.
- După înlăturarea acestuia, se poate folosi mecanismul propriu de escape al SGBD-ului respectiv.
- Iată un exemplu:

Exemplu

```
<?php
    if (get_magic_quotes_gpc()) {
        $nume = stripslashes($nume);
        $descriere = stripslashes($descriere);
        // Cererea:
        query = sprintf(
                "SELECT * FROM users WHERE user='%s'
                AND password='%s'",
        mysql_real_escape_string($nume),
        mysql_real_escape_string($descriere));
    . . . . .
?>
```

Exemplu

- Funcția mysql_real_escape_string va prefixa cu \ caracterele speciale din MySQL: \x00, \n, \r, \, ', " si \x1a.
- Analog, prin folosirea primei sintaxe (cu '), cele două cereri anterioare devin:

```
$query = "SELECT * FROM tabela where col = '1 or (1=1)'"

$query = "SELECT * FROM tabela where col = '1 and (1=0); drop table tabela;' "
```

Bibliografie

- Documentația PHP
  - http://www.php.net/docs.php
- Câteva cărți disponibile online:
  - Chris Newman: Sams - Teach Yourself PHP in 10 Minutes(2005): http://www.net130.com/CMS/Pub/book/book_web/book_web_php/2005_10_19_70383.htm
  - Ilia Alshanetsky: PHP Architects Guide to PHP Security (și altele), la adresa: http://cid-846ffdcf0d3320d8.skydrive.live.com/browse.aspx/eBook
  - SQL Injection Cheat Sheet: http://ferruh.mavituna.com/sql-injection-cheatsheet-oku/