PHP e DBMS

Laboratorio di basi di dati
Stefano Montanelli
Dipartimento di Informatica
Università degli Studi di Milano
http://islab.di.unimi.it/bdlab1

PHP: interazione con i DBMS

- L'utilizzo di DBMS mediante PHP si basa su una sequenza di azioni fondamentali
  1. Connessione al DBMS e handle della connessione
  2. Invio di comandi SQL al DBMS
  3. Analisi ed elaborazione del risultato prodotto dal comando SQL inviato
  4. Rilascio delle risorse e della connessione

Librerie di funzioni

- PHP è dotato di moduli (librerie) specifici per interagire con i vari DBMS
- Ogni DBMS ha una propria libreria di funzioni (realizzata come estensione di PHP)
- E' necessario abilitare in php.ini le librerie di interesse

- Sono disponibili ulteriori librerie di funzioni con finalità di astrazione dello specifico DBMS che consentono di scrivere codice compatibile con svariati DBMS (entro limiti dipendenti dall'implementazione)
  - e.g., PEAR

Interazione con PostgreSQL

- Per l'interazione con PostgreSQL, la corrispondente libreria di funzioni (php_pgsql) mette a disposizione le seguenti primitive:
  - Connessione
    - pg_connect (connessione semplice)
    - pg_pconnect (connessione persistente)
  - Invio comandi SQL al DBMS
    - pg_prepare e pg_execute
  - Elaborazione del risultato
    - pg_fetch_array
  - Rilascio del risultato
    - pg_free_result e pg_close

L’istruzione pg_query

- L’istruzione pg_query è spesso utilizzata come primitiva per inviare comandi SQL al DBMS in alternativa a pg_prepare e pg_execute
- **Si tratta di una soluzione fortemente sconsigliata** poiché introduce vulnerabilità e rende l’applicazione debole rispetto agli attacchi di sql injection

Connessione al DBMS

- Sono disponibili due tipologie di connessione
  - Semplice (pg_connect)
    - Viene chiusa alla terminazione dello script o tramite l'istruzione pg_close
  - Persistente (pg_pconnect)
    - Utilizza, se presente, una connessione precedentemente stabilita
    - Non viene chiusa alla terminazione dello script
    - Non viene chiusa dall'istruzione pg_close
    - Si veda il manuale di PHP per ulteriori informazioni (capitolo 41 - Persistent Database Connections)

Sintassi di connessione

- Sintassi

  resource pg_connect (connection_string

  [, connection_type])

  – Il parametro connection_string contiene i valori per le seguenti variabili:
    - host=<nomehost>
    - port=<numeroporta>
    - dbname=<nomedb>
    - user=<username>
    - password=<password>
  – Il parametro connection_type viene usato per forzare esplicitamente l'uso di una nuova connessione

Esempio di connessione

$db = pg_connect (“host=localhost user=usertest password=userpsw dbname=testdb”);

- Tutti i parametri hanno un valore di default
- L’handle ottenuto ($db) è utilizzato nelle successive chiamate al DBMS per indicare la connessione sulla quale si desidera operare

Invio di comandi SQL al DBMS

- Tramite le istruzioni pg_prepare e pg_execute è possibile inviare comandi SQL alla base di dati in modo parametrico. Questo permette di
  - Selezionare dati (istruzione SELECT)
  - Inserire dati (istruzione INSERT)
  - Modificare dati (istruzione UPDATE)
  - Eliminare dati (istruzione DELETE)

Istruzioni prepare - execute

resource pg_prepare (connection_handle, cmd_label, sql_cmd)

- L'istruzione pg_prepare compone e pre-analizza il comando SQL in *sql_cmd* senza però inviarla al DBMS ed eseguirla. Normalmente, sql_cmd contiene i segnaposto (indicati con il carattere ?) che saranno sostituiti dai parametri passati in fase di esecuzione. *Connection_handle* identifica la connessione verso il DBMS aperta in precedenza. *Cmd_label* è un'etichetta che identifica l'istruzione prepare all'interno dello script.

resource pg_execute (connection_handle, cmd_label, params)

- L'istruzione pg_execute invia al DBMS e richiede l'esecuzione del comando sql_cmd preparato con pg_prepare sostituendo i segnaposto con i parametri contenuti in *params*

Esempi di prepare - execute

- Selezionare dati
  - $sql= “SELECT * FROM mytab”;
  - $value = array();
- Inserire dati
  - $sql= “INSERT INTO mytab VALUES ($1, $2)”;
  - $value = array(‘val1’,’val2’);
- Modificare dati
  - $sql= “UPDATE mytab SET attr1=$1 WHERE attr2=$2”;
  - $value = array(‘val1’,’val2’);
- Cancellare dati
  - $sql= “DELETE FROM mytab WHERE attr1=$1”;
  - $value = array(‘val1’);

$resource = pg_prepare($db, “cmd”, $sql);
$resource = pg_execute($db, ”cmd”, $value);

Esito dell'istruzione execute

- L'istruzione pg_execute restituisce un handle (un puntatore a una risorsa) che identifica il risultato del comando SQL inviato alla base di dati
- Tramite questo handle sarà possibile eseguire elaborazioni sul risultato
- A supporto dell'elaborazione, è possibile utilizzare le seguenti istruzioni per sapere quante tuple sono state interessate da pg_execute
  - *pg_num_rows*. Restituisce il numero di tuple ritornate da pg_execute quando il comando SQL è SELECT
  - *pg_affected_rows*. Restituisce il numero di tuple modificate da pg_execute quando il comando SQL è INSERT, UPDATE, o DELETE

Elaborazione del risultato

- Nel caso di pg_execute associata ad un comando SELECT, il risultato dell'istruzione è un array costituito da un elemento per ogni tupla restituita dal DBMS
- L'istruzione pg_fetch_array scorre l'array del risultato:

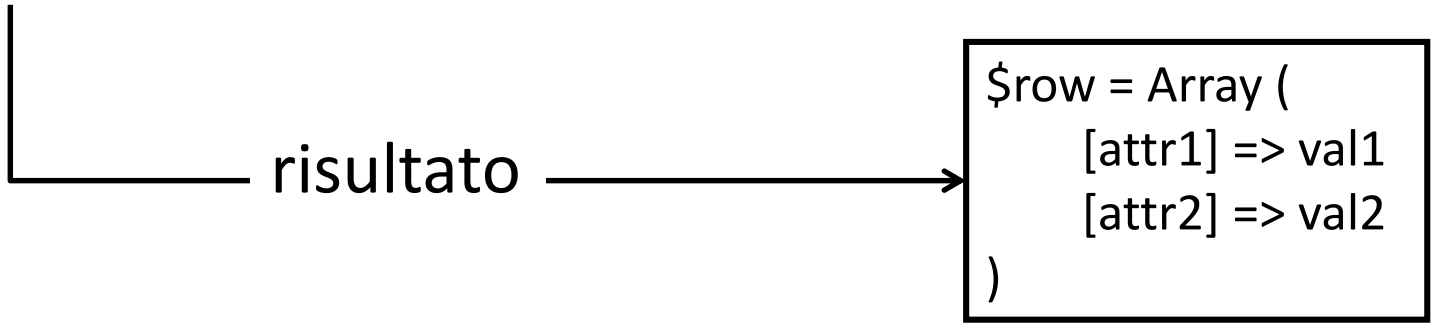
  array pg_fetch_array (result_handle
  [, row_index [, result_type]])

  - Result_handle è l'handle del risultato
  - Row_index è l'indice della riga da caricare
    - la numerazione parte dall'indice 0
    - se omesso, viene caricata la prima tupla disponibile
  - Result_type è uno a scelta fra
    - PGSQL_ASSOC (l'array del risultato è associativo)
    - PGSQL_NUM (l'array del risultato è numerico)
    - PGSQL_BOTH (l'array del risultato è sia associativo sia numerico – opzione di default)

Esempio di fetch con array associativo

```
$sql= “SELECT * FROM mytab;”;
$resource = pg_prepare($db, ”cmd”, $sql);
$value = array();
$resource = pg_execute($db, ”cmd”, $value);
$row = pg_fetch_array($resource, NULL, PGSQL_ASSOC);
print_r ($row);
```

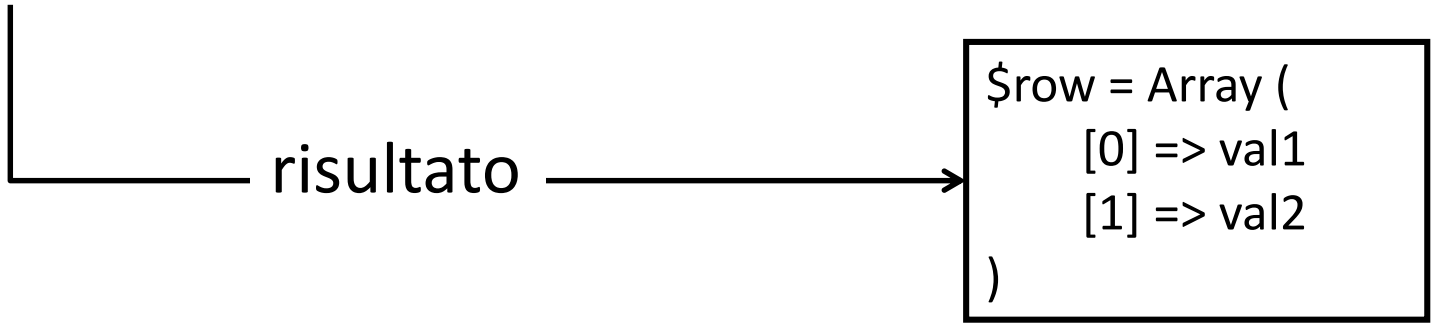
MYTAB	
attr1	attr2
val1	val2



Esempio di fetch con array numerico

```
$sql= “SELECT * FROM mytab;”;
$resource = pg_prepare($db, ”cmd”, $sql);
$value = array();
$resource = pg_execute($db, ”cmd”, $value);
$row = pg_fetch_array($resource, NULL, PGSQL_NUM);
print_r ($row);
```

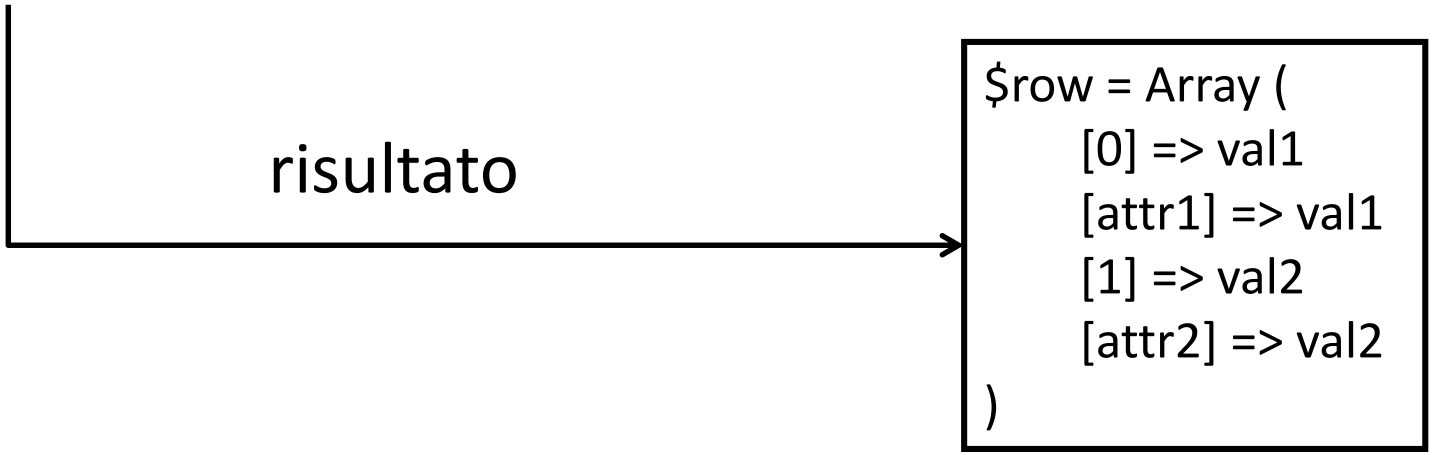
MYTAB	
attr1	attr2
val1	val2



Esempio di fetch con array assoc/num

```
$sql= “SELECT * FROM mytab;”;
$resource = pg_prepare($db, ”cmd”, $sql);
$value = array();
$resource = pg_execute($db, ”cmd”, $value);
$row = pg_fetch_array($resource, NULL, PGSQL_BOTH);
print_r ($row);
```

MYTAB	
attr1	attr2
val1	val2



Istruzioni analoghe a pg_fetch_array

- Oltre a pg_fetch_array, PHP offre altre istruzioni per caricare in un array il risultato di uno statement SQL
  - e.g., *array pg_fetch_row*(result [, $row])
  - e.g., *array pg_fetch_assoc*(result [, $row])
- Queste istruzioni si differenziano per segnatura, ma sono analoghe per funzionalità
  - si veda il manuale per maggiori dettagli
- In ogni caso, quando il risultato è costituito da più di una tupla è necessario utilizzare una struttura iterativa per elaborare le tuple del risultato in modo sequenziale

Rilascio delle risorse

- Il risultato di una pg_execute viene caricato in memoria principale
- Questo comporta un significativo consumo di risorse in caso di script contenenti numerose istruzioni pg_execute aventi molte tuple come risultato
- Durante l'interazione con un DBMS è consigliabile liberare progressivamente la memoria occupata dai risultati delle istruzioni pg_execute che non sono più necessari

  pg_free_result($resource)

  – *$resource* è l'handle di risultato restituito da una istruzione pg_execute
- In ogni caso, la memoria viene liberata al termine dell'esecuzione dello script

Chiusura della connessione

- Quando non sono più necessarie, è consigliabile chiudere le connessioni aperte con i vari DBMS

  pg_close($db)

  – $db è l'handle di connessione restituito da una istruzione pg_connect
- In ogni caso, le connessioni NON persistenti vengono chiuse automaticamente al termine dell'esecuzione dello script
- Anche se viene utilizzata l'istruzione pg_close, le connessioni persistenti non vengono chiuse