Php - mysql

Ανάπτυξη Διαδικτυακών Συστημάτων & Εφαρμογών

Τμ. Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΔιΠαΕ

Αντώνης Σιδηρόπουλος

SQL Basics

Queries

DDL(Data Definition Language)

CREATE TABLE - creates a new table
ALTER TABLE - modifies a table
DROP TABLE - deletes a table
CREATE INDEX - creates an index (search key)
DROP INDEX - deletes an index

DML(Data Manipulation Language)

SELECT - extracts data from a database
UPDATE - updates data in a database
DELETE - deletes data from a database
INSERT INTO - inserts new data into a database

Basic SQL Queries


```
CREATE DATABASE test;
```

```
CREATE TABLE `students` ( `id` int(11) AUTO_INCREMENT, `name` varchar(40), `email_id` varchar(30), `dob` date, `password` varchar(40), `fruit` varchar(10), PRIMARY KEY (`id`) );
```

```
INSERT INTO students (name, email_id, dob, password) VALUES ('your_name' ,' your_email@id.com ' , '1992-11-13' , md5('your_password'));
```

```
SELECT * FROM students;
```

```
SELECT password FROM students WHERE email_id="your_email@id.com";
```

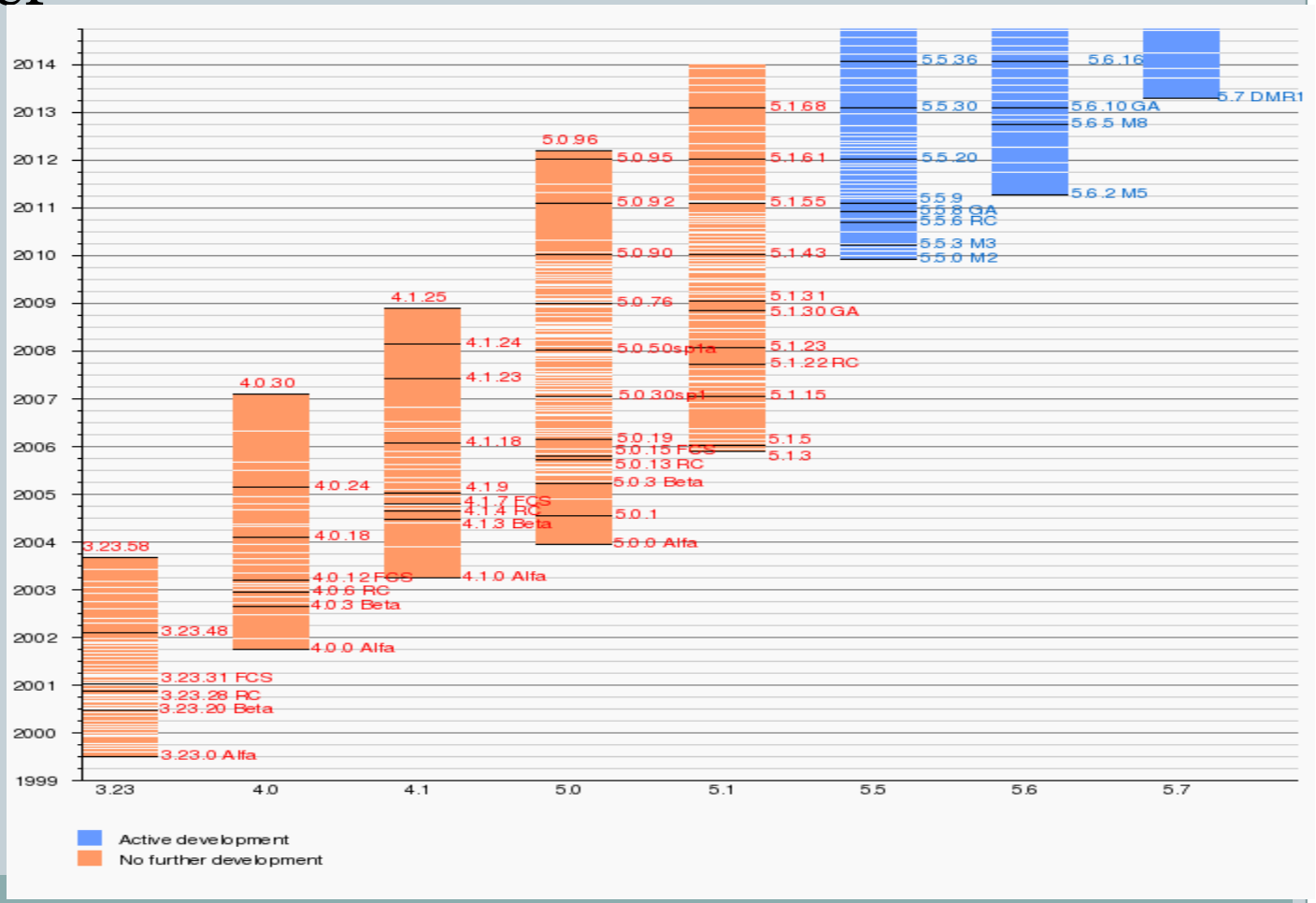
```
SELECT fruit, COUNT( * ) FROM students GROUP BY fruit
```

Mysql

- Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS), ανοικτού κώδικα.
- Από τον Ιούλιο του 2014 το 2ο πιο ευρέως χρησιμοποιούμενο σύστημα.
- Πλεονεκτήματα:
  - υψηλή απόδοση (?),
  - αξιοπιστία (?)
  - ευκολία χρήσης
  - δωρεάν

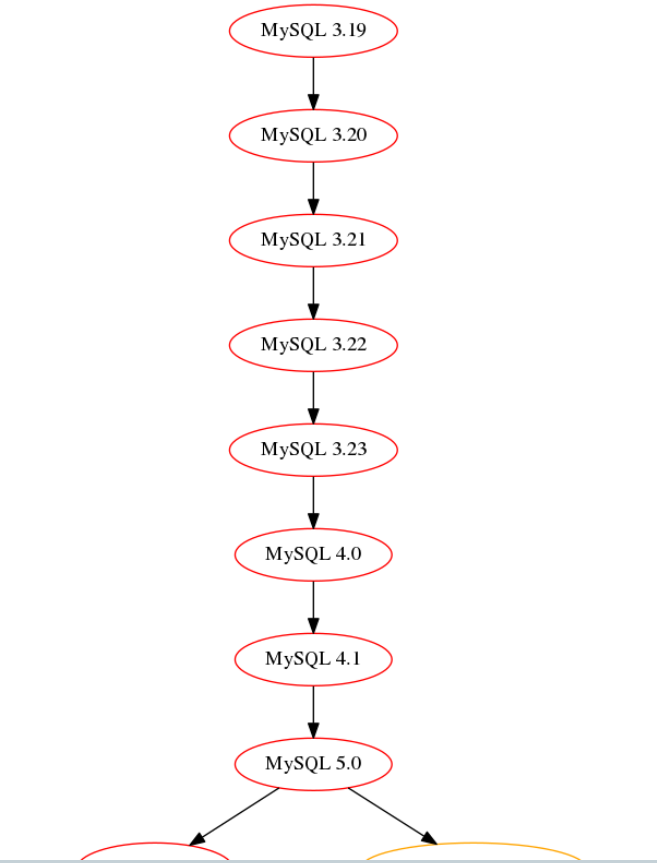
mysql

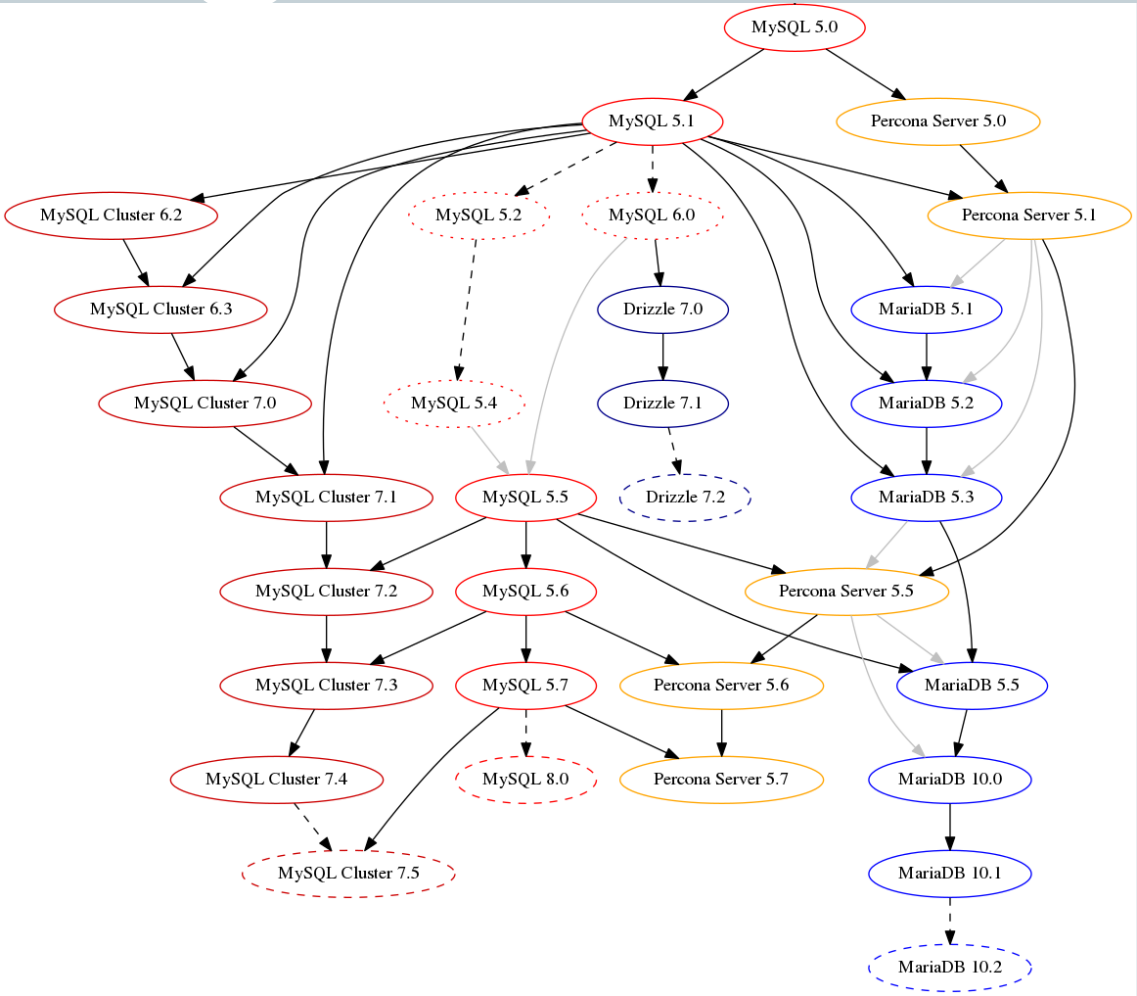
- mysql server



Mysql & derivatives

- mysql server





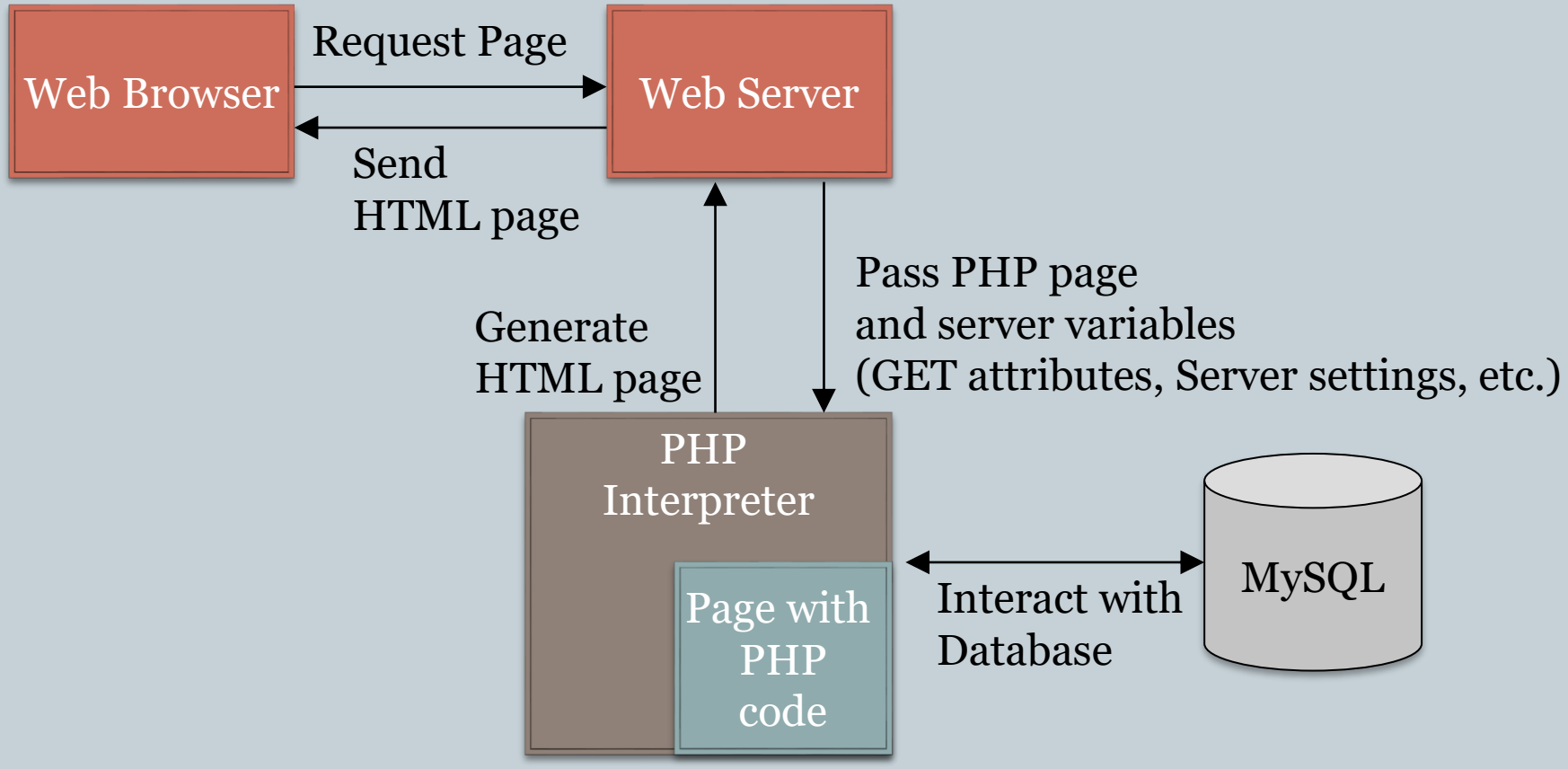
mysql client

- ένας client (μπορεί να) είναι ανεξάρτητος από τον server και την έκδοσή του.
- Clients:
  - Mysql Command line navite client (εγκαθίσταται με την mysql αλλά είναι δύσχρηστος)
  - Mysql WorkBench: Παρέχεται δωρεάν. είναι desktop application (Συνίσταται)
  - phpMyAdmin: Web based client. Πρέπει να γίνεται προσεκτική εγκατάσταση – υπάρχει περίπτωση κάποιος hacker να δεί ΟΛΗ την βάση μας!!
  - SQLyog: Εμπορικός desktop application client. (Συνίσταται)

Επικοινωνία με τον server

- Μέσω TCP/IP
  - Πχ: localhost:3306
- Μέσω unix socket:
  - Πχ:
    - /var/run/mysqld/mysqld.sock
    - /home/staff/it/asidirop/mysql/run/mysql.sock
- Στην προσωπική σας mysql στο users η πρόσβαση γίνεται μόνο τοπικά μέσω του socket: **$HOME/mysql/run/mysql.sock**
- Στα εργαστήρια η πρόσβαση γίνεται μέσω **tcp/ip** μόνο τοπικά
- Εάν εγκαταστήσετε εσείς την mysql μπορείτε να επιλέξετε την μέθοδο πρόσβασης. (windows μόνο tcp/ip)

php ⇔ database



php ⇔ database

- Η php έχει ενσωματωμένες βιβλιοθήκες για πρόσβαση σε βάσεις (σε όλες τις συνηθισμένες) και δεν απαιτείται η εγκατάσταση επι πλέον driver.
- Για την mysql υπάρχουν 3 "βιβλιοθήκες"
  - mysql - Original MySQL API (deprecated as of PHP 5.5.0)
    - functional library
  - mysqli - MySQL Improved Extension
    - functional & Object Oriented library
  - Mysqlnd — MySQL Native Driver
  - PDO - PHP Data Objects
    - Object Oriented library γενικής χρήσης (όπως DBI,ODBC,JDBC)

1. Σύνδεση σε ένα MySQL RDBMS

- Το πρώτο βήμα για να μπορέσουμε να δούμε το περιεχόμενο μίας βάσης είναι να δημιουργήσουμε μια σύνδεση (connection) με την MySQL. Αυτό γίνεται με διάφορους τρόπους, ανάλογα την βιβλιοθήκη που χρησιμοποιούμε:
  - `mysql_connect(<address>, <username>, <password>);`
  - `mysqli_connect(<address>, <username>, <password>, <db>);`
  - `$x = new mysqli(<address>, <username>, <password>, <db>);`

- *Το <address>* είναι η IP διεύθυνση ή το hostname του DB server
- Υπάρχουν και άλλα προερετικά ορίσματα όπως port, socket.

- Η συνάρτηση σύνδεσης επιστρέφει ένα ID (αριθμό) ή ένα object για να μπορούμε να αναγνωρίσουμε τη σύνδεση. Συνήθως αποθηκεύουμε το ID σε μία μεταβλητή, ως εξής :
  - `$dbcnx = mysqli_connect("localhost", "root", "mypasswd");`

1. Σύνδεση σε ένα MySQL RDBMS

- Θα πρέπει να εξετάζεται εάν ο server είναι προσβάσιμος και εάν ο συνδυασμός username/password που δώσαμε δεν γίνεται αποδεκτός από τον server. Σε αντίθετη περίπτωση, η mysqli_connect() δεν επιστρέφει έναν connection identifier αλλά την τιμή false:

```
$dbcnx = mysqli_connect("localhost", "root",
"mypasswd“,'mydb');
if (!$dbcnx) {
    echo("<P>Η σύνδεση με τον " .
    "database server είναι αδύνατη</P>");
    exit();
}
```

Tips

- Το connection το δημιουργούμε σε κάποιο αρχείο, το οποίο το κάνουμε include_once από όπου χρειάζεται. Το αρχείο αυτό δεν θα πρέπει να βρίσκεται μέσα στο document_root ή σε φάκελο στον οποίο έχει άδεια ο web server.
- Για λόγους ασφαλείας, η php πρέπει να συνδέεται στην βάση με username/password που δεν έχει "άδεια" να βλάψει την βάση (όχι με τα ίδια στοιχεία με τα οποία δημιουργούμε την βάση).

2. Επιλογή βάσης MySQL

- **Στην βιβλιοθήκη mysqli, η επιλογή βάσης γίνεται με την σύνδεση**
- Στην βιβλιοθήκη mysql, η επιλογή βάσης πρέπει να γίνει μετά την σύνδεση:

```
mysql_select_db("e-bookstoreDB", $dbcnx);
```

  - Όπου $dbcnx το αναγνωριστικό της σύνδεσης
  - Η mysql_select_db επιστρέφει true όταν είναι επιτυχής και false αν συμβεί κάποιο λάθος. Οπότε με τον παρακάτω κώδικα μπορούμε να κανουμε χειρισμό των λαθών :

```
if (! mysql_select_db("e-bookstoreDB", $dbcnx)) {
   echo( "<P>Αδύνατη η σύνδεση με την βάση</P>" );
   exit();
}
```

Εκτέλεση Insert query


```
$sql = "INSERT INTO category(Name)  Values('New Category')";

if ($mysqli->query($sql)) {
    echo("<P> Η κατηγορία έχει προστεθεί. </P>");
}
else {
    echo("<P>Λάθος: " . $mysqli->error() . "</P>");
}
```

4.2 Εκτέλεση Update query


```
$sql = "Update category SET Name = 'New Name' where ID=5";

if ($mysqli->query($sql)) {
    echo("<P> Το όνομα της κατηγορίας έχει ενημερωθεί. </P>");
}
else {
    echo("<P>Λάθος: " . $mysqli->error() . "</P>");
}
```

4.3 Εκτέλεση Delete query


```
$sql = "Delete category where ID=5";

if ($mysqli->query($sql)) {
    echo("<P> Η κατηγορία έχει διαγραφεί. </P>");
}
else {
    echo("<P>Λάθος: " . $mysqli->error() . "</P>");
}
```

3. Εκτέλεση και διαχείριση αποτελεσμάτων ερωτημάτων Select

- Όταν θέλουμε να εκτελέσουμε ένα ερώτημα SELECT, η συνάρτηση **mysqli_query**() επιστρέφει ένα σύνολο αποτελεσμάτων (result set), που περιέχει μια λίστα όλων των γραμμών (rows).

```
$result = $mysqli_query("SELECT Lname,Fname FROM customer ");
if (!$result) {
    echo("<P> Λάθος στην εκτέλεση του ερωτήματος (query) : " .
mysql_error() . "</P>");
    exit();
}
```

- Εάν το παραπάνω ερώτημα εκτελεσθεί σωστά, ο παραπάνω κώδικας θα τοποθετήσει στη μεταβλητή $result ένα σύνολο αποτελεσμάτων (result set) που θα περιέχει το κείμενο όλων των Lname (επιθέτων) που είναι αποθηκευμένα στον πίνακα Customer. Για να επεξεργαστούμε τις γραμμές (rows) του συνόλου αυτού μία κάθε φορά, μπορούμε να χρησιμοποιήσουμε τον βρόχο while, ως εξής :

```
while ( $row = mysqli_fetch_array($result) ) {
   echo("<P>" . $row["Lname"] . " ". $row["Fname"] . "</P>");
}
```

mysqli_connect() & mysqli_close()

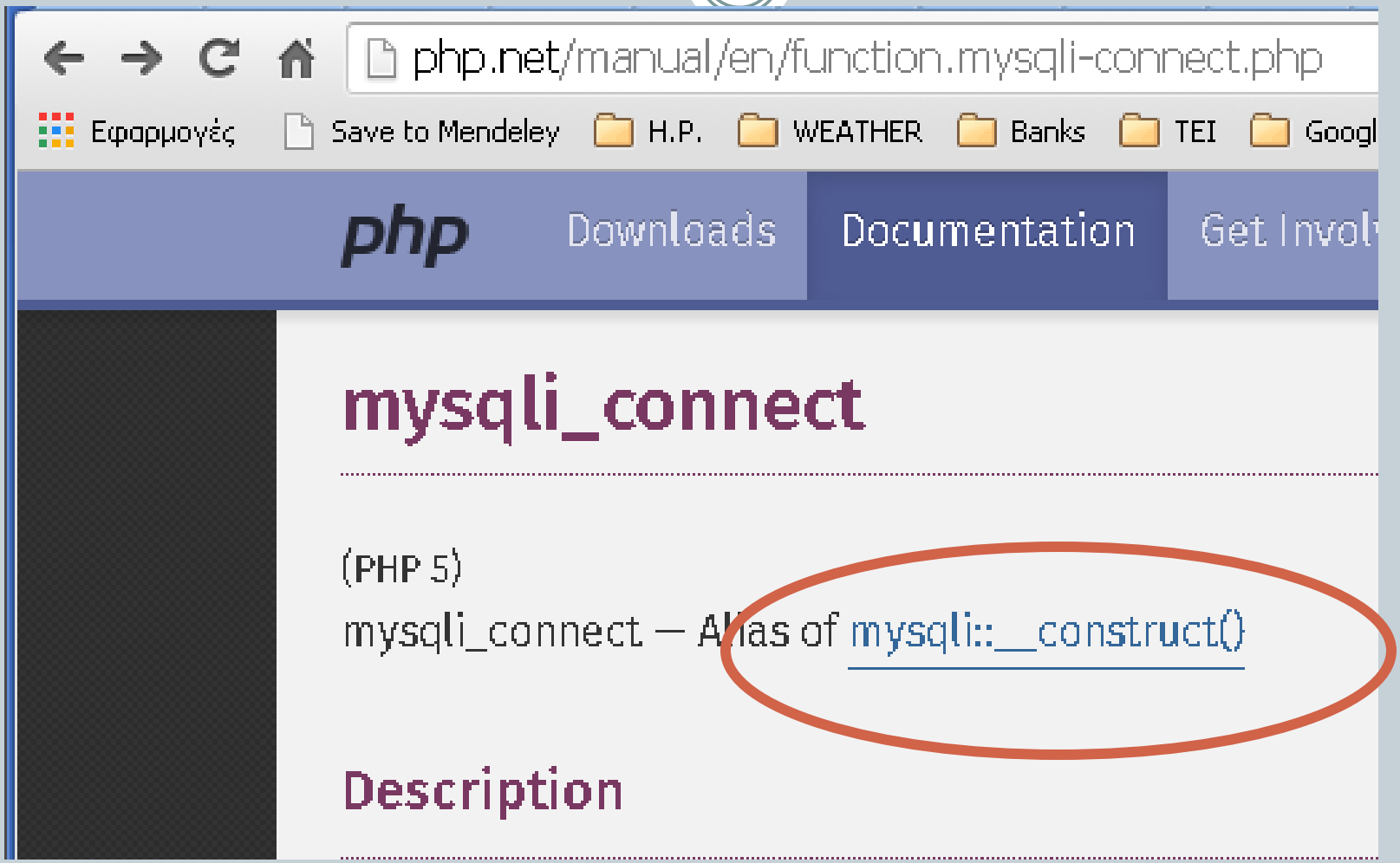
Δημιουργεί και κλείνει μία σύνδεση με ένα MySQL Server

Χρήση:

```
<?php
$con =
mysqli_connect("localhost","mysql_user","mysql_pwd","
db");
if (!$con)
  {
  die('Could not connect: ' . mysqli_error());
  }

mysqli_close($con);
?>
```

OO mysqli


php.net/manual/en/function.mysqli-connect.php
Εφαρμογές
Save to Mendeley
H.P.
WEATHER
Banks
TEI
Google
php
Downloads
Documentation
Get Involv
mysqli_connect
(PHP 5)
mysqli_connect — Alias of mysqli::__construct()
Description

OO mysqli connection

```
<?php
$mysqli = new mysqli('localhost', 'my_user', 'my_password', 'my_db');
/*
 * This is the "official" OO way to do it,
 * BUT $connect_error was broken until PHP 5.2.9 and 5.3.0.
 */
if ($mysqli->connect_error) {
    die('Connect Error (' . $mysqli->connect_errno . ') '
            . $mysqli->connect_error);
}
/*
 * Use this instead of $connect_error if you need to ensure
 * compatibility with PHP versions prior to 5.2.9 and 5.3.0.
 */
if (mysqli_connect_error()) {
    die('Connect Error (' . mysqli_connect_errno() . ') '
            . mysqli_connect_error());
}
echo 'Success... ' . $mysqli->host_info . "\n";
$mysqli->close();
?>
```

OO mysqli query

```
<?php
/* Create table doesn't return a resultset */
if ($mysqli->query("CREATE TEMPORARY TABLE myCity LIKE City") === TRUE) {
    printf("Table myCity successfully created.\n");
}

/* Select queries return a resultset */
if ($result = $mysqli->query("SELECT Name FROM City LIMIT 10")) {
    printf("Select returned %d rows.\n", $result->num_rows);

    /* free result set */
    $result->close();
}
?>
```

- Mysqli class: Αποθηκεύει πληροφορίες για το connection.
- Mysqli_stmt: Αποθηκεύει πληροφορίες για μια εντολή. (θα την δούμε σε λίγο).
- mysqli_result class: Αποθηκεύει πληροφορίες για αποτελέσματα query.

SQL Injections

- URL: http://host/path/showcategory.php?category_id=2

showcategory.php

```
…
<?php
$sql="select * from product where Category=                    2                    ";

if(! ($result = $mysqli->query($sql))) {
          print "Error: ". $mysqli->error;
} else {
          print "<p>Category Products:</p>\n<UL>";
          while ($row = $result->fetch_array()) {
                    print "<LI>{$row['Title']}: {$row['Description']}</LI>\n";
          }
          print "</UL>\n";
}?>
```

Τελικά θα εκτελεστεί το query:
select * from products where category_id=2

Αποτέλεσμα


```
<p>Category Products:</p>
<UL><LI>Unix in a Nutshell, Fourth Edition: Unix in a Nutshell is the standard desktop reference, without question. (Manpages come in a close second.) With a clean layout and superior command tables available at a glance, O'Reilly's third edition of Nutshell is an essential to own.</LI>
<LI>Windows 7: The Missing Manual: In early reviews, geeks raved about Windows 7. But if you're an ordinary mortal, learning what this new system is all about will be challenging. Fear not: David Pogue's Windows 7: The Missing Manual comes to the rescue. Like its predecessors, this book illuminates its subject with reader-friendly insight, plenty of wit, and hardnosed objectivity for beginners as well as veteran PC users. </LI>
<LI>Understanding the Linux Kernel, Third Edition: In order to thoroughly understand what makes Linux tick and why it works so well on a wide variety of systems, you need to delve deep into the heart of the kernel. The kernel handles all interactions between the CPU and the external world, and determines which programs will share processor time, in what order. It manages limited memory so well that hundreds of processes can share the system efficiently, and expertly organizes data transfers so that the CPU isn't kept waiting any longer than nece</LI>
</UL>
</body>
</html>
```

SQL Injections

- URL: http://host/path/showcategory.php?category_id=9+or+1%3D1

showcategory.php

```
…
<?php
$sql="select * from product where Category=9 or 1=1";

if(! ($result = $mysqli->query($sql))) {
        print "Error: ". $mysqli->error;
} else {
        print "<p>Category Products:</p>\n<UL>";
        while ($row = $result->fetch_array()) {
                print "<LI>{$row['Title']}: {$row['Description']}</LI>\n";
        }
        print "</UL>\n";
}
?>
```

Τελικά θα εκτελεστεί το query:
select * from product where Category=9 or 1=1
Θα δείξει στον χρήστη όλο τον πίνακα Products

SQL Injections – 1ο βήμα λύσης

- URL: http://host/path/showcategory.php?category_id=9+or+1%3D1

showcategory.php

```
…
<?php
$sql="select * from product where Category='9 or 1=1'";

if(! ($result = $mysqli->query($sql))) {
        print "Error: ". $mysqli->error;
} else {
        print "<p>Category Products:</p>\n<UL>";
        while ($row = $result->fetch_array()) {
                print "<LI>{$row['Title']}: {$row['Description']}</LI>\n";
        }
        print "</UL>\n";
}
?>
```

Τελικά θα εκτελεστεί το query:

```
select * from products where category_id='9 or 1=1'
```

SQL Injections – 1ο βήμα λύσης

- Το 1ο βήμα όμως μπορεί να παρακαμφθεί από τον hacker
  http://host/path/showcategory.php?category_id=9'+or+'1' %3D'1

showcategory.php

```
…
<?php
$sql="select * from product where Category='        9' or '1'='1        '";

if(! ($result = $mysqli->query($sql))) {
        print "Error: ". $mysqli->error;
} else {
        print "<p>Category Products:</p>\n<UL>";
        while ($row = $result->fetch_array()) {
                print "<LI>{$row['Title']}: {$row['Description']}</LI>\n";
        }
        print "</UL>\n";
}
?>
```

Τελικά θα εκτελεστεί το query:

```
select * from products where category_id='9' or '1'='1'
```

SQL Injections – 2ο βήμα λύσης

- ΠΑΝΤΑ πρέπει να ελέγχονται τα δεδομένα του χρήστη και ότι πρόκειται να χρησιμοποιηθεί σε βάση να γίνεται "escape"
  http://host/path/showcategory.php?category_id=9'+or+'1'='1

showcategory.php

```
…
<?php
$cat_id = 9\' or \'1\'=\'1 ;
$sql="select * from product where Category='$cat_id'";

if(! ($result = $mysqli->query($sql))) {
        print "Error: ". $mysqli->error;
} else {
        print "<p>Category Products:</p>\n<UL>";
        while ($row = $result->fetch_array()) {
                print "<LI>{$row['Title']}: {$row['Description']}</LI>\n";
        }

}
?>
```

Τελικά θα εκτελεστεί το query:

```
select * from product where Category='9\' or \'1\'=\'1'
```

Prepared Statements

1. Είναι ΑΣΦΑΛΗ
2. έχουν καλύτερη απόδοση
3. Είναι ευδιάκριτη η εμφάνισή/μορφή τους

- Ετοιμάζω ένα ερώτημα με την μορφή:
  - select * from Users where username=? and pass=?
- Κατά την εκτέλεσή του πρέπει να δώσω ορίσματα τις 2 τιμές που αντιστοιχούν στα ?.
  - πχ: ($_REQUEST['username'], $_REQUEST['passwd'])

Prepared Statements

- http://host/path/showcategory.php?category_id=9'+or+'1'='1

showcategory.php

```
…
<?php
$cat_id = $_REQUEST['category_id'];
$sql="select * from product where Category=?";
if (! $stmt = $mysqli->prepare($sql)) {
        echo "Error: " . $mysqli->error;
}
$stmt->bind_param("i",$cat_id);
If( ! $stmt->execute() ) {
        echo "Error: " . $mysqli->error;
}
$result = $stmt->get_result();
print "Category Products:<UL>";
while ($row = $result->fetch_array()) {
        print "<LI>{$row['Title']}: {$row['Description']}</LI>\n";
}
print "</UL>";
?>
```

Prepared Statements - Example

- Μπορώ να εκτελέσω το ίδιο prepared statement πολλές φορές με διαφορετικά ορίσματα

showcategory.php

```
…
$stmt = $mysqli->prepare("SELECT * from category order by Name");
$stmt_prods = $mysqli->prepare("SELECT * from product where Category=?");
print_r($stmt_prods);
$stmt->execute();
$result = $stmt->get_result();
while( $cat = $result->fetch_assoc()) {
        print "<LI>Category {$cat['Name']}:</LI>\n<UL>";
        $stmt_prods->bind_param("i",$cat['ID']);
        $stmt_prods->execute();
        $result_products = $stmt_prods->get_result();
        while ($row = $result_products->fetch_assoc()) {
                print "<LI>{$row['Title']}</LI>\n";
        }
        print "</UL>\n";
}
```

για να εκτελεστεί η get_result πρέπει να είναι εγκατεστημένος ο driver mysqlnd.

Tables of the previous example

Category

ID	Name
1	Programming Languages
2	Operating Systems
3	Databases
4	Networks
5	Web

Product

ID	Title	...	Price	Category
1	Sams Teach Yourself SQL in 10 Minutes (3rd Edition)	Sam	15	3
2	Fundamentals of Database Systems	Thi	30	3
3	Database Systems: The Complete Book	Cle	35	3
4	Java In A Nutshell, 5th Edition	Wit	30	1
5	Essential C# 4.0	Ess	40	1
6	PHP and MySQL Web Development	The	35	1
7	Unix in a Nutshell, Fourth Edition	Uni	25	2
8	Windows 7: The Missing Manual	In	25	2
9	Understanding the Linux Kernel, Third Edition	In	30	2
10	TCP/IP Illustrated, Vol. 1: The Protocols	TCP	50	4
11	CCNA: Cisco Certified Network Associate Study Guide	Cis	50	4
12	Network Security Essentials: Applications and Standards (4th Editio	Wil	60	4
13	Learning Web Design: A Beginner's Guide to (X)HTML, StyleSheets, an	Eve	40	5
14	Beginning Web Programming with HTML, XHTML, and CSS	Thi	35	5
15	Programming the World Wide Web	Pro	50	5

html output


```
<LI>Category Databases:</LI>
<UL><LI>Sams Teach Yourself SQL in 10 Minutes (3rd Edition)</LI>
<LI>Fundamentals of Database Systems</LI>
<LI>Database Systems: The Complete Book</LI>
</UL>
<LI>Category Networks:</LI>
<UL><LI>TCP/IP Illustrated, Vol. 1: The Protocols </LI>
<LI>CCNA: Cisco Certified Network Associate Study Guide</LI>
<LI>Network Security Essentials: Applications and Standards (4th Edition)</LI>
</UL>
<LI>Category Operating Systems:</LI>
<UL><LI>Unix in a Nutshell, Fourth Edition</LI>
<LI>Windows 7: The Missing Manual</LI>
<LI>Understanding the Linux Kernel, Third Edition</LI>
</UL>
<LI>Category Programming Languages:</LI>
<UL><LI>Java In A Nutshell, 5th Edition</LI>
<LI>Essential C# 4.0</LI>
<LI>PHP and MySQL Web Development </LI>
</UL>
<LI>Category Web :</LI>
<UL><LI>Learning Web Design: A Beginner's Guide to (X)HTML, StyleSheets, and Web Graphics</LI>
<LI>Beginning Web Programming with HTML, XHTML, and CSS</LI>
<LI>Programming the World Wide Web</LI>
</UL>
```

layout output

asidirop.it.teithe.gr/~asidirop/tmp/a.html - Google Chrome

asidirop.it.teithe.gr/~asidirop/tmp/a.html

Apps | H.P. | WEATHER | Banks | TEI | GoogleMore | Academic | Asid | » | Other bookmarks

- Category Databases:
  - Sams Teach Yourself SQL in 10 Minutes (3rd Edition)
  - Fundamentals of Database Systems
  - Database Systems: The Complete Book
- Category Networks:
  - TCP/IP Illustrated, Vol. 1: The Protocols
  - CCNA: Cisco Certified Network Associate Study Guide
  - Network Security Essentials: Applications and Standards (4th Edition)
- Category Operating Systems:
  - Unix in a Nutshell, Fourth Edition
  - Windows 7: The Missing Manual
  - Understanding the Linux Kernel, Third Edition
- Category Programming Languages:
  - Java In A Nutshell, 5th Edition
  - Essential C# 4.0
  - PHP and MySQL Web Development
- Category Web :
  - Learning Web Design: A Beginner's Guide to (X)HTML, StyleSheets, and Web Graphics
  - Beginning Web Programming with HTML, XHTML, and CSS
  - Programming the World Wide Web

bookstore | Show all downloads...

Bind_param uses call by reference


```
$stmt = $mysqli->prepare("INSERT INTO MyGuests
   (firstname, lastname, email) VALUES (?, ?, ?)");

$stmt->bind_param("sss", $firstname, $lastname, $email);

// insert a row
$firstname = "John";
$lastname = "Doe";
$email = "john@example.com";
$stmt->execute();

// insert another row
$firstname = "Mary";
$lastname = "Moe";
$email = "mary@example.com";
$stmt->execute();
```

Χρησιμοποιούμε την bind_param μόνο μια φορά. Μετά απλά αλλάζουμε τις τιμές στις μεταβλητές.

Mysqli object properties


```
mysqli Object
(
    [affected_rows] => 0
    [client_info] => 5.1.73
    [client_version] => 50173
    [connect_errno] => 0
    [connect_error] =>
    [errno] => 0
    [error] =>
    [field_count] => 0
    [host_info] => Localhost via UNIX socket
    [info] =>
    [insert_id] => 0
    [server_info] => 5.1.73-1+deb6u1-log
    [server_version] => 50173
    [sqlstate] => 00000
    [protocol_version] => 10
    [thread_id] => 54
    [warning_count] => 0
)
```

Mysqli_stmt object properties


```
mysqli_stmt Object
(
    [affected_rows] => 0
    [insert_id] => 0
    [num_rows] => 0
    [param_count] => 1
    [field_count] => 5
    [errno] => 0
    [error] =>
    [sqlstate] => 00000
    [id] => 2
)
```

Mysqli_stmt object methods

- Execute
- bind_param
- get_result
- bind_result
- Etc.

Mysqli_result object methods & properties

- mysqli_result::$current_field — Get current field offset of a result pointer
- mysqli_result::$lengths — Returns the lengths of the columns of the current row in the result set
- mysqli_result::$num_rows — Gets the number of rows in a result
- mysqli_result::$field_count — Get the number of fields in a result
- mysqli_result::data_seek — Adjusts the result pointer to an arbitrary row in the result
- mysqli_result::fetch_all — Fetches all result rows as an associative array, a numeric array, or both
- **mysqli_result::fetch_array** — Fetch a result row as an associative, a numeric array, or both
- **mysqli_result::fetch_assoc** — Fetch a result row as an associative array
- mysqli_result::fetch_field_direct — Fetch meta-data for a single field
- mysqli_result::fetch_field — Returns the next field in the result set
- mysqli_result::fetch_fields — Returns an array of objects representing the fields in a result set
- mysqli_result::fetch_object — Returns the current row of a result set as an object
- **mysqli_result::fetch_row** — Get a result row as an enumerated array
- mysqli_result::field_seek — Set result pointer to a specified field offset
- mysqli_result::free — Frees the memory associated with a result

PDO

- PDO - PHP Data Object.
- A set of PHP extensions that provide a core PDO class and database specific drivers.
- Provides a vendor-neutral lightweight data-access abstraction layer.
- Focus on data access abstraction rather than database abstraction.
- PDO requires the new object oriented features in the core of PHP 5, therefore it will not run with earlier versions of PHP.

Installing PDO

- PDO is divided into two components
  - CORE (provides the interface)
  - DRIVERS (access to particular database)
    - Ex. pdo_mysql
- The CORE is enabled by default, drivers with the exception of pdo_sqlite are not.

PDO Drivers

Database name	Driver name
Cubrid	PDO_CUBRID
FreeTDS / Microsoft SQL Server / Sybase	PDO_DBLIB
Firebird/Interbase 6	PDO_FIREBIRD
IBM DB2	PDO_IBM
IBM Informix Dynamic Server	PDO_INFORMIX
MySQL 3.x/4.x/5.x	PDO_MYSQL
Oracle Call Interface	PDO_OCI
ODBC v3 (IBM DB2, unixODBC and win32 ODBC)	PDO_ODBC
PostgreSQL	PDO_PGSQL
SQLite 3 and SQLite 2	PDO_SQLITE
Microsoft SQL Server / SQL Azure	PDO_SQLSRV
4D	PDO_4D

Mysql PDO Connection (try/catch)


```
<?php
try {
      $dbhost = 'localhost';
      $dbuser = 'root';
      $dbpass = '123456';
      $dbh = new PDO("mysql:host=$dbhost;dbname=
hr", $dbuser, $dbpass);
} catch (PDOException $e){
      echo "Error!: " . $e->getMessage() .
            "<br/>";
      die();
}
?>
```

Postgress PDO Connection (try/catch)


```
<?php
try {
      $dbhost = 'localhost';
      $dbuser = 'root';
      $dbpass = '123456';
      $dbh = new PDO("pgsql:host=$dbhost;dbname=
hr", $dbuser, $dbpass);
} catch (PDOException $e){
      echo "Error!: " . $e->getMessage() .
            "<br/>";
      die();
}
?>
```

What if the Connection Fails?

- As is the case with most native PHP objects, instantiation failure lead to an exception being thrown.

```
try {
    $db = new PDO(…);
} catch (PDOException $e) {
    echo $e->getMessage();
}
```

Persistent Connections

- Connecting to complex databases like Oracle is a slow process, it would be nice to re-use a previously opened connection.

```
$opt = array(PDO::ATTR_PERSISTENT => TRUE) ;
try {
      $db = new PDO("dsn",  $l,  $p,  $opt);
} catch (PDOException $e) {
      echo $e->getMessage();
}
```

Let's Run Some Queries

- Query execution in PDO can be done in two ways
  - Prepared Statements (recommended for speed & security)
  - Direct Execution

Direct Query Execution

- Queries that modify information need to be run via exec() method.

```
$db = new PDO("DSN");

$db->exec("INSERT INTO foo (id) VALUES('bar')");

$db->exec("UPDATE foo SET id='bar'");
```

- The return value is the number of rows affected by the operation or FALSE on error.

Direct Query Execution Cont.

- In some cases "change" queries may not affect any rows and will return 0, so type-sensitive compare is essential in avoiding false positives!

```
$res = $db->exec("UPDATE foo SET id='bar'");

if (!$res) // Wrong

if ($res !== FALSE) // Correct
```

Retrieving Error Information

- PDO Provides 2 methods of getting error information:
  - errorCode() – SQLSTATE error code
    - Ex. 42000 == Syntax Error
  - errorInfo() – Detailed error information
    - Ex. array(

      [0] => 42000,

      [1] => 1064

      [2] => You have an error in your SQL syntax; …

)

Better Error Handling

- It stands to reason that being an OO extension PDO would allow error handling via Exceptions.

```
$db->setAttribute(
        PDO::ATTR_ERRMODE,
        PDO::ERRMODE_EXCEPTION
);
```

- Now any query failure will throw an Exception.

Direct Execution Cont.

- When executing queries that retrieve information the query() method needs to be used.

```
$res = $db->query("SELECT * FROM foo");
// $res == PDOStatement Object
```

- On error FALSE is returned

Fetch Query Results

- Perhaps one of the biggest features of PDO is its flexibility when it comes to how data is to be fetched.
  - Array (Numeric or Associated Indexes)
  - Strings (for single column result sets)
  - Objects (stdClass, object of given class or into an existing object)
  - Callback function
  - Lazy fetching
  - Iterators
  - And more!

Array Fetching


```
$res = $db->query("SELECT * FROM foo");
while ($row = $res->fetch(PDO::FETCH_NUM)){
      // $row == array with numeric keys
}

$res = $db->query("SELECT * FROM foo");
while ($row = $res->fetch(PDO::FETCH_ASSOC)){
      // $row == array with associated (string) keys
}

$res = $db->query("SELECT * FROM foo");
while ($row = $res->fetch(PDO::FETCH_BOTH)){
      // $row == array with associated & numeric keys
}
```

Fetch as String

- Many applications need to fetch data contained within just a single column.

```
$u = $db->query("SELECT users WHERE login='login'
AND password='password'");

// fetch(PDO::FETCH_COLUMN)
if ($u->fetchColumn()) {
      // returns a string
      // login OK
} else {
      /* authentication failure */
}
```

Fetch as Standard Object

- You can fetch a row as an instance of stdClass where column name == property name.

```
$res = $db->query("SELECT * FROM foo");

while ($obj = $res->fetch(PDO::FETCH_OBJ)) {
      // $obj == instance of stdClass
}
```

Fetch Into a Class

- PDO allows the result to be fetched into a class type of your choice.

```
$res = $db->query("SELECT * FROM foo");
$res->setFetchMode(
      PDO::FETCH_CLASS,
      "className",
      array('optional'='Constructor Params')
);
while ($obj = $res->fetch()) {
      // $obj == instance of className
}
```

Fetch Into a Class Cont.

- PDO allows the query result to be used to determine the destination class.

```
$res = $db->query("SELECT * FROM foo");
$res->setFetchMode(
      PDO::FETCH_CLASS |
      PDO::FETCH_CLASSTYPE
);
while ($obj = $res->fetch()) {
      // $obj == instance of class who's name is
      // found in the value of the 1st column
}
```

Fetch Into an Object

- PDO even allows retrieval of data into an existing object.

```
$u = new userObject;

$res = $db->query("SELECT * FROM users");
$res->setFetchMode(PDO::FETCH_INTO, $u);

while ($res->fetch()) {
      // will re-populate $u with row values
}
```

Result Iteration

- PDOStatement implements Iterator interface, which allows for a method-less result iteration.

```
$res = $db->query(
      "SELECT * FROM users",  PDO::FETCH_ASSOC
);
foreach ($res as $row) {
      // $row == associated array representing
      // the row's values.
}
```

Lazy Fetching

- Lazy fetches returns a result in a form object, but holds of populating properties until they are actually used.

```
$res = $db->query(
      "SELECT * FROM users",  PDO::FETCH_LAZY
);
foreach ($res as $row) {
    echo $row['name']; // only fetch name column
}
```

fetchAll()

- The fetchAll() allows retrieval of all results from a query right away. (handy for templates)

```
$qry = "SELECT * FROM users";
$res = $db->query($qry)->fetchAll(PDO::FETCH_ASSOC);
// $res == array of all result rows, where each row
// is an associated array.
```

- Can be quite memory intensive for large results sets!

Callback Function

- PDO also provides a fetch mode where each result is processed via a callback function.

```
function draw_message($subject,$email) {
      …
}

$res = $db->query("SELECT * FROM msg");

$res->fetchAll( PDO::FETCH_FUNC, "draw_message");
```

Direct Query Problems

- Query needs to be interpreted on each execution can be quite waste for frequently repeated queries.
- Security issues, un-escaped user input can contain special elements leading to SQL injection.

Prepared Statements

- Compile once, execute as many times as you want.
- Clear separation between structure and input, which prevents SQL injection.
- Often faster then query()/exec() even for single runs.

Prepared Statements in Action


```
$stmt = $db->prepare("SELECT * FROM
users WHERE id=?" );

$stmt->execute(array($_GET['id']));

$stmt->fetch(PDO::FETCH_ASSOC);
```

Bound Parameters

- Prepared statements parameters can be given names and bound to variables.

```
$stmt = $db->prepare(
"INSERT INTO users VALUES(:name,:pass,:mail)");

$stmt->bindParam(':name',$name);
$stmt->bindParam(':pass',$pass);
$stmt->bindParam(':mail',$mail);

$fp = fopen("./users", "r");
while (list($name,$pass,$mail)=fgetcsv($fp,4096))
{
      $stmt->execute();
}
```

Bound Parameters

- Prepared statements parameters can be given names and bound to variables.

```
$stmt = $db->prepare(
"INSERT INTO users VALUES(:name,:pass,:mail)");

foreach (array('name','pass','mail') as $v)
      $stmt->bindParam(':'.$v,$$v);

$fp = fopen("./users", "r");
while (list($name,$pass,$mail)=fgetcsv($fp,4096))
{
      $stmt->execute();
}
```

Bound Parameters

- Prepared statements parameters can be given names and bound to array elements.

```
# $arr = [ 'name'=>'Antonis', 'pass'=>'hello',
'mail'=>'asidirop@nowhere'];
$stmt = $db->prepare(
"INSERT INTO users VALUES(:name,:pass,:mail)");
foreach ($arr as $k=>&$v)
      $stmt->bindParam(':'.$k,$v);

$fp = fopen("./users", "r");
while (list($name,$pass,$mail)=fgetcsv($fp,4096))
{
      $stmt->execute();
}
```

Bound Result Columns

- Result columns can be bound to variables as well.

```
$qry = "SELECT :type, :data FROM images LIMIT 1";
$stmt = $db->prepare($qry);

$stmt->bindColumn(':type',$type);
$stmt->bindColumn(':data',STDOUT,PDO::PARAM_LOB);
$stmt->execute(PDO::FETCH_BOUND);

header("Content-Type: ".$type);
```

Partial Data Retrieval

- In some instances you only want part of the data on the cursor. To properly end the cursor use the closeCursor() method.

```
$res = $db->query("SELECT * FROM users");
foreach ($res as $v) {
      if ($res['name'] == 'end') {
                  $res->closeCursor();
                  break;
      }
}
```

Transactions

- Nearly all PDO drivers talk with transactional DBs, so PDO provides handy methods for this purpose.

```
$db->beginTransaction();
if ($db->exec($qry) === FALSE) {
      $db->rollback();
}
$db->commit();
```

Metadata

- Like most native database interfaces PDO provides means of accessing query metadata.

```
$res = $db->query($qry);

$ncols = $res->columnCount();
for ($i=0; $i < $ncols; $i++) {
      $meta_data = $stmt->getColumnMeta($i);
}
```

getColumnMeta() Result

- native_type – PHP data type
- driver:decl_type - The data type of the column according to the database.
- flags – will return any flags particular to this column in a form of an array.
- name – the name of the column as returned by the database without any normalization.
- len – maximum length of a string column, may not always be available, will be set to -1 if it isn't.
- precision - The numeric precision of this column.
- pdo_type - The column type according to PDO as one of the PDO_PARAM constants.

lastInsertId()

- Many databases have unique identifier assigned to each newly inserted row. PDO provides access to this value via lastInsertId() method.

```
if ($db->exec("INSERT INTO …")) {
      $id = $db->lastInsertId();
}
```

- Can take optional sequence name as parameter.
  - Useful for PostgreSQL

Connection Information

- Some connection information can be obtained via the getAttribute() PDO method.

```
$db->getAttribute(PDO::ATTR_SERVER_VERSION);
// Database Server Version
$db->getAttribute(PDO::ATTR_CLIENT_VERSION);
// Client Library Server Version
$db->getAttribute(PDO::ATTR_SERVER_INFO);
// Misc Server information
$db->getAttribute(PDO::ATTR_CONNECTION_STATUS);
// Connection Status
```

Extending PDO


```
class DB extends PDO {
      function query($qry, $mode=NULL) {
            $res = parent::query($qry, $mode);
            if (!$res) {
               var_dump($qry, $this->errorInfo());
               return null;
            } else {
                  return $res;
            }
      }
}
```