CSC 375
Database Management Systems

Haidar Harmanani
Department of Computer Science and Mathematics
Lebanese American University
Byblos, 1401 2010 Lebanon

Introduction

- PHP
  - PHP Hypertext Preprocessor
  - Originally called "Personal Home Page Tools"
  - Popular server-side scripting technology
  - Open-source
    - Anyone may view, modify and redistribute source code
    - Supported freely by community
  - Platform independent

PHP

- Basic application
  - Scripting delimiters
    - <? php ?>
    - Must enclose all script code
  - Variables preceded by $ symbol
    - Case-sensitive
  - End statements with semicolon
  - Comments
    - // for single line
    - /* */ for multiline
  - Filenames end with .php by convention

LAU Computer Science

What do You Need?

- Our server supports PHP
  - You don't need to do anything special!
  - You don't need to compile anything or install any extra tools!
  - Create some .php files in your web directory - and the server will parse them for you.
- Most servers support PHP
  - Download PHP for free here: http://www.php.net/downloads.php
  - Download MySQL for free here: http://www.mysql.com/downloads/index.html
  - Download Apache for free here: http://httpd.apache.org/download.cgi

LAU Computer Science

Basic PHP syntax

A PHP scripting block always starts with <?php and ends with ?>. A PHP scripting block can be placed anywhere in an HTML document.

```
<html>
<!-- hello.php COMP519 -->
<head><title>Hello World</title></head>
<body>
 <p>This is going to be ignored by the PHP interpreter.</p>

  <?php echo '<p>While this is going to be parsed.</p>'; ?>

 <p>This will also be ignored by PHP.</p>

  <?php print('<p>Hello and welcome to <i>my</i> page!</p>');
   ?>

 <?php
  //This is a comment
 /*
  This is
  a comment
  block
 */   ?>
</body>
</html>
```

view the output page

print and echo for output

a semicolon (;) at the end of each statement (can be omitted at the end of block/file)

// for a single-line comment

/* and */ for a large comment block.

The server executes the print and echo statements, substitutes output.

PHP

- Variables
  - Can have different types at different times
  - Variable names inside strings replaced by their value
  - Type conversions
    - `settype` function
    - Type casting
  - Concatenation operator
    - `.` (period)
    - Combine strings

Scalars

All variables in PHP start with a $ sign symbol.

A variable's type is determined by the context in which that variable is used (i.e. there is no strong-typing in PHP).

```
<html><head></head>
<!-- scalars.php COMP519 -->
<body> <p>
<?php
$foo = True; if ($foo) echo "It is TRUE! <br /> \n";
$txt='1234'; echo "$txt <br /> \n";
$a = 1234; echo "$a <br /> \n";
$a = -123;
echo "$a <br /> \n";
$a = 1.234;
echo "$a <br /> \n";
$a = 1.2e3;
echo "$a <br /> \n";
$a = 7E-10;
echo "$a <br /> \n";
echo 'Arnold once said: "I\'ll be back"', "<br /> \n";
$beer = 'Heineken';
echo "$beer's taste is great <br /> \n";
$str = <<<EOD
Example of string
spanning multiple lines
using “heredoc” syntax.
EOD;
echo $str;
?> </p>
</body>
</html>
```

Four scalar types:

boolean
TRUE or FALSE

integer,
just numbers

float
float point numbers

string
single quoted
double quoted

view the output page

Constants

A constant is an identifier (name) for a simple value. A constant is case-sensitive by default. By convention, constant identifiers are always uppercase.

```
<?php

// Valid constant names
define("FOO",    "something");
define("FOO2",    "something else");
define("FOO_BAR", "something more");

// Invalid constant names
define("2FOO",    "something");

// This is valid, but should be avoided:
// PHP may one day provide a “magical” constant
// that will break your script
define("__FOO__", "something");

?>
```

You can access constants anywhere in your script without regard to scope.

Operators

- Arithmetic Operators: +, -, *,/ , %, ++, --
- Assignment Operators: =, +=, -=, *=, /=, %=

Example	Is the same as
x+=y	x=x+y
x-=y	x=x-y
x*=y	x=x*y
x/=y	x=x/y
x%=y	x=x%y

- Comparison Operators: ==, !=, >, <, >=, <=
- Logical Operators: &&, ||, !
- String Operators: . , .=

```
$a = "Hello ";
$b = $a . "World!"; // now $b contains "Hello World!"

$a = "Hello ";
$a .= "World!";
```

Conditionals: if else

- Can execute a set of code depending on a condition

```
<html><head></head>
<!-- if-cond.php COMP519 -->
<body>
<?php
$d=date("D");
if ($d=="Fri")
    echo "Have a nice weekend! <br/>";
else
    echo "Have a nice day! <br/>";
$x=10;
if ($x==10)
{
    echo "Hello<br />";
    echo "Good morning<br />";
}
?>
</body>
</html>
```

```
if (condition)
code to be executed if condition is
true;
else
code to be executed if condition is
false;
```

date() is a built-in function that can be called with many different parameters to return the date (and/or local time) in various formats

In this case we get a three letter string for the day of the week.

view the output page

Conditionals: switch

- Can select one of many sets of lines to execute

```
<html><head></head>
<body>
<!-- switch-cond.php COMP519 -->
<?php
$x = rand(1,5);  // random integer
echo "x = $x <br/><br/>";
switch ($x)
{
case 1:
  echo "Number 1";
  break;
case 2:
  echo "Number 2";
  break;
case 3:
  echo "Number 3";
  break;
default:
  echo "No number between 1 and 3";
}
?>
</body>
</html>
```

```
switch (expression)
{
case label1:
  code to be executed if expression =
label1;
  break;
case label2:
  code to be executed if expression =
label2;
  break;
default:
  code to be executed
  if expression is different
  from both label1 and label2;
}
```

view the output page

Looping: while and do-while

Can loop depending on a condition

```
<html><head></head>
<body>
<?php
$i=1;
while($i <= 5)
{
  echo "The number is $i <br />";
  $i++;
}
?>
</body>
</html>
```

loops through a block of code if and as long as a specified condition is true

view the output page

```
<html><head></head>
<body>
<?php
$i=0;
do
{
  $i++;
  echo "The number is $i <br />";
}
while($i <= 10);
?>
</body>
</html>
```

loops through a block of code once, and then repeats the loop as long as a special condition is true

view the output page

Looping: foreach

```
<?php
foreach (array_expression as $value)
    statement
foreach (array_expression as $key => $value)
    statement?>
```

- The first loops over the array given by *array_expression.*
  - *On each loop, the value of the current element is assigned to $value and the internal array pointer is advanced by one (so on the next loop, you'll be looking at the next element).*
- *The second form does the same thing, except that the current element's key will be assigned to the variable $key on each loop.*

Looping: for and foreach

Can loop depending on a "counter"

```
<?php
for ($i=1; $i<=5; $i++)
{
echo "Hello World!<br />";
}
?>
```

loops through a block of code a specified number of times

view the output page

```
<?php
$a_array = array(1, 2, 3, 4);
foreach ($a_array as $value)
{
  $value = $value * 2;
  echo “$value <br/> \n”;
}
?>
```

```
<?php
$a_array=array("a","b","c");
foreach ($a_array as $key=>$value)
{
  echo $key." = ".$value."\n";
}
?>
```

loops through a block of code for each element in an array

Looping: foreach

```
<?php
$arr = array(1, 2, 3, 4);
foreach ($arr as &$value) {
    $value = $value * 2;
}
// $arr is now array(2, 4, 6, 8)
unset($value); // break the reference with the last element
?>
```

modify array's elements by preceding *$value with &*.
This will assign reference instead of copying the value

Variable Scope

The scope of a variable is the context within which it is defined.

```
<?php
$a = 1; /* global scope */
function Test()
{
  echo $a;
 /* reference to local scope variable */
}
Test();
?>
```

The scope is local within functions.

```
<?php
$a = 1;
$b = 2;
function Sum()
{
  global $a, $b;
  $b = $a + $b;
}
Sum();
echo $b;
?>
```

global

refers to its global version.

view the output page

```
<?php
function Test()
{
  static $a = 0;
  echo $a;
  $a++;
}
Test1();
Test1();
Test1();
?>
```

static

does not lose its value.

Arrays

An array in PHP is actually an ordered map. A map is a type that maps values to keys.

```
<?php
$arr = array("foo" => "bar", 12 => true);
echo $arr["foo"]; // bar
echo $arr[12];    // 1
?>
```

array() = creates arrays

key = either an integer or a string.

value = any PHP type.

```
<?php
array(5 => 43, 32, 56, "b" => 12);
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

if no key, the maximum of the integer indices + 1.

if an existing key, its value will be overwritten.

```
<?php
$arr = array(5 => 1, 12 => 2);
$arr[] = 56;    // the same as $arr[13] = 56;
$arr["x"] = 42; // adds a new element
unset($arr[5]); // removes the element
unset($arr);    // deletes the whole array
$a = array(1 => 'one', 2 => 'two', 3 => 'three');
unset($a[2]);
$b = array_values($a);
?>
```

view the output page

can set values in an array

unset() removes a key/value pair

array_values() makes reindex effect (indexing numerically)

*Find more on arrays

User Defined Functions

Can define a function using syntax such as the following:

```
<?php
function foo($arg_1, $arg_2, /* ..., */ $arg_n)
{
  echo "Example function.\n";
  return $retval;
}
?>
```

Can also define conditional functions, functions within functions, and recursive functions.

Can return a value of any type

```
<?php
function square($num)
{
  return $num * $num;
}
echo square(4);
?>
```

```
<?php
function small_numbers()
{
  return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
echo $zero, $one, $two;
?>
```

```
<?php
function takes_array($input)
{
  echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
 takes_array(array(1,2));
?>
```

view the output page

Files

- PHP can:
  - Deal with any files on the server
  - Deal with any files on the Internet, using either http or ftp
  - Instead of file handles, PHP associates a variable with a file, called the *file variable* (for program reference)

File Open

The `fopen("file_name","mode")` function is used to open files in PHP.

r	Read only.	r+	Read/Write.
w	Write only.	w+	Read/Write.
a	Append.	a+	Read/Append.
x	Create and open for write only.	x+	Create and open for read/write.

```
<?php
$fh=fopen("welcome.txt","r");
?>
```

For w, and a, if no file exists, it tries to create it (use with caution, i.e. check that this is the case, otherwise you'll overwrite an existing file).

For x if a file exists, it returns an error.

```
<?php
if (!($fh=fopen("welcome.txt","r")))
exit("Unable to open file!");
?>
```

If the fopen() function is unable to open the specified file, it returns 0 (false).

Examples: Working with files

fclose() **closes a file.**

fgetc() **reads a single character**

fwrite(), fputs **writes a string with and without \n**

feof() **determines if the end is true.**

fgets() **reads a line of data**

file() **reads entire file into an array**

```
<?php
$myFile = "welcome.txt";
if (!($fh=fopen($myFile,'r')))
exit("Unable to open file.");
while (!feof($fh))
{
$x=fgetc($fh);
echo $x;
}
fclose($fh);
?>
```

view the output page

```
<?php
$myFile = "welcome.txt";
$fh = fopen($myFile, 'r');
$theData = fgets($fh);
fclose($fh);
echo $theData;
?>
```

view the output page

```
<?php
$lines = file('welcome.txt');
foreach ($lines as $l_num => $line)
{
 echo "Line #{$l_num}:“ .$line.”<br/>”;
}
?>
```

view the output page

```
<?php
$myFile = "testFile.txt";
$fh = fopen($myFile, 'a') or die("can't open file");
$stringData = "New Stuff 1\n";
fwrite($fh, $stringData);
$stringData = "New Stuff 2\n";
fwrite($fh, $stringData);
fclose($fh);
?>
```

view the output page

Including Files

The include() statement includes and evaluates the specified file.

```
vars.php
<?php

$color = 'green';
$fruit = 'apple';

?>
test.php
<?php

echo "A $color $fruit"; // A

include 'vars.php';

echo "A $color $fruit"; // A green apple

?>
```

view the output page

```
<?php

function foo()
{
  global $color;

  include ('vars.php‘);

  echo "A $color $fruit";
}

/* vars.php is in the scope of foo() so     *
 * $fruit is NOT available outside of this  *
 * scope.  $color is because we declared it *
 * as global.                               */

foo();                  // A green apple
echo "A $color $fruit";  // A green

?>
```

view the output page

The scope of variables in “included” files depends on where the “include” file is added!

You can use the include_once, require, and require_once statements in similar ways.

PHP Information

The `phpinfo()` function is used to output PHP information about the version installed on the server, parameters selected when installed, etc.

```
<html><head></head>
<!– info.php COMP519
<body>
<?php
// Show all PHP information
phpinfo();
?>
<?php
// Show only the general information
phpinfo(INFO_GENERAL);
?>
</body>
</html>
```

view the output page

INFO_GENERAL	The configuration line, php.ini location, build date, Web Server, System and more
INFO_CREDITS	PHP 4 credits
INFO_CONFIGURATION	Local and master values for php directives
INFO_MODULES	Loaded modules
INFO_ENVIRONMENT	Environment variable information
INFO_VARIABLES	All predefined variables from EGPCS
INFO_LICENSE	PHP license information
INFO_ALL	Shows all of the above (default)

Server Variables

The `$_SERVER` is a reserved variable that contains all server information.

```
<html><head></head>
<body>

<?php
echo "Referer: " . $_SERVER["HTTP_REFERER"] . "<br />";
echo "Browser: " . $_SERVER["HTTP_USER_AGENT"] . "<br
/>";
echo "User's IP address: " . $_SERVER["REMOTE_ADDR"];
?>
</body>
</html>
```

view the output page

The $_SERVER is a super global variable, i.e. it's available in all scopes of a PHP script.

Getting Time and Date

`date() and time ()` **formats a time or a date.**

```
<?php
//Prints something like: Monday
echo date("l");

//Like: Monday 15th of January 2003 05:51:38 AM
echo date("l dS of F Y h:i:s A");

//Like: Monday the 15th
echo date("l \\t\h\e jS");
?>
```

view the output page

date() returns a string formatted according to the specified format.

```
<?php
$nextWeek = time() + (7 * 24 * 60 * 60);
                // 7 days; 24 hours; 60 mins; 60secs
echo 'Now:     '. date('Y-m-d') ."\n";
echo 'Next Week: '. date('Y-m-d', $nextWeek) ."\n";
?>
```

view the output page

time() returns current Unix timestamp

*Here is more on date/time formats: http://uk.php.net/manual/en/function.date.php

Forms Handling

Any form element is automatically available via one of the built-in PHP variables.

```
<html>
<-- form.html COMP519 -->
<body>
<form action="welcome.php" method="POST">
Enter your name: <input type="text" name="name" /> <br/>
Enter your age: <input type="text" name="age" /> <br/>
<input type="submit" /> <input type="reset" />
</form>
</body>
</html>
```

```
<html>
<!–- welcome.php COMP 519 -->
<body>

Welcome <?php echo $_POST["name"].”.”; ?><br />
You are <?php echo $_POST["age"]; ?> years old!

</body>
</html>
```

$_POST
contains all POST data.

$_GET
contains all GET data.

view the output page

Cookies

`setcookie(name,value,expire,path,domain)` **creates cookies.**

```
<?php
setcookie("uname", $_POST["name"], time()+36000);
?>
<html>
<body>
<p>
Dear <?php echo $_POST["name"] ?>, a cookie was set on this
page! The cookie will be active when the client has sent the
cookie back to the server.
</p>
</body>
</html>
```

view the output page

NOTE:
setcookie() must appear BEFORE <html> (or any output) as it's part of the header information sent with the page.

```
<html>
<body>
<?php
if (isset($_COOKIE["uname"]))
echo "Welcome " . $_COOKIE["uname"] . "!<br />";
else
echo "You are not logged in!<br />";
?>
</body>
</html>
```

view the output page

$_COOKIE
contains all COOKIE data.

isset()
finds out if a cookie is set

use the cookie name as a variable

PHP sessions

- Web servers don't keep track of information entered on a page when the client's browser opens another page
  - Difficult to do anything involving the same information across several pages
- Sessions help by maintaining data during a user's visit by storing data that can be accessed from page to page in a site.
- One can use session variables for storing information

Session Tracking

- A session is the time span during which a browser interacts with a particular server
- For session tracking, PHP creates and maintains a session tracking id
- Create the id with a call to `session_start` with no parameters
- Subsequent calls to `session_start` retrieves any session variables that were previously registered in the session

The MySQL Database System

- A free, efficient, widely used SQL implementation
- Available from http://www.mysql.org
- Logging on to MySQL (starting it):

`mysql` [`-h` host] [`-u` username] [database name] [`-p`]

  - Host is the name of the MySQL server
    - Default is the user's machine
  - Username is that of the database
    - Default is the name used to log into the system
  - The given database name becomes the "focus" of MySQL
  - If you want to access an existing database, but it was not named in the `mysql` command, you must choose it for focus

```
use cars;
```

  - Response is: `Database changed`

Logging into mySQL Server

- You can log into our mySQL server from Linux by typing in the prompt

```
bash-2.05b$ mysql -h mysql <user name>
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 209201 to server version: 5.0.22

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

From here you can create and drop tables, and modify the data in your tables. But first you must specify which database on the server you want to use.

```
mysql> use <database>;
```

```
Database changed
```

Creating a Table

Next, you can create the table you will be using for the upcoming project. For example,

```
mysql> CREATE TABLE students(
    -> num INT NOT NULL AUTO_INCREMENT,
    -> f_name VARCHAR(48),
    -> l_name VARCHAR(48),
    -> student_id INT,
    -> email VARCHAR(48),
    -> PRIMARY KEY(num));
```

Hit **Enter** after each line (if you want). MySQL doesn't try to interpret the command itself until it sees a semicolon (;)

```
Query OK, 0 rows affected (0.02 sec)
```

*If the server gives you a big ERROR, just try again from the top!

Viewing The Table Structure

Using `DESCRIBE` you can see the structure of a table

```
mysql> DESCRIBE students;
```

```
+------------+-------------+------+-----+---------+----------------+
| Field      | Type        | Null | Key | Default | Extra          |
+------------+-------------+------+-----+---------+----------------+
| num        | int(11)     | NO   | PRI | NULL    | auto_increment |
| f_name     | varchar(48) | YES  |     | NULL    |                |
| l_name     | varchar(48) | YES  |     | NULL    |                |
| student_id | int(11)     | YES  |     | NULL    |                |
| email      | varchar(48) | YES  |     | NULL    |                |
+------------+-------------+------+-----+---------+----------------+
```

Inserting Data

Using `INSERT INTO` you can insert a new row into your table. For example,

```
mysql> INSERT INTO students
    -> VALUES(NULL,'Russell','Martin',396310,'martin@csc.liv.ac.uk');
```

```
Query OK, 1 row affected (0.00 sec)
```

Using SELECT FROM you select some data from a table.

```
mysql> SELECT * FROM students;
```

```
+-----+---------+--------+------------+--------------------+
| num | f_name  | l_name | student_id | email              |
+-----+---------+--------+------------+--------------------+
|   1 | Russell | Martin |     396310 | martin@csc.liv.ac.uk |
+-----+---------+--------+------------+--------------------+
1 row in set (0.00 sec)
```

Inserting Some More Data

You can repeat inserting until all data is entered into the table.

```
mysql> INSERT INTO students
    -> VALUES(NULL,'James','Bond',007,'bond@csc.liv.ac.uk');
Query OK, 1 row affected (0.01 sec)


mysql> SELECT * FROM students;
+-----+---------+--------+------------+---------------------+
| num | f_name  | l_name | student_id | email               |
+-----+---------+--------+------------+---------------------+
|   1 | Russell | Martin |     396310 | martin@csc.liv.ac.uk |
|   2 | James   | Bond   |          7 | bond@csc.liv.ac.uk   |
+-----+---------+--------+------------+---------------------+
2 rows in set (0.00 sec)
```

Altering the Table

- The ALTER TABLE statement is used to add or drop columns in an existing table.

```
mysql> ALTER TABLE students ADD date DATE;
```

```
Query OK, 2 rows affected (0.00 sec)

Records: 2  Duplicates: 0  Warnings: 0
```

```
mysql> SELECT * FROM students;
+-----+---------+--------+------------+---------------------+------+
| num | f_name  | l_name | student_id | email               | date |
+-----+---------+--------+------------+---------------------+------+
|   1 | Russell | Martin |     396310 | martin@csc.liv.ac.uk | NULL |
|   2 | James   | Bond   |          7 | bond@csc.liv.ac.uk   | NULL |
+-----+---------+--------+------------+---------------------+------+
2 rows in set (0.00 sec)
```

Updating the Table

- The UPDATE statement is used to modify the data in a table.

```
mysql> UPDATE students SET date='2007-11-15' WHERE num=1;
```

```
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

```
mysql> SELECT * FROM students;
+-----+---------+--------+------------+---------------------+------------+
| num | f_name  | l_name | student_id | email               | date       |
+-----+---------+--------+------------+---------------------+------------+
|   1 | Russell | Martin |     396310 | martin@csc.liv.ac.uk | 2007-11-15 |
|   2 | James   | Bond   |          7 | bond@csc.liv.ac.uk   | NULL       |
+-----+---------+--------+------------+---------------------+------------+
2 rows in set (0.00 sec)
```

Note that the date is expected in the format "YYYY-MM-DD" and I don't believe this default setting can be changed.

Deleting Some Data

- The DELETE statement is used to delete rows in a table.

```
mysql> DELETE FROM students WHERE l_name='Bond';
```

```
Query OK, 1 row affected (0.00 sec)
```

```
mysql> SELECT * FROM students;
+-----+---------+--------+------------+---------------------+------------+
| num | f_name  | l_name | student_id | email               | date       |
+-----+---------+--------+------------+---------------------+------------+
|   1 | Russell | Martin |     396310 | martin@csc.liv.ac.uk | 2006-11-15 |
+-----+---------+--------+------------+---------------------+------------+
1 row in set (0.00 sec)
```

Other SQL Commands

- SHOW tables;
  - Give a list of tables that have been defined in the database
- ALTER TABLE students DROP email;
  - Drop the "email" column from all records
- DROP TABLE students;
  - Delete the entire "students" table, and its definition (use the DROP command with extreme care!!)
- DELETE FROM students;
  - Removes all rows from the "students" table (so once again, use the DELETE command with great caution), the table definition remains to be used again
- A more useful(?) command is something like

  DELETE FROM students WHERE (num > 5) AND (num <= 10);
  - Selectively deletes students based on their "num" values (for example).
- HELP;
  - Give SQL help
- HELP DROP;
  - Give help on the DROP command, etc.

Using aliases in queries

- Especially long queries might benefit from the SQL capability for using aliases.
- Aliases can be used to refer to a column in GROUP BY, ORDER BY, or HAVING clauses
  - An alias uses the SQL keyword 'AS' to associate a new identifier with a table.
  - It should appear after the table name and before the alias. Once a table is aliased you must use that alias everywhere in the SQL query.
  - SQL doesn't allow references to a column alias in a WHERE clause

Using aliases in queries

- mysql> select p.purchase_id, title FROM purchases AS p, clients AS c, itemlist AS i, books WHERE (p.client_id=c.client_id) AND (l_name='Martin') AND (p.purchase_id=i.purchase_id) AND (i.book_id=books.book_id) ORDER BY p.purchase_id;

```
+-------------+-----------------+
| purchase_id | title           |
+-------------+-----------------+
|           1 | Blackbeard      |
|           1 | Learning SQL    |
|           1 | Abstract Algebra |
|           2 | Rising Sun      |
|           2 | Round the Moon  |
+-------------+-----------------+
5 rows in set (0.00 sec)
```

Searching tables

The SQL "wildcard" character is the % symbol. That is, it can literally represent anything. Using it we can build searches like the following:

```
mysql> SELECT * FROM clients WHERE l_name LIKE '%a%';
+-----------+---------+--------+--------------------------+----------------+----------+
| client_id | f_name  | l_name | address                  | city           | postcode |
+-----------+---------+--------+--------------------------+----------------+----------+
|         1 | Russell | Martin | Dept of Computer Science | Liverpool      | L69 3BX  |
|         4 | Larry   | Vance  | 76 Jarhead Ln            | Liverpool      | L12 4RT  |
|         5 | Paul    | Abbott | 90 Crabtree Pl           | Leamingotn Spa | CV32 7YP |
+-----------+---------+--------+--------------------------+----------------+----------+
3 rows in set (0.00 sec)
```

This above search finds all data that has a letter "a" in the user_id column.

```
mysql> SELECT * FROM clients where l_name LIKE '%an%';
+-----------+--------+--------+---------------+-----------+----------+
| client_id | f_name | l_name | address       | city      | postcode |
+-----------+--------+--------+---------------+-----------+----------+
|         4 | Larry  | Vance  | 76 Jarhead Ln | Liverpool | L12 4RT  |
+-----------+--------+--------+---------------+-----------+----------+
1 row in set (0.00 sec)
```

Searching tables (cont.)

```
mysql> SELECT clients.client_id, f_name, l_name FROM clients NATURAL JOIN purchases
   where l_name LIKE '%a%';
+-----------+---------+--------+
| client_id | f_name  | l_name |
+-----------+---------+--------+
|         1 | Russell | Martin |
|         1 | Russell | Martin |
|         4 | Larry   | Vance  |
|         5 | Paul    | Abbott |
+-----------+---------+--------+
4 rows in set (0.00 sec)


mysql> SELECT clients.client_id, f_name, l_name FROM clients, purchases WHERE (l_name
   LIKE '%a%') AND (clients.client_id=purchases.client_id)
    AND (clients.client_id > 1);
+-----------+--------+--------+
| client_id | f_name | l_name |
+-----------+--------+--------+
|         4 | Larry  | Vance  |
|         5 | Paul   | Abbott |
+-----------+--------+--------+
2 rows in set (0.00 sec)
```

Architectures for Database Access

- *Client-Server Architectures*
  - Client tasks:
    - Provide a way for users to submit queries
    - Run applications that use the results of queries
    - Display results of queries
  - Server tasks:
    - Implement a data manipulation language that can directly access and update the database
- A two-tier system has clients that are connected directly to the server
- Problems with a two-tier system:
  - Because the relative power of clients has grown considerably, we could shift processing to the client, but then keeping all clients current with application updates is difficult1

Architectures for Database Access

- A solution to the problems of two-tier systems is to add a component in the middle
  - Create a three-tier system
- For Web-based database access, the middle tier can run applications (client just gets results)

Architectures for Database Access

- *Microsoft* Access *Architecture*
  - A tool to access any common database structure
  - Use either the Jet database engine, or go through the Open Database Connectivity (ODBC) standard
  - ODBC is an API for a set of objects and methods that are an interface to different databases
- Database vendors provide ODBC drivers for their products
  - The drivers implement the ODBC objects and methods
  - An application can include SQL statements that work for any database for which a driver is available

Architectures for Database Access

- *PHP & Database Access*
  - An API for each specific database system
  - Also convenient for Web access to databases, because PHP is run on the Web server
- *The Java JDBC Architecture*
  - Related to both embedded languages and to ODBC
  - JDBC is a standard protocol that can be implemented as a driver for any database system
  - JDBC allows SQL to be embedded in Java applications, applets, and servlets
  - JDBC has the advantage of portability over embedded SQL
  - A JDBC application will work with any database system for which there is a JDBC driver

Database Access with PHP/MySQL

- To connect PHP to a database, use `mysql_connect`, which can have three parameters:
  - host (default is localhost)
  - Username (default is the username of the PHP script)
  - Password (default is blank, which works if the database does not require a password)
  - `$db = mysql_connect()` is usually checked for failure
- Sever the connection to the database with `mysql_close`

Putting Content into Your Database with PHP

- Connect to the database server and login
  - mysql_connect("host","username","password");
- Choose the database
  - mysql_select_db("database");
- Choose the database
  - mysql_select_db("database");
- Close the connection to the database server
  - mysql_close();

Database Access with PHP/MySQL

- To focus MySQL,

```
mysql_select_db("cars");
```

- Requesting MySQL Operations
  - Call `mysql_query` with a string parameter, which is an SQL command

```
$query = "SELECT * from States";
$result = mysql_query($query);
```

Database Access with PHP/MySQL

- Dealing with the result:
  - Get the number of rows in the result

```
$num_rows = mysql_num_rows($result);
```

  - Get the rows with `mysql_fetch_array`

```
for ($row_num = 0; $row_num < $num_rows; $row_num++) {
  $row = mysql_fetch_array($result);
  print "<p> Result row number" .
            ($row_num + 1) .
            " State_id: ";
  print htmlspecialchars($row["State_id"]);
  print "State: ";
   etc.
```

Getting Content out of Your Database with PHP

Similarly, we can get some information from a database:

- Connect to the server and login, choose a database

```
mysql_connect("host","username","password");
mysql_select_db("database");
```

- Send an SQL query to the server to select data from the database into an array

```
$result=mysql_query("query");
```

- Either, look into a row and a fieldname

```
$num=mysql_numrows($result);
$variable=mysql_result($result,$i,"fieldname"
```

- Or, fetch rows one by one

```
$row=mysql_fetch_array($result);
```

- Close the connection to the database server

```
mysql_close();
```