customerid	name	orderid
1	Julie Smith	2
2	Alan Wong	3
3	Michelle Arthur	1
3	Michelle Arthur	4
4	Melissa Jones	NULL
5	Michael Archer	NULL

customerid	name
4	Melissa Jones
5	Michael Archer

10 MySQL II

MySQL II

- Working with Your MySQL Database
  - What is SQL?
  - Inserting Data into a Database
  - Retrieving Data from the Database
  - Updating Records in the Database
  - Altering Tables After Creation
  - Deleting Record from Database
  - Dropping Tables and Database
- Accessing Your MySQL Database
- Workshop

Working w/ Your MySQL Database

- What is SQL?
  - SQL stands for *Structured Query Language*.
  - It 's the standard language for accessing *relational database management systems (RDBMS)*.
  - SQL is used to store and retrieve data to and from a database.
  - It is used in database systems such as MySQL, Oracle, PostgreSQL, Sybase, and Microsoft SQL Server among others.

Working w/ Your MySQL Database

- Inserting Data into a Database
  - Using the INSERT statement to put rows of data into the database.
  - The usual form of an INSERT statement is

```
INSERT [INTO] table [(column1,column2,column3,...)]
   VALUES(value1,value2,value3,...);
```

  - For example, to insert a record into Book-O-Rama 's Customers table, you could type

```
insert into customers values
   (NULL, "Julie Smith", "25 Oak Street", "Airport West");
```

    - NOTE: Strings should always be enclosed in pairs of single or double quotes in MySQL. Numbers and dates do not need quotes.

Inserting Data into a Database (cont.)

- If you want to fill in only some of the columns, or if you want to specify them in a different order, you can list the specific columns in the columns part of the statement. For example,

```
insert into customers (name, city)
   values("Melissa Jones", "Nar Nar Goon North");
```

- You can also achieve the same effect with the following syntax:

```
insert into customers
   set name="Michael Archer",
       address="12 Adderley Avenue",
       city="Leeton";
```

  - NOTE: You can also insert multiple rows into a table at once. Each row should be in its own set of brackets, and each set of brackets should be separated by comma.

Retrieving Data from the Database

- Using the SELECT statement to retrieve data from a database by selecting rows that match specified criteria from a table.
- The basic form of a SELECT is

```
SELECT items FROM tables
[ WHERE condition ]
[ GROUP BY group_type ]
[ HAVING where_definition ]
[ ORDER BY order_type ]
[ LIMIT limit_criteria ];
```

  - For example, this query lists the contents of the name and city columns from the Customers table:

```
select name, city from customers;
```

  - Or you can use the wildcard operator ,*, which matches all the columns in the specified table or tables.

name	city
Julie Smith	Airport West
Alan Wong	Box Hill
Michelle Arthur	Yarraville
Melissa Jones	Nar Nar Goon North
Michael Archer	Leeton

Retrieving Data from the Database (cont.)

- Retrieving Data with Specific Criteria
  - The following command will select all the columns from the orders table, but only the rows with a customerid of 3.

```
select * from orders where customerid = 3;
```

  - The following command will select all the columns from the orders table, but only the rows with a customerid of 3 or 4.

```
select * from orders where customerid = 3 or
custimerid = 4;
```


- Useful Comparison Operators for WHERE Clauses

Operator	Example	Description
=	customerid = 3	Tests whether two values are equal
>	amount > 60.00	Tests whether one value is greater than another
<	amount < 60.00	Tests whether one value is less then another
>=	amount >= 60.00	Tests whether one value is greater than or equal to another
<=	amount <= 60.00	Tests whether one value is less than or equal to another
!= or <>	quantity != 0	Tests whether two values are not equal
IS NOT NULL	address is not NULL	Tests whether field actually contains a value
IS NULL	address is null	Tests whether field does not contain a value
BETWEEN	amount between 0 and 60.00	Tests whether a value is greater than or equal to a minimum value and less than or equal to a maximum value
IN	city in ("Carlton", "Moe")	Tests whether a value is in particular set
NOT IN	city not in ("Carlton", "Moe")	Tests whether a value is not in a set
LIKE	name like ("Fred %")	Checks whether a value matches a pattern using simple SQL pattern matching
NOT LIKE	name not like ("Fred %")	Checks whether a value doesn't match a pattern
REGEXP	name regexp	Checks whether a value matches regular expression

Retrieving Data from the Database (cont.)

- Retrieving Data from Multiple Tables
  - Simple Two-Table Joins

```
select orders.orderid, orders.amount,
orders.date
from customers, orders
where customers.name = 'Julie Smith'
and customers.customerid =
orders.customerid;
```

- Telling MySQL to only put rows in the result table if the customerid from the Customers table matches the customerid from the Orders table.

orderid	amount	date
2	49.99	2000-04-15

Retrieving Data from the Database (cont.)

- Joining More Than Two Tables

```
select customers.name
from customers, orders, order_items, books
where customers.customerid =
orders.customerid
and orders.orderid = order_items.orderid
and order_items.isbn = books.isbn
and books.title like '%Java%';
```

- This query will return the following output:

name
Michelle Arthur

Retrieving Data from the Database (cont.)

- Finding Rows That Don 't Match
  - Sometimes we specifically want the rows where there 's no match – for example, customers who have never placed an order, or books that have never been ordered.
  - The easiest way to answer this type of question in MySQL is to use a **left join**. A left join will match up rows on a specified join condition between two tables. If there 's no matching row in the right table, row will be added to the result that contains NULL values in the right columns.

```
select customers.customerid, customers.name, orders.orderid
from customers left join orders
on customers.customerid = orders.customerid;
```

customerid	name	orderid
1	Julie Smith	2
2	Alan Wong	3
3	Michelle Arthur	1
3	Michelle Arthur	4
4	Melissa Jones	NULL
5	Michael Archer	NULL

Retrieving Data from the Database (cont.)

- Finding Rows That Don 't Match
  - If we want to see only the customers who haven 't ordered anything, we can do this by checking for those NULL in the primary key field of the right table (in this case orderid as that should not be NULL in any real rows:

```
select customers.customerid,
customers.name
from customers left join orders
using (customerid)
where orders.orderid is null;
```

customerid	name
4	Melissa Jones
5	Michael Archer

Notice that the USING syntax doesn't specify the table from which the join attribute comes—for this reason, the columns in the two tables must have the same name if you want to use USING.

Retrieving Data from the Database (cont.)

- Retrieving Data in a Particular Order
  - If you want to display rows retrieved by query in a particular order, you can use the ORDER BY clause of the SELECT statement. The **default ordering** is ascending order.

```
select name, address
from customers
order by name;
```

```
+-----------------+--------------------+
| name            | address            |
+-----------------+--------------------+
| Alan Wong       | 1/47 Haines Avenue |
| Julie Smith     | 25 Oak Street      |
| Melissa Jones   |                    |
| Michael Archer  | 12 Adderley Avenue |
| Michelle Arthur | 357 North Road     |
+-----------------+--------------------+
```

```
select name, address
from customers
order by name asc;
```

```
select name, address
from customers
order by name desc;
```


Retrieving Data from the Database (cont.)

- Grouping and Aggregating Data

Name	Description
AVG(*column*)	Average of values in the specified column.
COUNT(*items*)	If you specify a column, this will give you the number of non-NULL values in that column. If you add the word DISTINCT in front of the column name, you will get a count of the distinct values in that column only. If you specify COUNT(*), you will get a row count regardless of NULL values.
MIN(*column*)	Minimum of values in the specified column.
MAX(*column*)	Maximum of values in the specified column.
STD(*column*)	Standard deviation of values in the specified column.
STDDEV(*column*)	Same as STD(column).
SUM(*column*)	Sum of values in the specified column.

```
select avg(amount)
from orders;
```

```
+-------------+
| avg(amount) |
+-------------+
|   54.985002 |
+-------------+
```

```
select customerid, avg(amount)
from orders group by customerid;
```

```
+------------+-------------+
| customerid | avg(amount) |
+------------+-------------+
|          1 |   49.990002 |
|          2 |   74.980003 |
|          3 |   47.485002 |
+------------+-------------+
```


Retrieving Data from the Database (cont.)

- Choosing Which Rows to Return
  - Using **LIMIT** to specify which rows from the output should be returned.
  - It takes two parameters: the row number from which to start and the number of rows to return.

```
select name from customers
limit 2, 3;
```

  - This query can be read as, "Select name from customers, and then return 3 rows, starting from row 2 in the output.
  - Note that row numbers are zero indexed –that is, the first row in the output is row number zero.

Updating Records in the Database

- The usual form of an UPDATE statement is

```
UPDATE tablename
SET column1 = expression1,
column2 = expression2,...
[WHERE condition ]
[LIMIT number ]
```

- If we want to increase all the book prices by 10%,we can use an UPDATE statement without a WHERE clause:

```
update books
set price = price*1.1;
```

- If, on the other hand, we want to change a single row –say, to update a customer 's address –we can do it like this:

```
update customers
set address = '250 Olsens Road'
where customerid = 4;
```

Working w/ Your MySQL Database

Altering Tables After Creation

- The basic form of this statement is

```
ALTER TABLE tablename alteration [, alteration ...]
```

Syntax	Description
ADD [COLUMN] *column_description* [FIRST | AFTER *column*]	Add a new column in the specified location (if not specified,then the column goes at the end).Note that *column_descriptions* need a name and a type,just as in a CREATE statement.
ADD [COLUMN] *(description , column_description ,...)*	Add one or more new columns at the end of the table.
ADD INDEX [*index*] *(column,...)*	Add an index to the table on the specified column or columns.
ADD PRIMARY KEY *(column,...)*	Make the specified column or columns the primary key of the table.
ADD UNIQUE [*index*] *(column,...)*	Add a unique index to the table on the specified column or columns.
ALTER [COLUMN] *column* {SET DEFAULT *value* |DROP DEFAULT}	Add or remove default value for a particular column.

---------- more ----------

Working w/ Your MySQL Database

Altering Tables After Creation (cont.)

Syntax	Description
CHANGE [COLUMN] *column new_column _description*	Change the column called *column* so that it has the description listed. Note that this can be used to change the name of a column because a *column_description* includes name.
MODIFY [COLUMN] *column_description*	Similar to CHANGE. Can be used to change column types, not names.
DROP [COLUMN] *column*	Delete the named column.
DROP PRIMARY KEY	Delete the primary index (but not the column).
DROP INDEX *index*	Delete the named index.
RENAME [AS] *new_table_name*	Rename a table.

- We can change the data type of the column to be 45 characters long

```
alter table customers
modify name char(45) not null;
```

- We can add and drop a tax column to the Orders table as follows:

```
alter table orders
add tax float(6,2) after amount;
```

```
alter table orders
drop tax;
```


Working w/ Your MySQL Database

Deleting Record from Database

- Deleting rows from the database is very simple. You can do this using the DELETE statement, which generally looks like this:

```
DELETE FROM table
[WHERE condition ] [LIMIT number ]
```

  - If a particular customer hadn't placed any orders for a long time, and you wanted to do some housekeeping:

```
delete from customers
where customerid=5;
```

- Dropping a table and Dropping a whole database
  - To get rid of an entire table.

```
DROP TABLE table;
```

  - To get rid of a whole database

```
DROP DATABASE database;
```


Accessing Your MySQL Database

- Connecting to Your Database and Creating a Table
- Deleting a Database using PHP
- How Web Database Architectures Work
- The Basic Steps in Querying a Database from the Web
- Putting New Information in the Database
- Other Useful PHP-MySQL Functions

Creating Your Web Database

Connecting to Your Database and Create a Table

- Using function mysqli_connect() and mysqli_query()

```
<?php

  //กำหนดค่าตัวแปรสำหรับ ชื่อโฮสต์ ชื่อฐานข้อมูล รหัสผ่าน
  $host = "localhost";
  $user = "root";
  $password = "";
  //เปิดการเชื่อมต่อกับ MySQL Server โดยใช้ฟังก์ชัน mysqli_connect()
  $conn = mysqli_connect($host,$user,$password)
     or die("ไม่สามารถติดต่อกับเซิร์ฟเวอร์ได้");

  echo "ยินดีต้อนรับ.....";
  mysqli_select_db($conn,"mydb");

  // continue next slide
```


Creating Your Web Database


```
 $sql = "CREATE TABLE person(
          code VARCHAR(8) NOT NULL,
          firstname VARCHAR(20),
          lastname VARCHAR(20),
          sex VARCHAR(5),
          level VARCHAR(9),
          note TEXT,
          PRIMARY KEY (code))";
  $result = mysqli_query($conn,$sql) or die ("ไม่สามารถสร้างตารางได้");

  //ยืนยันการสร้างตารางสำเร็จ
  echo "สร้างตาราง person ในฐานข้อมูล mydb สำเร็จแล้ว";

  //ปิดการเชื่อมต่อกับ MySQL server
  mysqli_close($conn);
?>
```

Type tinytext has a maximum length of $2^8 - 1$ or 255 characters.

Type text has a maximum length of $2^{16} - 1$ or 65535 characters.

Creating Your Web Database

Deleting a Database using PHP

- Using SQL command and function mysqli_query()

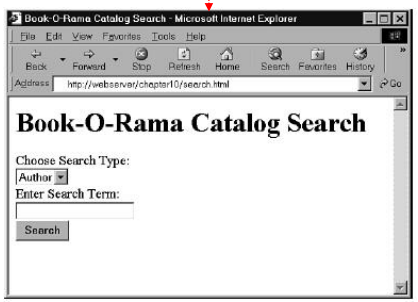
```
<?php
  $host = "localhost";
  $user = "root";
  $password = "";
  $conn = mysqli_connect($host, $user, $password)
          or die("ไม่สามารถติดต่อกับเซิร์ฟเวอร์ได้");
  $sql = "drop database tempp";
  $result = mysqli_query($conn, $sql)
        or die("ไม่สามารถลบฐานข้อมูลได้");
  //ยืนยันการลบฐานข้อมูลสำเร็จ
  echo "ลบฐานข้อมูล mydatabase สำเร็จแล้ว";

  //ปิดการเชื่อมต่อกับ MySQL server
  mysqli_close($conn);
?>
```


Accessing Your MySQL Database

How Web Database Architectures Work

search.html

```
<html>
<head>
  <title>Book-O-Rama Catalog Search</title>
</head>
<body>
  <h1>Book-O-Rama Catalog Search</h1>
  <form action="results.php" method="post">
    Choose Search Type:<br />
    <select name="searchtype">
      <option value="author">Author</option>
      <option value="title">Title</option>
      <option value="isbn">ISBN</option>
    </select>
  <br />
  Enter Search Term:<br />
  <input name="searchterm" type="text">
  <br />
  <input type="submit" value="Search">
  </form>
</body>
</html>
```



How Web Database Architectures Work (cont.)

results.php

```
<html>
<head>
  <title>Book-O-Rama Search Results</title>
</head>
<body>
  <h1>Book-O-Rama Search Results</h1>
  <?php
    // create short variable names
    $searchtype=$_POST['searchtype'];
    $searchterm=$_POST['searchterm'];
    $searchterm= trim($searchterm);
    if (!$searchtype || !$searchterm){
      echo 'You have not entered search details. Please go back and try again.';
      exit;
    }
    $searchtype = addslashes($searchtype);
    $searchterm = addslashes($searchterm);
    @ $conn = mysqli_connect('localhost', 'bookorama', 'bookorama123');
    if (!$conn){
      echo 'Error: Could not connect to database. Please try again later.';
      exit;
    }
// continue to next page
```

Returns a string with **backslashes** before characters that need to be quoted in database queries etc. These characters are single quote (’), double quote (”), backslash (\) and NUL (the **NULL** byte)

How Web Database Architectures Work (cont.)

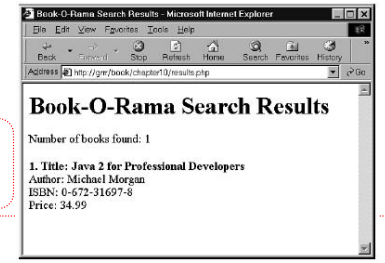
results.php

```
    mysqli_select_db($conn, 'books');
    $query = "select * from books where ".$searchtype." like '%".$searchterm."%'";
    $result = mysqli_query($conn, $query);
    $num_results = mysqli_num_rows($result);
    echo '<p>Number of books found: '.$num_results.'</p>';
    for ($i=0; $i<$num_results; $i++){
      $row = mysqli_fetch_array($result);
      echo '<p><strong>'.($i+1).'. Title: ';
      echo htmlspecialchars(stripslashes($row['title']));
      echo '</strong><br />Author: ';
      echo stripslashes($row['author']);
      echo '<br />ISBN: ';
      echo stripslashes($row['isbn']);
      echo '<br />Price: ';
      echo stripslashes($row['price']);
      echo '</p>';
    }
    mysqli_close($conn);
  ?>
</body>
</html>
```

a% Finds any values that start with "a“
%a Finds any values that end with "a“
%a% Finds any values that have "or" in any position

Fetch a result row as an associative array, a numeric array, or both

Convert special characters to HTML entities
eg. ‘&’ becomes ‘&amp’,
‘>’ becomes ‘&gt. etc.



The Basic Steps in Querying a Database from the Web

- 1. Check and filter data coming from the user.
- 2. Set up a connection to the appropriate database.
- 3. Query the database.
- 4. Retrieve the results.
- 5. Present the results back to the user.

The Basic Steps in Querying a Database from the Web

- 1. Check and filter data coming from the user.
  - Stripping any whitespace that the user might have inadvertently entered at the beginning or end of his search term.

```
$searchterm = trim($searchterm);
if (!$searchtype || !$searchterm){
  echo 'You have not entered search details. Please go
        back and try again.';
  exit;
}
```

  - Need to use **addslashes()** when submitting any user input to a database such as MySQL and **stripslashes()** when returning output to the user who has had control characters slashed out.
    - Certain characters are perfectly valid as part of a string but can cause problems, particularly when inserting data into a database because the database could interpret these characters as control characters.
    - The problematic ones are quotes (single and double), backslashes (\), and the NULL character.

The Basic Steps in Querying a Database from the Web

- 2. Set up a connection to the appropriate database.
  - Connecting to MySQL server using **mysqli_connect()**:
  - Prototype of mysqli_connect():
    - **resource mysqli_connect ([string server [, string username [, string password [, bool new_link [, int client_flags]]]]])**

```
@ $conn = mysqli_connect('localhost', 'bookorama', 'bookorama123');
```


The Basic Steps in Querying a Database from the Web

- 3. Query the database.
  - When we reusing MySQL from a command line interface, we need to tell it which database we plan to use with a command such as

```
use books;
```

  - We perform this from PHP with a call to the **mysqli_select_db()** function:

mysqli_select_db(*connection, dbname*);

```
mysqli_select_db($conn, 'books');
```

  - To actually perform the query, we can use the mysqli_query() function.
  - Before doing this, however, it 's a good idea to set up the query you want to run:

```
$query = "select * from books where ".$searchtype." like '%". $searchterm. "%'";
```

  - We can now run the query:

```
$result = mysqli_query($conn, $query);
```


The Basic Steps in Querying a Database from the Web

- 4. Retrieve the results.
  - Using **mysqli_num_rows()** and **mysqli_fetch_array().**
    - The function mysqli_num_rows() gives you the number of rows returned by the query.

```
$num_results = mysqli_num_rows($result);
for ($i=0; $i <$num_results; $i++){
// process results
}
```

    - In each iteration of this loop, we are calling mysqli_fetch_array().
      - » The loop will not execute if no rows are returned.
      - » This is a function that takes each row from the result set and returns the row as an associative array, with each key an attribute name and each value the corresponding value in the array:

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


The Basic Steps in Querying a Database from the Web

- 5. Present the results back to the user.
  - Given the associative array $row we can go through each field and display them appropriately , for example:

```
echo '<br />ISBN: ';
echo stripslashes($row['isbn']);
```

    - There re several variations on getting results from a result identifier. Instead of an associative array, we can retrieve the results in an enumerated array with **mysqli_fetch_row(),** as follows:

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

    - The attribute values will be listed in each of the array values $row[0], $row[1], …
    - You could also fetch a row into an object with the **mysqli_fetch_object()** function:

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

    - You can then access each of the attributes via $row->title, $row->author, …
  - Disconnecting from the database

```
mysqli_close(database_connection);
```

Putting New Information in the Database

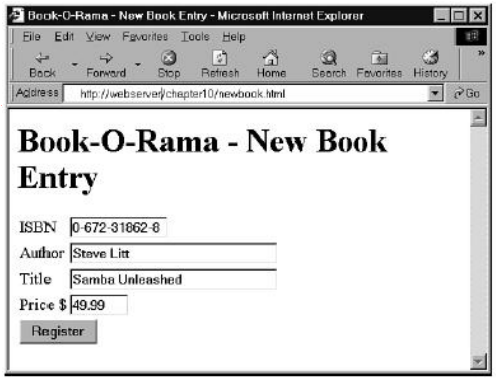
newbook.html

```
<html>
<head>
  <title>Book-O-Rama - New Book Entry</title>
</head>
<body>
  <h1>Book-O-Rama - New Book Entry</h1>
  <form action="insert_book.php" method="post">
  <table border="0">
    <tr>
      <td>ISBN</td>
      <td><input type="text" name="isbn" maxlength="13" size="13"><br /></td>
    </tr>
    <tr>
      <td>Author</td>
      <td> <input type="text" name="author" maxlength="30" size="30"><br /></td>
    </tr>
    <tr>
      <td>Title</td>
      <td> <input type="text" name="title" maxlength="60" size="30"><br></td>
    </tr>
// continue next page
```


Putting New Information in the Database (cont.)

newbook.html

```
    <tr>
      <td>Price $</td>
      <td><input type="text" name="price" maxlength="7" size="7"><br /></td>
    </tr>
    <tr>
      <td colspan="2"><input type="submit" value="Register"></td>
    </tr>
  </table>
  </form>
</body>
</html>
```



Putting New Information in the Database (cont.)

insert_book.php

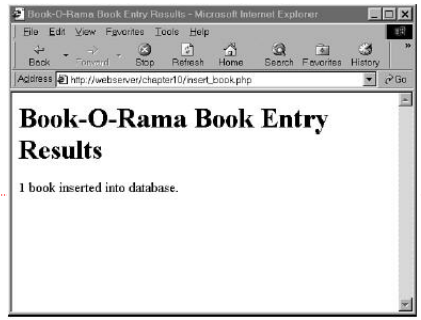
```
<html>
<head>
  <title>Book-O-Rama Book Entry Results</title>
</head>
<body>
  <h1>Book-O-Rama Book Entry Results</h1>
  <?php
    // create short variable names
    $isbn=$_POST['isbn'];
    $author=$_POST['author'];
    $title=$_POST['title'];
    $price=$_POST['price'];
    if (!$isbn || !$author || !$title || !$price){
      echo 'You have not entered all the required details.<br />'
            .'Please go back and try again.';
      exit;
    }
    $isbn = addslashes($isbn);
    $author = addslashes($author);
    $title = addslashes($title);
    $price = floatvar($price);
// continue next page
```

Return the float value of $price.
Example:
$price = "250.75text";
echo floatvar($price); // 250.75

Putting New Information in the Database (cont.)

insert_book.php

```
    @ $conn = mysqli_connect('localhost', 'bookorama', 'bookorama123');
    if (!$conn){
      echo 'Error: Could not connect to database. Please try again later.';
      exit;
    }
    mysqli_select_db($conn, 'books');
    $query = "insert into books values
              ('".$isbn."', '".$author."', '".$title."', '".$price."')";
    $result = mysqli_query($conn, $query);
    if($result)
      echo mysqli_affected_rows().
          ' book inserted into database.';
    mysqli_free_result($result);
    mysqli_close($conn);
  ?>
</body>
</html>
```



Accessing Your MySQL Database

Other Useful PHP-MySQL Functions

- Freeing Up Resources
  - Using function **mysqli_free_result()** which has the following prototype:
    - bool mysql_free_result(resource *result*);
  - You call it with a result identifier, like this:

```
mysqli_free_result($result);
```

    - This has the effect of freeing up the memory used to store the result.
- Creating and Deleting Databases
  - Creating a Database: Using SQL and function mysqli_query()

    $sql = "create database *dbname*";

    $result = mysqli_query($conn, $sql) or die("ไม่สามารถสร้างฐานข้อมูลได้")
  - Deleting a Database: Using function mysql_drop_db() to drop a database

    $sql = "drop database *dbname*";

    $result = mysqli_query($conn, $sql) or die("ไม่สามารถลบฐานข้อมูลได้")

Workshop

MySQL II

- Page 262 (1-3)