

CSC 498: Web Programming

Haidar Harmanani

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



Introduction to SQL

- SQL is a ANSI standard computer language for accessing and manipulating databases
 - Stands for Structured Query Language
- Using SQL you can:
 - Access a database
 - Execute queries, and retrieve data
 - Insert, delete and update records
- SQL works with database programs like MS Access, DB2, Informix, MS SQL Server, Oracle, Sybase, mySQL, etc.
- Unfortunately, there are many different versions. But, they must support the same major keywords in a similar manner such as SELECT, UPDATE, DELETE, INSERT, WHERE, etc.
- Most of the SQL database programs also have their own proprietary extensions!



SQL Database Tables

- A database most often contains one or more tables. Each table is identified by a name (e.g. "Customers" or "Orders"). Tables contain records (rows) with data
- The table below, called "Persons", contains three records (one for each person) and four columns (LastName, FirstName, Address, and City)

LastName	FirstName	Address	City
Hansen	Ola	Timoteivn 10	Sandnes
Svendson	Tove	Borgvn 23	Sandnes
Pettersen	Kari	Storgt 20	Stavanger



SQL Queries

- With SQL, you can query a database and have a result set returned.
 - The mySQL database system requires a semicolon at the end of the SQL statement!
- Example: `SELECT LastName FROM Persons;`

Gives a result set like this:

LastName
Hansen
Svendson
Pettersen



SQL Data Languages

- The query and update commands together form the Data Manipulation Language (DML) part of SQL:
 - SELECT - extracts data from a database table
 - UPDATE - updates data in a database table
 - DELETE - deletes data from a database table
 - INSERT INTO - inserts new data into a database table
- The Data Definition Language (DDL) part of SQL permits database tables to be created or deleted:
 - CREATE TABLE - creates a new database table
 - ALTER TABLE - alters (changes) a database table
 - DROP TABLE - deletes a database table
 - CREATE INDEX - creates an index (search key)
 - DROP INDEX - deletes an index



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 |
+-----+-----+-----+-----+
```



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)
```



Getting Data Out of the Table

- The `SELECT` command is the main way of getting data out of a table, or set of tables.

```
SELECT * FROM students;
```

– Here the asterisk means to select (i.e. return the information in) all columns.

- You can specify one or more columns of data that you like, such as

```
SELECT f_name,l_name FROM students;
```

```
+-----+-----+
| f_name | l_name |
+-----+-----+
| Russell | Martin |
| James  | Bond   |
+-----+-----+
2 rows in set (0.00 sec)
```



Getting Data Out of the Table (cont.)

- You can specify other information that you want in the query using the `WHERE` clause.

```
SELECT * FROM students WHERE l_name='Bond';
```

```
+-----+-----+-----+-----+
+-----+-----+
| num | f_name | l_name | student_id | email          |
+-----+-----+-----+-----+
| 2 | James | Bond   | 7       | bond@csc.liv.ac.uk  |
+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
SELECT student_id, email FROM students WHERE l_name='Bond';
```

```
+-----+-----+
| student_id | email          |
+-----+-----+
| 7 | bond@csc.liv.ac.uk |
+-----+-----+
```



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)
```



The Final Table

We'll first add another column, update the (only) record, then insert more data.

```
mysql> ALTER TABLE students ADD gr INT;
```

Query OK, 1 row affected (0.01 sec)

Records: 1 Duplicates: 0 Warnings: 0

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

```
+-----+-----+-----+-----+-----+-----+
| num | f_name | l_name | student_id | email          | date       | gr |
+-----+-----+-----+-----+-----+-----+
| 1 | Russell | Martin | 396310 | martin@csc.liv.ac.uk | 2007-11-15 | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> UPDATE students SET gr=3 WHERE num=1;
```

Query OK, 1 row affected (0.00 sec)

Rows matched: 1 Changed: 1 Warnings: 0

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

```
+-----+-----+-----+-----+-----+-----+
| num | f_name | l_name | student_id | email          | date       | gr |
+-----+-----+-----+-----+-----+-----+
| 1 | Russell | Martin | 396310 | martin@csc.liv.ac.uk | 2007-11-15 | 3 |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

```
mysql> INSERT INTO students VALUES(NULL,'James','Bond',007,'bond@csc.liv.ac.uk','2007-11-15',4);
```

...



The Final Table (cont.)

```
...
...
mysql> INSERT INTO students VALUES(NULL,'Hugh','Milner',75849789,'hugh@poughkeepsie.ny', CURRENT_DATE,
2);
Note: CURRENT_DATE is a built-in SQL command which (as expected) gives the current (local) date.
mysql> SELECT * FROM students;
+-----+-----+-----+-----+-----+-----+-----+
| num | f_name | l_name | student_id | email | date | gr |
+-----+-----+-----+-----+-----+-----+-----+
| 1 | Russell | Martin | 396310 | martin@csc.liv.ac.uk | 2007-11-15 | 3 |
| 5 | Kate | Ash | 124309 | kate@ozymandius.co.uk | 2007-11-16 | 5 |
| 3 | James | Bond | 7 | bond@csc.liv.ac.uk | 2007-11-15 | 4 |
| 4 | Bob | Jones | 12190 | bob@nowhere.com | 2007-11-16 | 3 |
| 6 | Pete | Lofton | 76 | lofton@iwannabesedated.com | 2007-11-17 | 2 |
| 7 | Polly | Crackers | 1717 | crackers@polly.org | 2007-11-17 | 4 |
| 8 | Hugh | Milner | 75849789 | hugh@poughkeepsie.ny | 2007-11-17 | 2 |
+-----+-----+-----+-----+-----+-----+-----+
7 rows in set (0.00 sec)
mysql> exit
Bye
```



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;



Putting Content into Your Database with

- 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();



Student Database: data_in.php

```
<html>
<head>
<title>Putting Data in the DB</title>
</head>
<body>
<?php
/*insert students into DB*/
if(isset($_POST["submit"])) {

    $db = mysql_connect("mysql", "martin");
    mysql_select_db("martin");

    $date=date("Y-m-d"); /* Get the current date in the right SQL format */

    $sql="INSERT INTO students VALUES(NULL, '$_POST['f_name']', '$_POST['l_name']', '$_POST['student_id']', '$_POST['email']', '$date', '$_POST['gr']'); /* construct the query */

    mysql_query($sql);
    mysql_close();

    echo"<h3>Thank you. The data has been entered.</h3> \n";

    echo"<p><a href='data_in.php'>Back to registration</a></p>". "\n";

    echo"<p><a href='data_out.php'>View the student lists</a></p>". "\n";

}
}
```



Student Database: data_in.php

```
else {
?>
<h3>Enter your items into the database</h3>
<form action="data_in.php" method="POST">
First Name: <input type="text" name="f_name" /> <br/>
Last Name: <input type="text" name="l_name" /> <br/>
ID: <input type="text" name="student_id" /> <br/>
email: <input type="text" name="email" /> <br/>
Group: <select name="gr">
  <option value ="1">1</option>
  <option value ="2">2</option>
  <option value ="3">3</option>
  <option value ="4">4</option>
</select><br/><br/>
<input type="submit" name="submit" /> <input type="reset" />
</form>

<?php
}
?>

</body>
</html>
```

view the output page



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();



Student Database: data_out.php

```
<html>
<head>
<title>Getting Data out of the DB</title>
</head>
<body>
<h1> Student Database </h1>
<p> Order the full list of students by
<a href="data_out.php?order=date">date</a>,
<a href="data_out.php?order=student_id">id</a>, or
by <a href="data_out.php?order=l_name">surname</a>.
</p>
```

(continued)



Student Database: data_out.php

```
...
<p>
<form action="data_out.php" method="POST">
Or only see the list of students in group
<select name="gr">
  <option value ="1">1</option>
  <option value ="2">2</option>
  <option value ="3">3</option>
  <option value ="4">4</option>
</select>
<br/>
<input type="submit" name="submit" />
</form>
</p>
```



Student Database: data_out.php

```
<?php
/*get students from the DB */
$db = mysql_connect("mysql","martin");
mysql_select_db("martin", $db);

switch($_GET["order"]){
case 'date': $sql = "SELECT * FROM students ORDER BY date"; break;
case 'student_id': $sql = "SELECT * FROM students ORDER BY student_id"; break;
case 'l_name': $sql = "SELECT * FROM students ORDER BY l_name"; break;

default: $sql = "SELECT * FROM students";
}

if(isset($_POST["submit"])){
    $sql = "SELECT * FROM students WHERE gr=" . $_POST["gr"];
}

$result=mysql_query($sql);
while($row=mysql_fetch_array($result)){
    echo "<h4> Name: " . $row["l_name"] . ', ' . $row["f_name"] . "</h4> \n";
    echo "<h5> ID: " . $row["student_id"] . "<br/> Email: " . $row["email"] . "<br/> Group: " . $row["gr"] . "<br/> Posted: " . $row["date"] . "</h5> \n";
}
mysql_close();
?>
</body>
</html>
```

[view the output page](#)



Can Do Even More with PHP

- Can create a table
- Can delete rows and columns
- Can make updates
- Can make queries to several tables
- Can get connected to several databases



Using several tables

- SQL is a relational database, meaning that it's designed to work with multiple tables and allows you to make queries that involve several tables
- Using multiple tables allows us to store lots of information without much duplication
- Allows for easier updating (both insertion and deletion)
- We can also perform different types of queries, combining the information in different ways depending upon our needs.



Advanced queries

- Suppose that we have defined several tables as follows:

```
mysql> describe clients;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| client_id | int(11) | NO | PRI | NULL | auto_increment |
| f_name | varchar(20) | YES | | NULL | |
| l_name | varchar(30) | NO | | | |
| address | varchar(40) | YES | | NULL | |
| city | varchar(30) | YES | | NULL | |
| postcode | varchar(12) | YES | | NULL | |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql> describe purchases;
+-----+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| purchase_id | int(11) | NO | PRI | NULL | auto_increment |
| client_id | int(11) | NO | | | |
| date | date | NO | | | |
+-----+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```



```
mysql> describe itemlist;
```

Field	Type	Null	Key	Default	Extra
item_id	int(11)	NO	PRI	NULL	auto_increment
purchase_id	int(11)	NO			
book_id	int(11)	NO			

```
3 rows in set (0.00 sec)
```

```
mysql> describe books;
```

Field	Type	Null	Key	Default	Extra
book_id	int(11)	NO	PRI	NULL	auto_increment
title	varchar(50)	NO			
pages	int(11)	YES		NULL	

```
3 rows in set (0.00 sec)
```

```
mysql>
```



Advanced queries

- The idea here is that clients can make multiple purchases. Each purchase will be assigned a unique id, but can consist of multiple items. Each item is a book, which can be purchased by many different people.
- Defining the tables in this fashion allows us to avoid (excessive) repetition of information, and lets us query the data in different fashions.
- The “id” fields are the keys that we use to tie the various tables together, namely a customer (or client) with “client_id” will make several purchases. We can identify their purchases by searching for matches of the client_id in the purchases table.
- Similarly, we can find the actual items that comprise a particular purchase by searching for the purchase_id key in the itemlist table.



Populate the tables

- Suppose that we have this data in the tables:

```
mysql> select * from clients;
```

client_id	f_name	l_name	address	city	postcode
1	Russell	Martin	Dept of Computer Science	Liverpool	L69 3BX
2	Bob	Milnor	12 Peachtree Ln	Liverpool	L12 3DX
3	Sarah	Ford	542b Jersey Rd	West Kirby	L43 8JK
4	Larry	Vance	76 Jarhead Ln	Liverpool	L12 4RT
5	Paul	Abbott	90 Crabtree Pl	Leamingotn Spa	CV32 7YP

```
5 rows in set (0.01 sec)
```

```
mysql> select * from books;
```

book_id	title	pages
1	Linux in a Nutshell	120



```
mysql> SELECT * FROM purchases;
```

purchase_id	client_id	date
1	1	2007-11-09
2	1	2007-11-10
4	2	2007-11-20
5	4	2007-11-20
6	3	2007-11-21
7	5	2007-11-25
8	3	2007-11-25

```
7 rows in set (0.00 sec)
```

```
mysql> SELECT * FROM itemlist;
```

item_id	purchase_id	book_id
1	1	2
2	1	6
3	1	3
4	2	4
5	2	5
6	4	5
7	4	6
8	5	1
9	5	3
10	6	5
11	7	2
12	8	3

```
12 rows in set (0.00 sec)
```

```
mysql>
```



Advanced Queries

- We can link these tables together by queries of this type:

```
mysql> SELECT * from clients, purchases WHERE clients.client_id=purchases.client_id ORDER BY purchase_id;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| client_id | f_name | l_name | address                | city      | postcode | purchase_id | client_id | date      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Russell | Martin | Dept of Computer Science | Liverpool | L69 3BX  | 1 | 1 | 2007-11-09 |
| 1 | Russell | Martin | Dept of Computer Science | Liverpool | L69 3BX  | 2 | 1 | 2007-11-10 |
| 2 | Bob    | Milnor | 12 Peachtree Ln        | Liverpool | L12 3DX  | 4 | 2 | 2007-11-20 |
| 4 | Larry  | Vance  | 76 Jarhead Ln          | Liverpool | L12 4RT  | 5 | 4 | 2007-11-20 |
| 3 | Sarah  | Ford   | 542b Jersey Rd         | West Kirby | L43 8JK  | 6 | 3 | 2007-11-21 |
| 5 | Paul   | Abbott | 90 Crabtree Pl         | Leamingotn Spa | CV32 7YP | 7 | 5 | 2007-11-25 |
| 3 | Sarah  | Ford   | 542b Jersey Rd         | West Kirby | L43 8JK  | 8 | 3 | 2007-11-25 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

7 rows in set (0.01 sec)

mysql>

So you can see that this query basically gives us all of the purchase orders that have been placed by the clients (but not the number of items, or the items themselves).



Advanced Queries

- You can see that the "client_id" field is repeated. This is because we selected all columns (using the * option) in both tables, and it appears in each table.
- To avoid this repeated information, we can make a query like:

```
mysql> SELECT clients.client_id, f_name, l_name, address, city, postcode, purchases.purchase_id, date from clients, purchases WHERE clients.client_id=purchases.client_id ORDER BY purchase_id;
```

```
+-----+-----+-----+-----+-----+-----+-----+-----+
| client_id | f_name | l_name | address                | city      | postcode | purchase_id | date      |
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | Russell | Martin | Dept of Computer Science | Liverpool | L69 3BX  | 1 | 2007-11-09 |
| 1 | Russell | Martin | Dept of Computer Science | Liverpool | L69 3BX  | 2 | 2007-11-10 |
| 2 | Bob    | Milnor | 12 Peachtree Ln        | Liverpool | L12 3DX  | 4 | 2007-11-20 |
| 4 | Larry  | Vance  | 76 Jarhead Ln          | Liverpool | L12 4RT  | 5 | 2007-11-20 |
| 3 | Sarah  | Ford   | 542b Jersey Rd         | West Kirby | L43 8JK  | 6 | 2007-11-21 |
| 5 | Paul   | Abbott | 90 Crabtree Pl         | Leamingotn Spa | CV32 7YP | 7 | 2007-11-25 |
| 3 | Sarah  | Ford   | 542b Jersey Rd         | West Kirby | L43 8JK  | 8 | 2007-11-25 |
+-----+-----+-----+-----+-----+-----+-----+-----+
```

7 rows in set (0.00 sec)

mysql>



Advanced Queries

- We need not select all columns:

```
mysql> SELECT f_name, l_name, purchases.purchase_id FROM clients NATURAL JOIN purchases ORDER BY purchase_id;
```

```
+-----+-----+-----+
| f_name | l_name | purchase_id |
+-----+-----+-----+
| Russell | Martin | 1 |
| Russell | Martin | 2 |
| Bob    | Milnor | 4 |
| Larry  | Vance  | 5 |
| Sarah  | Ford   | 6 |
| Paul   | Abbott | 7 |
| Sarah  | Ford   | 8 |
+-----+-----+-----+
```

7 rows in set (0.00 sec)

mysql>



More Complex Queries

- We can create almost any type of query that you might think of with a (more complicated) "WHERE" clause:

```
mysql> SELECT purchases.purchase_id, f_name, l_name, date FROM purchases, clients WHERE purchases.client_id=clients.client_id;
```

```
+-----+-----+-----+-----+
| purchase_id | f_name | l_name | date      |
+-----+-----+-----+-----+
| 1 | Russell | Martin | 2007-11-09 |
| 2 | Russell | Martin | 2007-11-10 |
| 4 | Bob    | Milnor | 2007-11-20 |
| 5 | Larry  | Vance  | 2007-11-20 |
| 6 | Sarah  | Ford   | 2007-11-21 |
| 7 | Paul   | Abbott | 2007-11-25 |
| 8 | Sarah  | Ford   | 2007-11-25 |
+-----+-----+-----+-----+
```

7 rows in set (0.00 sec)



More Complex Queries (cont.)

- Find the purchases by the person named "Ford".

```
mysql> SELECT purchases.purchase_id, f_name,
         l_name, date FROM purchases, clients
         WHERE
         (purchases.client_id=clients.client_id) AND
         (l_name='Ford');
```

purchase_id	f_name	l_name	date
6	Sarah	Ford	2007-11-21
8	Sarah	Ford	2007-11-25

2 rows in set (0.01 sec)

mysql>



Querying multiple tables

- In addition, we can query many tables (i.e. more than two) at once:
- First we'll find all the purchases by a person with l_name='Martin'.

```
mysql> select purchases.purchase_id, f_name, l_name, date
        FROM purchases, clients WHERE
        (purchases.client_id=clients.client_id) AND
        (l_name='Martin') ORDER BY purchases.purchase_id;
```

purchase_id	f_name	l_name	date
1	Russell	Martin	2007-11-09
2	Russell	Martin	2007-11-10

2 rows in set (0.00 sec)

mysql>



Querying multiple tables (cont.)

- Now let's find out the items (the "book_id") in each purchase:

```
mysql> SELECT purchases.purchase_id, f_name, l_name, date, itemlist.book_id
        FROM purchases, clients, itemlist
        WHERE (purchases.client_id=clients.client_id) AND (l_name='Martin')
        AND (purchases.purchase_id=itemlist.purchase_id)
        ORDER BY purchases.purchase_id;
```

purchase_id	f_name	l_name	date	book_id
1	Russell	Martin	2007-11-09	2
1	Russell	Martin	2007-11-09	6
1	Russell	Martin	2007-11-09	3
2	Russell	Martin	2007-11-10	5
2	Russell	Martin	2007-11-10	4

5 rows in set (0.00 sec)



Querying multiple tables (cont.)

- Finally we can find the actual book titles by querying all four tables at once:

```
mysql> SELECT purchases.purchase_id, f_name, l_name, date,
         itemlist.book_id, title
        FROM purchases, clients, itemlist, books
        WHERE (purchases.client_id=clients.client_id) AND
        (l_name='Martin') AND
        (purchases.purchase_id=itemlist.purchase_id) AND
        (itemlist.book_id=books.book_id)
        ORDER BY purchases.purchase_id;
```

purchase_id	f_name	l_name	date	book_id	title
1	Russell	Martin	2007-11-09	6	Blackbeard
1	Russell	Martin	2007-11-09	2	Learning SQL
1	Russell	Martin	2007-11-09	3	Abstract Algebra
2	Russell	Martin	2007-11-10	4	Rising Sun
2	Russell	Martin	2007-11-10	5	Round the Moon

5 rows in set (0.00 sec)



Querying multiple tables (cont.)

- As before, we need not select all of the columns:

```
mysql> SELECT purchases.purchase_id, title
      FROM purchases, clients, itemlist, books
      WHERE (purchases.client_id=clients.client_id) AND (l_name='Martin')
      AND (purchases.purchase_id=itemlist.purchase_id) AND
      (itemlist.book_id=books.book_id)
      ORDER BY purchases.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)
```



Using aliases in queries

- Especially long queries might benefit from the SQL capability for using aliases.

```
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)
```

- 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



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)
```



More on PHP and SQL

- To increase security of your PHP/SQL setup (and to make it easier to change the database you use), it's recommended that you build an "include" file that will have the information you use to connect to the database.

```
<?php
/* Save this as db_login.php (or whatever you like) and include it
in your php script. */

// Here's the information to connect to the database.
$db_host = 'mysql';
$db_database='martin';
$db_username='martin';
$db_password='';
?>
```

- If someone tries to view this file through their browser, the PHP interpreter will process it and return a blank page to the user (there's no HTML in the file).



Connecting to the database

- Now you can build your PHP script as follows (using the commands that we discussed previously):

```
<?php
include_once ('db_login.php');
$connection = mysql_connect($db_host, $db_username, $db_password);
if (!$connection)
{
    exit("Could not connect to the database: <br/>" .
        htmlspecialchars(mysql_error()) );
}
else {
    // more statements here. . .
}
?>
```

- Note: The function 'htmlspecialchars()' converts special characters in a string into their HTML escape sequences (like '&' into '&' and so forth). This can also be used to increase the security of your code by and help thwart attacks on your database by passing it information that



MySQL queries inside of PHP

- Your MySQL queries from a PHP script are the same as they are as when you're using the MySQL program from the command line with one difference... the queries do not have a semi-colon at the end.
- Aside from this difference, all of the regular SQL commands and keywords are available when you perform your queries.
- You can create new tables, alter, and delete them from inside of a PHP script, and you can also insert and delete rows of tables as normal too. For any such operation, you'll likely want to check to see if it's successful (especially if you're trying to extract data from the database).

```
<?php
// Assuming a valid database connection has been established.
// Build the query string by assigning variables...
$query = $select . $column . $from . $tables . $where;
$result = mysql_query($query);
if(!$result) {
    exit("Could not query the database: <br/>" .
        htmlspecialchars(mysql_error()) );
}
else {
    // process the data
}
```



Processing the results of a query

- There are two main PHP methods to fetch the results of an SQL query, these being 'mysql_fetch_row()' and 'mysql_fetch_array()'.

```
<?php
// Assuming a database connection, and a valid query
string.
$result = mysql_query( $query );
while ($result_row = mysql_fetch_sql($result)) {
    echo $result_row[2] . '<br/>';
}
?>
```

- The 'mysql_fetch_row()' command fetches the query results as an enumerated array, one row at a time, returning FALSE when there are no more rows (ending the 'while loop in this case).



Processing the results of a query (cont.)

- The other command can get a row of results as an associative array. It takes a result as its first parameter, and an optional second parameter as a way to bind the results in the array.
- If `MYSQL_ASSOC` is specified, the results are indexed using the column names in the query. If `MYSQL_NUM` is specified, then the number starting at zero accesses the results. The default value `MYSQL_BOTH` returns an array with both types.

```
while ( $row = mysql_fetch_array($result, MYSQL_ASSOC) ) {  
    echo $row["title"] . '<br/>';  
}
```

- Using a statement like `'mysql_fetch_array($result, MYSQL_NUM)'` is essentially equivalent to the statement `'mysql_fetch_row($result)'` as they both return arrays stored with numerical indices.
- The `'mysql_fetch_array()'` command can be used to save memory by specifying, say, `MYSQL_ASSOC` instead of the default value.



Other useful PHP/SQL related functions

- The function `'mysql_data_seek($result, $value)'` can be used to move the internal result pointer (which is advanced automatically when commands like `'mysql_fetch_array()'` is called). This allows you, for example, to reprocess the results of a query (without having to access the database again). `'mysql_data_seek($result, 0)'` will reset the pointer to the initial row in the query result (but is an error if the query result was empty).
- `'mysql_num_rows ($result)'` returns the number of rows in the query result.
- `'mysql_affected_rows()'` gives the number of affected rows by the last `INSERT`, `DELETE`, `UPDATE`, or `REPLACE` query.
- `'mysql_insert_id()'` returns the ID generated by the `AUTO_INCREMENT` of the most recent query if there was one, or `'0'` if there was no `AUTO_INCREMENT` value.

