

Web Programming Step by Step

Lecture 23

Relational Databases and SQL; HTML Tables

Reading: 11.1 - 11.3; 2.2.2

References: [SQL syntax reference](#), [w3schools tutorial](#)

Except where otherwise noted, the contents of this presentation are Copyright 2009 Marty Stepp and Jessica Miller.



11.1: Database Basics

- 11.1: Database Basics
- 11.2: SQL
- 11.3: Databases and PHP
- 11.4: Multi-table Queries

Relational databases

- **relational database**: A method of structuring data as tables associated to each other by shared attributes.
- a table row corresponds to a unit of data called a **record**; a column corresponds to an attribute of that record
- relational databases typically use **Structured Query Language (SQL)** to define, manage, and search data

Why use a database? (11.1.1)

- **powerful**: can search it, filter data, combine data from multiple sources
- **fast**: can search/filter a database very quickly compared to a file
- **big**: scale well up to very large data sizes
- **safe**: built-in mechanisms for failure recovery (e.g. **transactions**)
- **multi-user**: concurrency features let many users view/edit data at same time
- **abstract**: provides layer of abstraction between stored data and app(s)
 - many database programs understand the same SQL commands

Database software

- Oracle
- Microsoft SQL Server (powerful) and Microsoft Access (simple)
- PostgreSQL (powerful/complex free open-source database system)
- SQLite (transportable, lightweight free open-source database system)
- MySQL (simple free open-source database system)
 - many servers run "LAMP" (Linux, Apache, MySQL, and PHP)
 - Wikipedia is run on PHP and MySQL
 - we will use MySQL in this course



Example simpsons database

students			teachers		courses			grades		
id	name	email	id	name	id	name	teacher_id	student_id	course_id	grade
123	Bart	bart@fox.com	1234	Krabappel	10001	Computer Science 142	1234	123	10001	B-
456	Milhouse	milhouse@fox.com	5678	Hoover	10002	Computer Science 143	5678	123	10002	C
888	Lisa	lisa@fox.com	9012	Stepp	10003	Computer Science 190M	9012	456	10001	B+
404	Ralph	ralph@fox.com			10004	Informatics 100	1234	888	10002	A+
								888	10003	A+
								404	10004	D+

Example world database (11.1.2)

countries

code	name	continent	independence_year	population	gnp	head_of_state	...
AFG	Afghanistan	Asia	1919	22720000	5976.0	Mohammad Omar	...
NLD	Netherlands	Europe	1581	15864000	371362.0	Beatrix	...
...

cities

id	name	country_code	district	population
3793	New York	USA	New York	8008278
1	Los Angeles	USA	California	3694820
...

languages

country_code	language	official	percentage
AFG	Pashto	T	52.4
NLD	Dutch	T	95.6
...

Example imdb database (11.1.2)

actors

id	first_name	last_name	gender
433259	William	Shatner	M
797926	Britney	Spears	F
831289	Sigourney	Weaver	F
...

movies

id	name	year	rank
112290	Fight Club	1999	8.5
209658	Meet the Parents	2000	7
210511	Memento	2000	8.7
...

roles

actor_id	movie_id	role
433259	313398	Capt. James T. Kirk
433259	407323	Sgt. T.J. Hooker
797926	342189	Herself
...

- also available, `imdb_small` with fewer records (for testing queries)
- other tables:
 - `directors` (`id`, `first_name`, `last_name`)
 - `movies_directors` (`director_id`, `movie_id`)
 - `movies_genres` (`movie_id`, `genre`)

11.2: SQL

- 11.1: Database Basics
- **11.2: SQL**
- 11.3: Databases and PHP
- 11.4: Multi-table Queries

SQL basics

```
SELECT name FROM cities WHERE id = 17;
```

SQL

```
INSERT INTO countries VALUES ('SLD', 'ENG', 'T', 100.0);
```

SQL

- **Structured Query Language (SQL)**: a language for searching and updating a database
- a standard syntax that is used by all database software (with minor incompatibilities)
 - generally case-insensitive
- a **declarative** language: describes what data you are seeking, not exactly how to find it

Issuing SQL commands directly in MySQL (11.2.1 - 11.2.2)

```
SHOW DATABASES;  
USE database;  
SHOW TABLES;
```

SQL

- SSH to Webster, then type:

```
$ mysql -u yourusername -p  
Password:  
Welcome to the MySQL monitor.  Commands end with ; or \g.  
  
mysql> USE world;  
Database changed  
  
mysql> SHOW TABLES;  
+-----+  
| cities  |  
| countries |  
| languages |  
+-----+  
3 rows in set (0.00 sec)
```

The SQL **SELECT** statement

```
SELECT column(s) FROM table;
```

SQL

```
SELECT name, code FROM countries;
```

SQL

name	code
China	CHN
United States	IND
Indonesia	USA
Brazil	BRA
Pakistan	PAK
...	...

- the **SELECT** statement searches a database and returns a set of results
 - the column name(s) written after SELECT filter which parts of the rows are returned
 - table and column names are case-sensitive
 - **SELECT * FROM *table*;** keeps all columns

The DISTINCT modifier

```
SELECT DISTINCT column(s) FROM table;
```

SQL

```
SELECT language  
FROM languages;
```

SQL

```
SELECT DISTINCT language  
FROM languages;
```

SQL

language
Dutch
English
English
Papiamentu
Spanish
Spanish
Spanish
...

language
Dutch
English
Papiamentu
Spanish
...

- eliminates duplicates from the result set

The WHERE clause

```
SELECT column(s) FROM table WHERE condition(s);
```

SQL

```
SELECT name, population FROM cities WHERE country_code = "FSM";
```

SQL

name	population
Weno	22000
Palikir	8600

- WHERE clause filters out rows based on their columns' data values
- in large databases, it's critical to use a WHERE clause to reduce the result set size
- suggestion: when trying to write a query, think of the FROM part first, then the WHERE part, and lastly the SELECT part

More about the WHERE clause

```
WHERE column operator value(s)
```

SQL

```
SELECT name, gnp FROM countries WHERE gnp > 2000000;
```

SQL

code	name	gnp
JPN	Japan	3787042.00
DEU	Germany	2133367.00
USA	United States	8510700.00
...

- the WHERE portion of a SELECT statement can use the following operators:
 - =, >, >=, <, <=
 - <> : not equal
 - BETWEEN *min* AND *max*
 - LIKE *pattern*
 - IN (*value, value, ..., value*)

Multiple WHERE clauses: AND, OR

```
SELECT * FROM cities WHERE code = 'USA' AND population >= 2000000;
```

SQL

id	name	country_code	district	population
3793	New York	USA	New York	8008278
3794	Los Angeles	USA	California	3694820
3795	Chicago	USA	Illinois	2896016
...

- multiple WHERE conditions can be combined using AND and OR

Approximate matches: LIKE

```
WHERE column LIKE pattern
```

SQL

```
SELECT code, name, population FROM countries WHERE name LIKE 'United%';
```

SQL

code	name	population
ARE	United Arab Emirates	2441000
GBR	United Kingdom	59623400
USA	United States	278357000
UMI	United States Minor Outlying Islands	0

- LIKE '*text*%' searches for text that starts with a given prefix
- LIKE '%*text*' searches for text that ends with a given suffix
- LIKE '%*text*%' searches for text that contains a given substring

Sorting by a column: ORDER BY

```
ORDER BY column(s)
```

SQL

```
SELECT code, name, population FROM countries  
WHERE name LIKE 'United%' ORDER BY population;
```

SQL

code	name	population
UMI	United States Minor Outlying Islands	0
ARE	United Arab Emirates	2441000
GBR	United Kingdom	59623400
USA	United States	278357000

- can write ASC or DESC to sort in ascending (default) or descending order:

```
SELECT * FROM countries ORDER BY population DESC;
```

SQL

- can specify multiple orderings in decreasing order of significance:

```
SELECT * FROM countries ORDER BY population DESC, gnp;
```

SQL

Limiting rows: LIMIT

LIMIT *number*

SQL

```
SELECT name FROM cities WHERE name LIKE 'K%' LIMIT 5;
```

SQL

name
Kabul
Khulna
Kingston upon Hull
Koudougou
Kafr al-Dawwar

- also useful as a sanity check to make sure your query doesn't return 10^7 rows

11.3: Databases and PHP

- 11.1: Database Basics
- 11.2: SQL
- **11.3: Databases and PHP**
- 11.4: Multi-table Queries

PHP MySQL functions

name	description
<code>mysql_connect</code>	connects to a database server
<code>mysql_select_db</code>	chooses which database on server to use (similar to SQL USE <i>database</i> ; command)
<code>mysql_query</code>	performs a SQL query on the database
<code>mysql_real_escape_string</code>	encodes a value to make it safe for use in a query
<code>mysql_fetch_array,...</code>	returns the query's next result row as an associative array
<code>mysql_close</code>	closes a connection to a database

Complete PHP MySQL example

```
# connect to world database on local computer
$db = mysql_connect("localhost", "traveler", "packmybags");
mysql_select_db("world");

# execute a SQL query on the database
$results = mysql_query("SELECT * FROM countries WHERE population > 10000000");

# loop through each country
while ($row = mysql_fetch_array($results)) {
    ?>
    <li> <?= $row["name"] ?>, ruled by <?= $row["head_of_state"] ?> </li>
    <?php
}
?>
```

PHP

Connecting to MySQL: `mysql_connect` (11.3.1)

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

PHP

```
# connect to world database on local computer  
mysql_connect("localhost", "traveler", "packmybags");  
mysql_select_db("world");
```

PHP

- `mysql_connect` opens connection to database on its server
 - any/all of the 3 parameters can be omitted (default: localhost, anonymous)
- `mysql_select_db` sets which database to examine

Performing queries: `mysql_query` (11.3.2)

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

```
$results = mysql_query("SQL query");  
...
```

PHP

```
$results = mysql_query("SELECT * FROM cities WHERE code = 'USA'  
                        AND population >= 2000000;");
```

PHP

- `mysql_query` sends a SQL query to the database
- returns a special result-set object that you don't interact with directly, but instead pass to later functions
- SQL queries are in " ", end with ;, and nested quotes can be ' or \"

Result rows: `mysql_fetch_array`

```
mysql_connect("host", "username", "password");
mysql_select_db("database name");
$results = mysql_query("SQL query");

while ($row = mysql_fetch_array($results)) {
    do something with $row;
}
```

PHP

- `mysql_fetch_array` returns one result row as an associative array
 - the column names are its keys, and each column's values are its values
 - example: `$row["population"]` gives the population from that row of the results

Error-checking: `mysql_error` (11.3.3)

```
if (!mysql_connect("localhost", "traveler", "packmybags")) {
    die("SQL error occurred on connect: " . mysql_error());
}
if (!mysql_select_db("world")) {
    die("SQL error occurred selecting DB: " . mysql_error());
}
$query = "SELECT * FROM countries WHERE population > 100000000;";
$results = mysql_query($query);
if (!$results) {
    die("SQL query failed:\n$query\n" . mysql_error());
}
```

PHP

- SQL commands can fail: database down, bad password, bad query, ...
- for debugging, always test the results of PHP's `mysql` functions
 - if they fail, stop script with `die` function, and print `mysql_error` result to see what failed
 - give a descriptive error message and also print the query, if any

Complete example w/ error checking

```
# connect to world database on local computer
check(mysql_connect("localhost", "traveler", "packmybags"), "connect");
check(mysql_select_db("world"), "selecting db");

# execute a SQL query on the database
$query = "SELECT * FROM countries WHERE population > 100000000;";
$results = mysql_query($query);
check($results, "query of $query");

# loop through each country
while ($row = mysql_fetch_array($results)) {
    ?>
    <li> <?= $row["name"] ?>, ruled by <?= $row["head_of_state"] ?> </li>
    <?php
}

# makes sure result is not false/null; else prints error
function check($result, $message) {
    if (!$result) {
        die("SQL error during $message: " . mysql_error());
    }
}
?>
```

PHP

Other MySQL PHP functions

name	description
mysql_num_rows	returns number of rows matched by the query
mysql_num_fields	returns number of columns per result in the query
mysql_list_dbs	returns a list of databases on this server
mysql_list_tables	returns a list of tables in current database
mysql_list_fields	returns a list of fields in the current data
complete list	

HTML tables: `<table>`, `<tr>`, `<td>`

A 2D table of rows and columns of data (block element)

```
<table>
  <tr><td>1,1</td><td>1,2 okay</td></tr>
  <tr><td>2,1 real wide</td><td>2,2</td></tr>
</table>
```

HTML

```
1,1      1,2 okay
2,1 real wide 2,2
```

output

- `table` defines the overall table, `tr` each row, and `td` each cell's data
- tables are useful for displaying large row/column data sets
- NOTE: tables are sometimes used by novices for web page layout, but this is not proper semantic HTML and should be avoided

Table headers, captions: `<th>`, `<caption>`

```
<table>
  <caption>My important data</caption>
  <tr><th>Column 1</th><th>Column 2</th></tr>
  <tr><td>1,1</td><td>1,2 okay</td></tr>
  <tr><td>2,1 real wide</td><td>2,2</td></tr>
</table>
```

HTML

```
My important data
Column 1 Column 2
1,1      1,2 okay
2,1 real wide 2,2
```

output

- `th` cells in a row are considered headers; by default, they appear bold
- a `caption` at the start of the table labels its meaning

Styling tables (3.2.6)

```
table { border: 2px solid black; caption-side: bottom; }  
tr { font-style: italic; }  
td { background-color: yellow; text-align: center; width: 30%; }
```

CSS

<i>Column 1</i>	<i>Column 2</i>
<i>1,1</i>	<i>1,2 okay</i>
<i>2,1 real wide</i>	<i>2,2</i>

My important data

output

- all standard CSS styles can be applied to a table, row, or cell
- table specific CSS properties:
 - `border-collapse`, `border-spacing`, `caption-side`, `empty-cells`, `table-layout`

The `border-collapse` property

```
table, td, th { border: 2px solid black; }  
table { border-collapse: collapse; }
```

CSS

Without `border-collapse`

Column 1	Column 2
1,1	1,2
2,1	2,2

With `border-collapse`

Column 1	Column 2
1,1	1,2
2,1	2,2

- by default, the overall table has a separate border from each cell inside
- the `border-collapse` property merges these borders into one

The rowspan and colspan attributes

```
<table>
  <tr><th>Column 1</th><th>Column 2</th><th>Column 3</th></tr>
  <tr><td colspan="2">1,1-1,2</td>
    <td rowspan="3">1,3-3,3</td></tr>
  <tr><td>2,1</td><td>2,2</td></tr>
  <tr><td>3,1</td><td>3,2</td></tr>
</table>
```

HTML

Column 1	Column 2	Column 3
1,1-1,2		1,3-3,3
2,1	2,2	
3,1	3,2	

output

- colspan makes a cell occupy multiple columns; rowspan multiple rows
- text-align and vertical-align control where the text appears within a cell

Column styles: <col>, <colgroup>

```
<table>
  <col class="urgent" />
  <colgroup class="highlight" span="2"></colgroup>

  <tr><th>Column 1</th><th>Column 2</th><th>Column 3</th></tr>
  <tr><td>1,1</td><td>1,2</td><td>1,3</td></tr>
  <tr><td>2,1</td><td>2,2</td><td>2,3</td></tr>
</table>
```

HTML

Column 1	Column 2	Column 3
1,1	1,2	1,3
2,1	2,2	2,3

output

- col tag can be used to define styles that apply to an entire column (self-closing)
- colgroup tag applies a style to a group of columns (NOT self-closing)

Don't use tables for layout!

- (borderless) tables appear to be an easy way to achieve grid-like page layouts
 - many "newbie" web pages do this (including many UW CSE web pages...)
 - but, a `table` has semantics; it should be used only to represent an actual table of data
 - instead of tables, use `divs`, widths/margins, floats, etc. to perform layout
-

- tables should not be used for layout!
- Tables should not be used for layout!!
- TABLES SHOULD NOT BE USED FOR LAYOUT!!!
- **TABLES SHOULD NOT BE USED FOR LAYOUT!!!!**