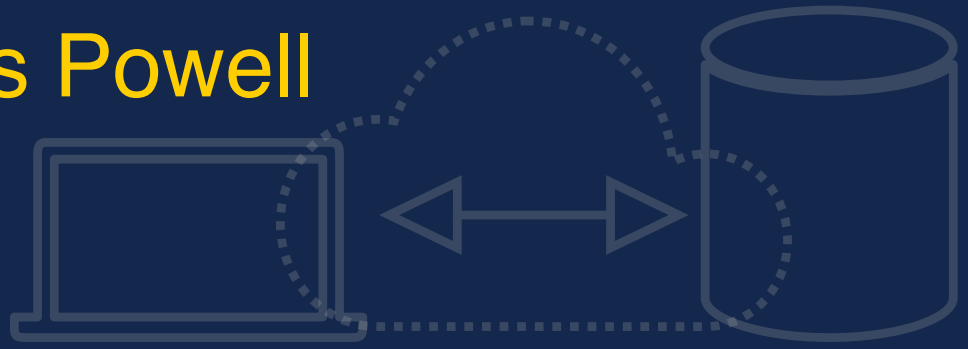
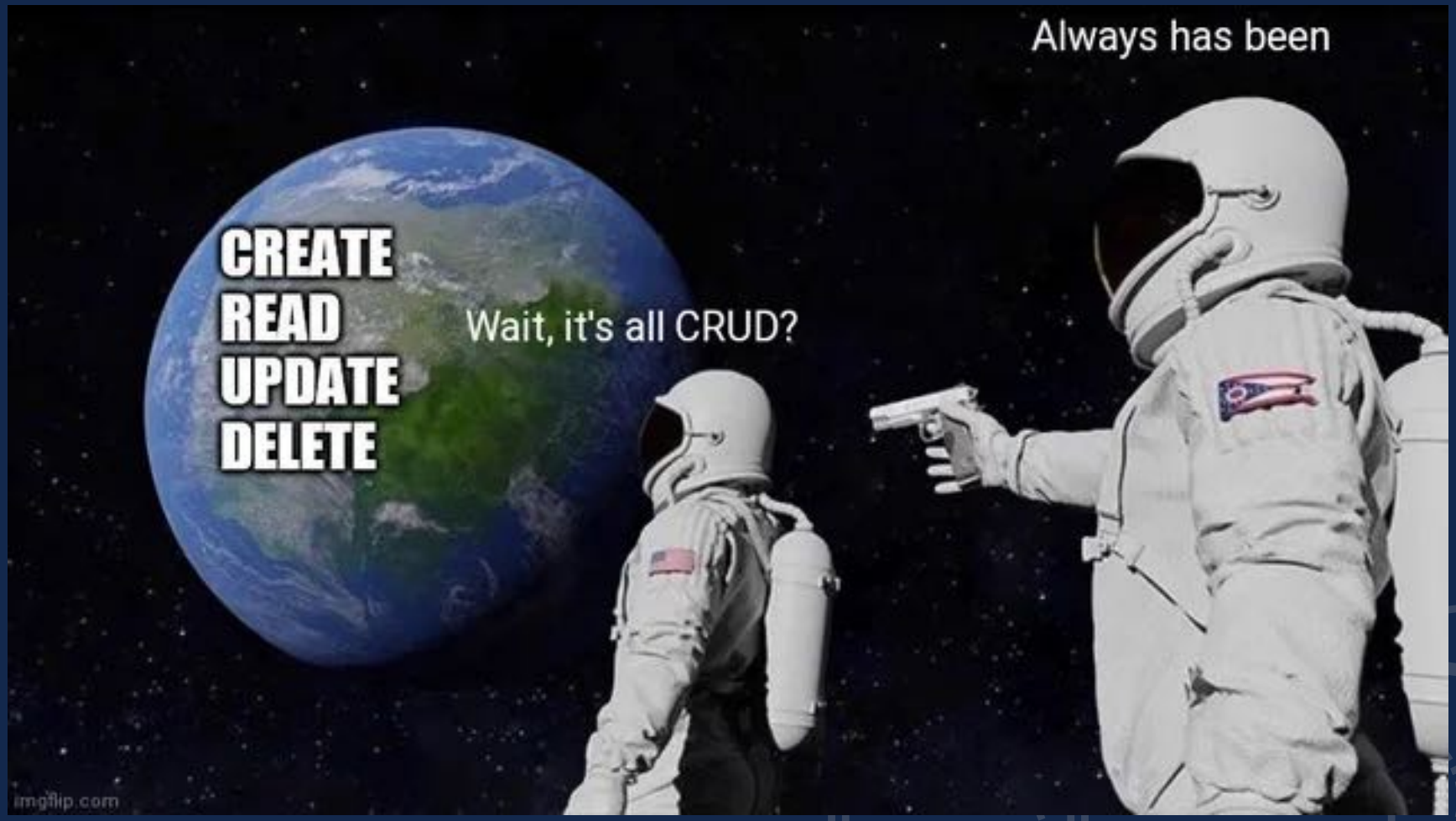
Database Driven Sites
Part I: MySQL

with Thomas Powell



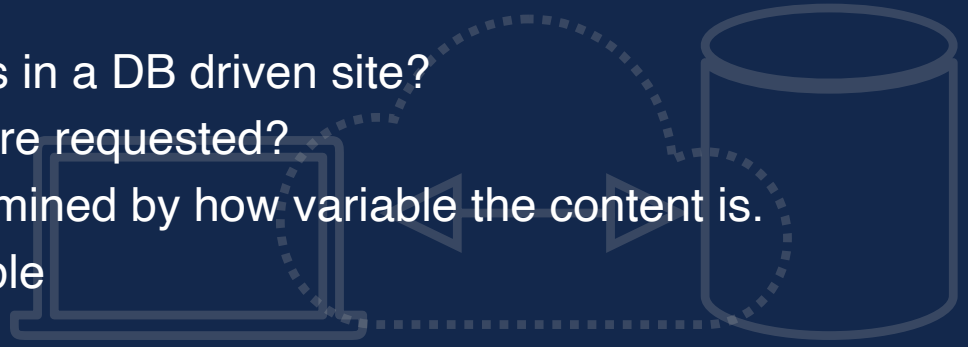
Today's Meme Sponsor



DB Driven Sites

- Rather than storing the content of our Web site directly within files it may be better to store the content within a datastore (XML files, Database tables, etc.)
  - We will then pull out the desired data from the database and combine it with HTML markup and CSS in the form of a template to create a complete page
    - Separation of code, presentation, and data is a good aim – though not unique to server-side dev.
  - The DB driven site provides for the possibility of dynamic or even personalized pages
    - User preferences, promotions in e-commerce, custom reports, portals, etc.
  - DB driven sites also afford us the opportunity to build content management systems (CMS)

- Important Idea – When do you build the pages in a DB driven site?
    - As they are requested? Before they are requested?
    - Your ability to prebuild pages is determined by how variable the content is. Avoid the dynamic-static trap if possible

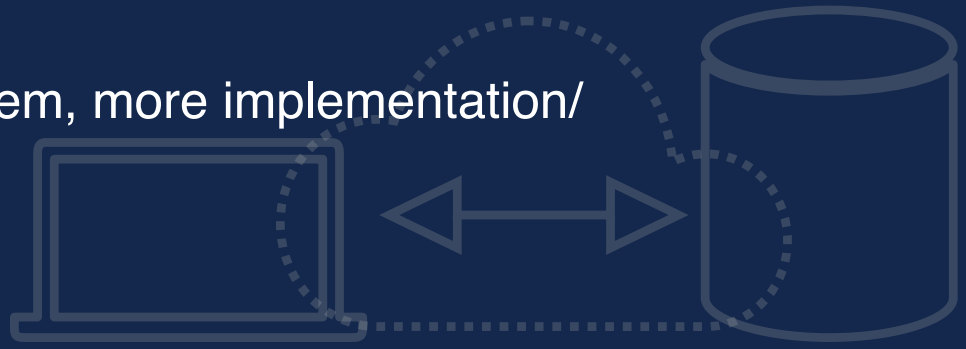


The pieces of the DB Driven Site

- The DB
  - Oracle, MS SQL Server, mySQL, MongoDB, Redis, etc.
- Programming Environment
  - PHP, Node.js, Python, ASP.NET, JSP, ASP, CFM, CGI/Perl, etc.
  - APIs and Drivers for DB Connectivity
    - Abstraction layers for DB
      - ODBC, native drivers, ADO.NET, JDBC
    - Question: How often will the data store change?
    - The mapping between table relations and objects also may be handled here – ORM (Object-relational mapping)
  - Separate Templating Environment
    - EJS, Twig, Smarty, Handlebars, etc.

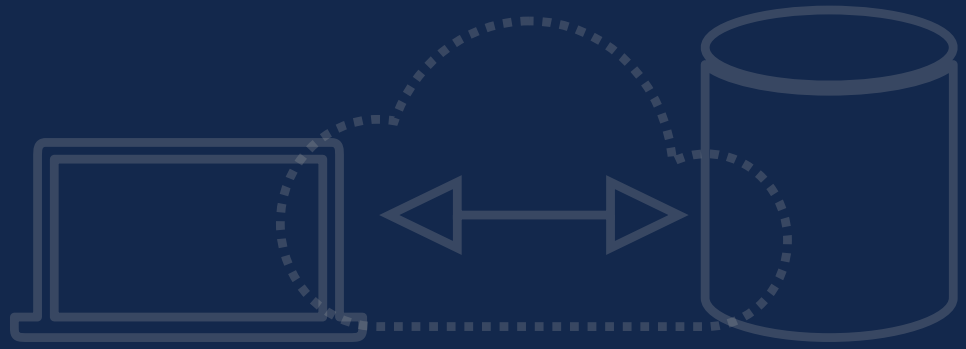
The Choice

- Mostly server-rendered (SSR) - aka traditional MVC on the server where fully built HTML/CSS pages returned
- Mostly client-rendered (SPA style) where you hit an end-point (REST, GraphQL) and fetch data (JSON) which then composited client-side into the resulting HTML/CSS
- Ultimately the client focused will win but it requires some difficulty to do it properly and assumes appropriate client devices
  - Actual testing has shown many “modern” solutions significantly slower than plain boring old SSR style sites
  - Sadly this is less an architecture problem, more implementation/ practitioner problem



Approach Redux

- User driven
  - UI first
- Data driven
  - Tables and data first
- Top down vs. bottom up?
- Middle out the framework / code
- All are intertwined the answer really is all of the above at the correct times.



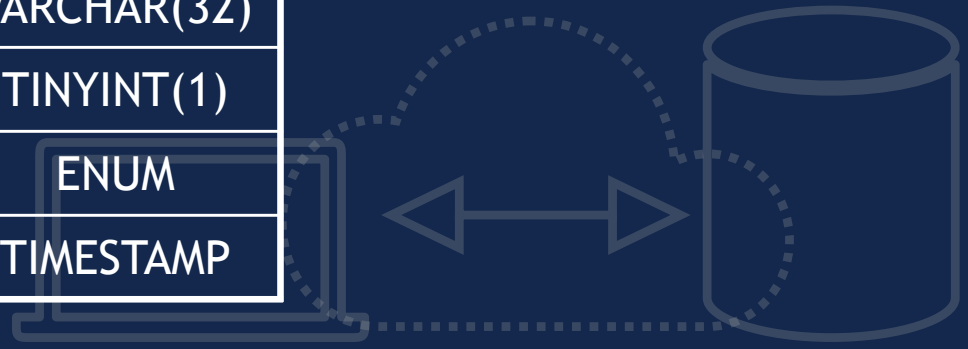
MEME Break for Repeat Glory!

Step 1: Model Your Data

- Once you have your columns and their data types selected, the scaffold for your table in MySQL might look something like this

User Table

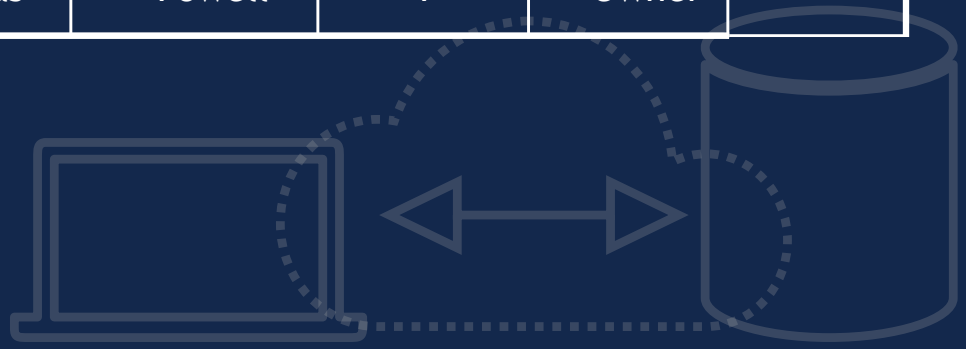
id (primary)	MEDIUMINT
username	VARCHAR(32)
email	VARCHAR(32)
password	VARCHAR(32)
first_name	VARCHAR(32)
last_name	VARCHAR(32)
admin	TINYINT(1)
role	ENUM
timestamp	TIMESTAMP



Step 1: Model Your Data

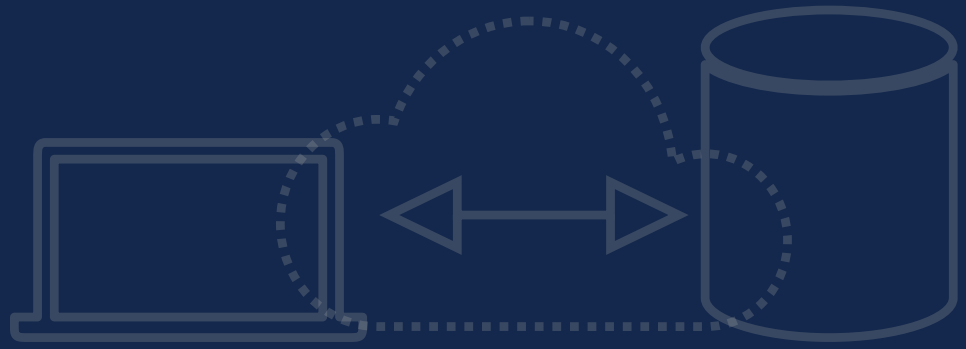
- It is often easier to visualize the data like so
- We could imagine a filled in set of data looking something like
- Warning: see BIG problem?

id	username	email	password	first_name	last_name	admin	role	time stamp
347	lsimpson	l@simpson.com	superSafe	Lisa	Simpson	0	Guest	8/25/20
348	bgates	b@gates.com	p4$$w0rd	Bill	Gates	0	Dev	8/25/20
349	tpowell	t@powell.com	1L0v3p1nt	Thomas	Powell	1	Owner	8/25/20



Step 1: Model Your Data

- Very often you will not have a single table but a collection of tables
- In this case you generally need to have something that relates data between the tables, typically this is the primary key

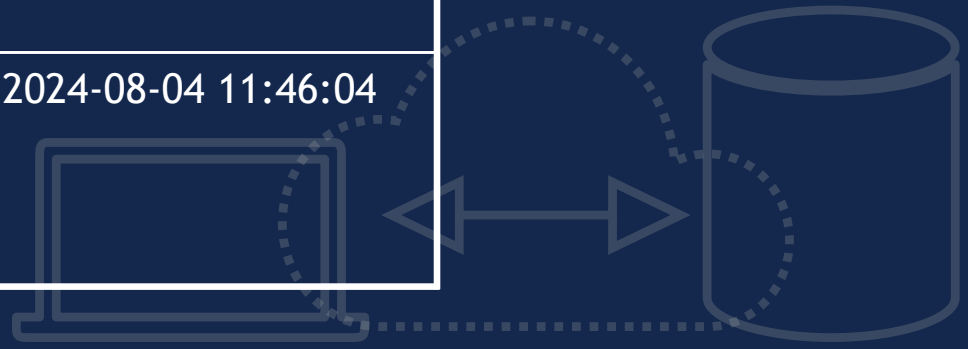


Step 1: Model Your Data

- Given our last example we might imagine a table called “Log” that has the system’s activity log.

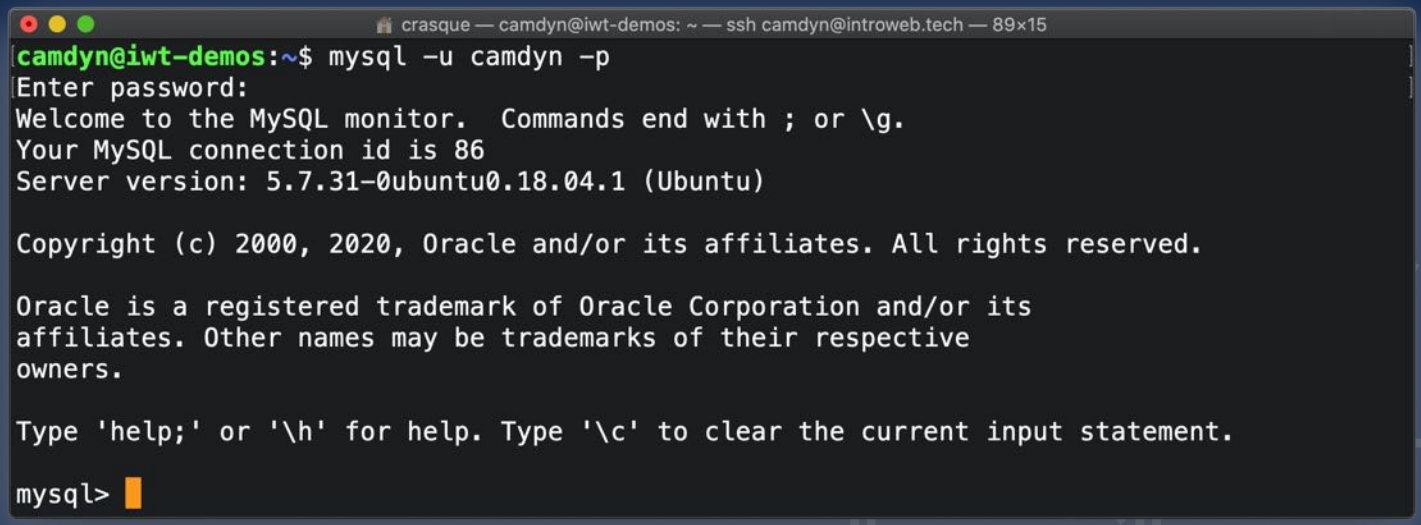
Log Table

id	Activity	Time
347	Login-fail	2024-08-04 11:46:04
347	Login-fail	2024-08-04 11:46:04
348	Login	2024-08-04 11:46:04
347	Login	2024-08-04 11:46:04
348	Logout	2024-08-04 11:46:04



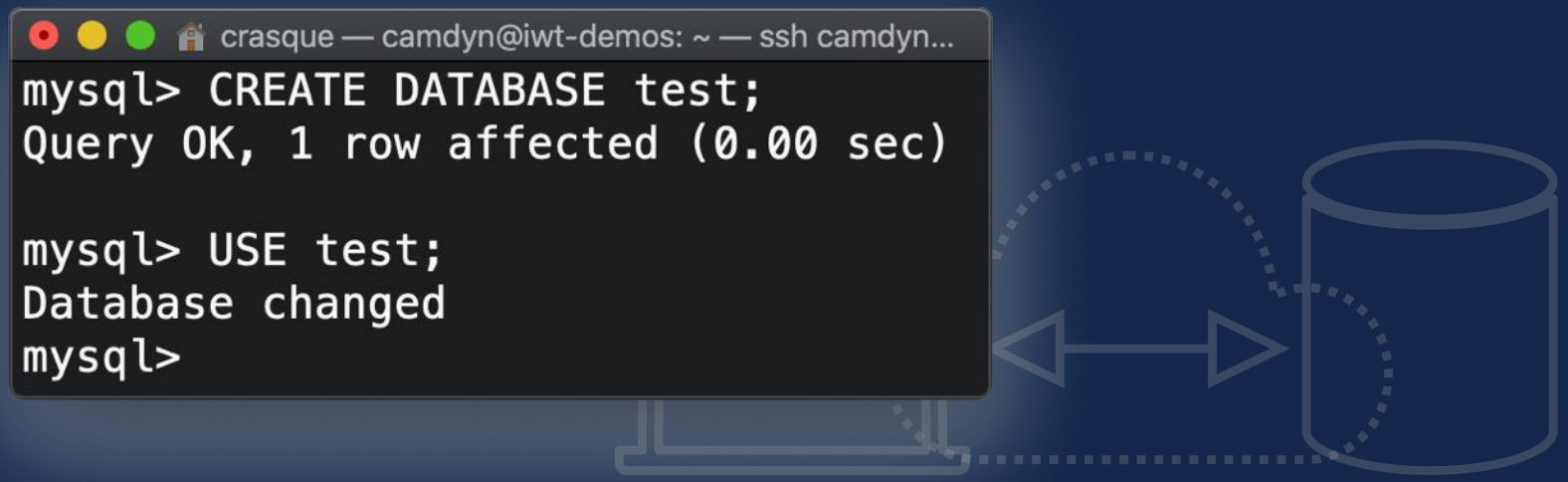
Step 2 – Create the Database and Tables

- If you have shell access to a server you might use the mysql monitor to admin the database
- mysql –h *hostname* –u *username* –p *password*
  - Hostname will be localhost in the case you run this on the same box



Step 2 – Create the Database and Tables

- Once you are connected you normally would create a database
  - mysql> CREATE DATABASE *databasename;*
- In this situation we have a database specified already, so just start using it.
  - mysql> USE *databasename*;

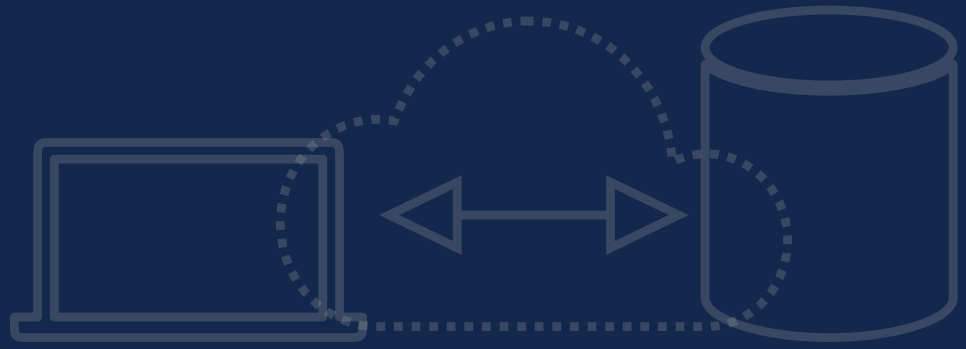


Step 2 – Create the Database and Tables

- Now create the table(s) you want based upon your defined schema
- CREATE TABLE users (

```
id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
username  VARCHAR(32) NOT NULL,
email VARCHAR(32) NOT NULL,
password VARCHAR(32) NOT NULL,
first_name VARCHAR(32) NOT NULL,
last_name VARCHAR(32) NOT NULL,
admin TINYINT(1) NOT NULL,
role ENUM(“Guest”, “Dev”, “Owner”) NOT NULL,
timestamp  TIMESTAMP NOT NULL
);
```

- Watch out for returns and syntax!!



Step 2 – Create the Database and Tables

crasque — camdyn@iwt-demos: ~ — ssh camdyn@introweb.tech — 72×13

```
mysql> CREATE TABLE users (
    -> id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY,
    -> username VARCHAR(32) NOT NULL,
    -> email VARCHAR(32) NOT NULL,
    -> password VARCHAR(32) NOT NULL,
    -> first_name VARCHAR(32) NOT NULL,
    -> last_name VARCHAR(32) NOT NULL,
    -> admin TINYINT(1) NOT NULL,
    -> role ENUM('Guest', 'Dev', 'Owner'),
    -> timestamp TIMESTAMP NOT NULL);
Query OK, 0 rows affected (0.04 sec)

mysql>
```

Showing the Results

- Once using a DB issue SHOW TABLES; statement (left)
- To look deeper at a table use SHOW COLUMNS from *tablename;* statement (right)

```
crasque — camdyn@iwt-d...
mysql> SHOW TABLES;
+----------------+
| Tables_in_test |
+----------------+
| users          |
+----------------+
1 row in set (0.00 sec)

mysql>
```

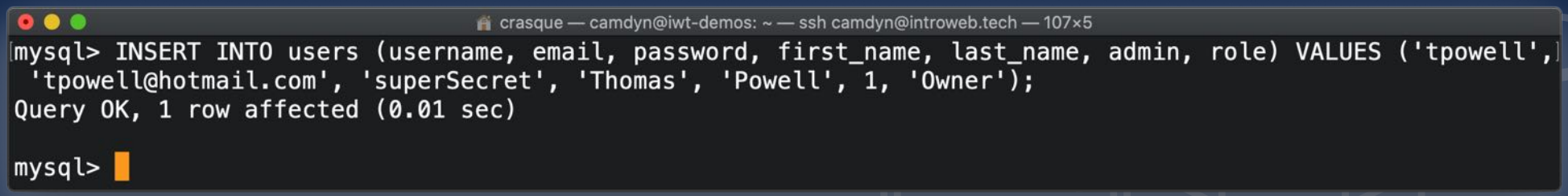
```
crasque — camdyn@iwt-demos: ~ — ssh camdyn@introweb.tech — 107×17
mysql> SHOW COLUMNS FROM users;
+------------+-----------------------------+------+-----+-------------------+-----------------------------+
| Field      | Type                        | Null | Key | Default           | Extra                       |
+------------+-----------------------------+------+-----+-------------------+-----------------------------+
| id         | mediumint(8) unsigned       | NO   | PRI | NULL              | auto_increment              |
| username   | varchar(32)                 | NO   |     | NULL              |                             |
| email      | varchar(32)                 | NO   |     | NULL              |                             |
| password   | varchar(32)                 | NO   |     | NULL              |                             |
| first_name | varchar(32)                 | NO   |     | NULL              |                             |
| last_name  | varchar(32)                 | NO   |     | NULL              |                             |
| admin      | tinyint(1)                  | NO   |     | NULL              |                             |
| role       | enum('Guest','Dev','Owner') | YES  |     | NULL              |                             |
| timestamp  | timestamp                   | NO   |     | CURRENT_TIMESTAMP | on update CURRENT_TIMESTAMP |
+------------+-----------------------------+------+-----+-------------------+-----------------------------+
9 rows in set (0.00 sec)

mysql>
```

Inserting Data

- Use the INSERT statement with the following syntax:
  - INSERT INTO *tablename* (*column* 1,…,*column* n)

    VALUES (‘*value* 1’,…,*’value* n’);

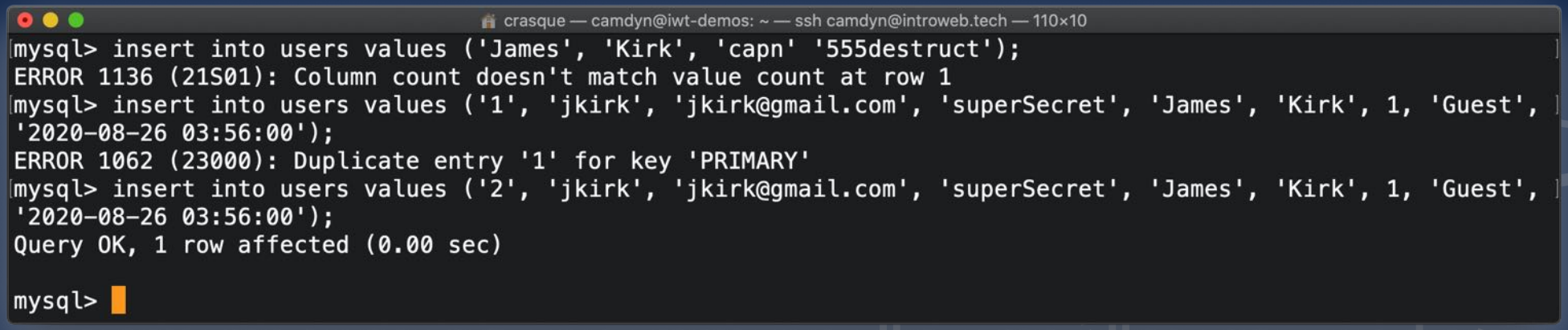
  - Note: The 1 – n indication just indicates a number of columns to be used, the columns do not have to be contiguous
  - INSERT INTO users (username, email, password, first_name, last_name, admin, role) VALUES (‘tpowell’, ‘tpowell@hotmail.com', ‘superSecret’, ‘Thomas’, ‘Powell’, 1, ‘Owner’);



Inserting Data

- You can also use a simple syntax like
  - INSERT INTO tablename VALUES (‘value1’,…’value n’);

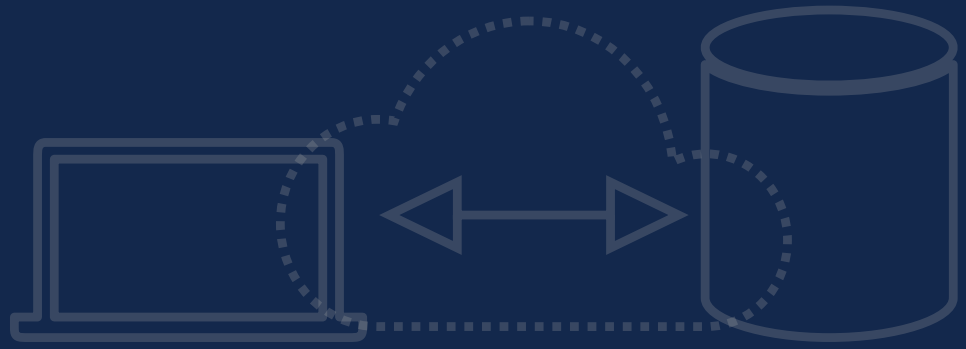
  - Note: In this case you must specify values for all columns in the table, often NULL will be used.
  - Note: In case of auto-increment values you still have to add something.



Inserting Data

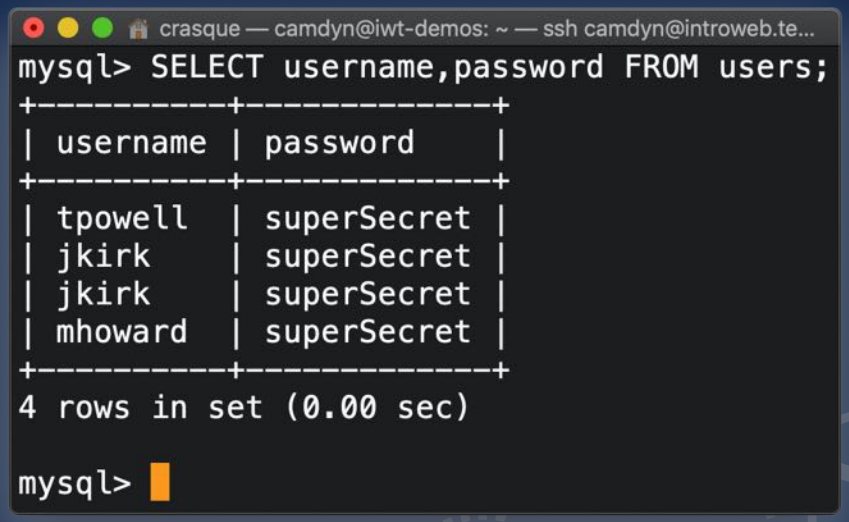
- It is also possible to insert multiple records at once

```
crasque — camdyn@iwt-demos: ~ — ssh camdyn@introweb.tech — 103×7
mysql> insert into users values ('3', 'jkirk', 'jkirk@gmail.com', 'superSecret', 'James', 'Kirk', 1, 'G
uest', '2020-08-26 03:56:00'), ('4', 'mhoward', 'mhoward@gmail.com', 'superSecret', 'Moe', 'Howard', 0,
 'Guest', '2020-08-26 03:57:00');
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

mysql>
```



Selecting Data

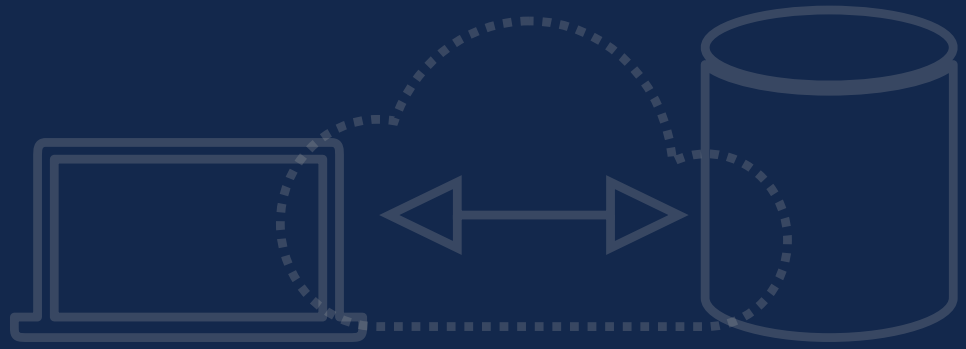
- SELECT used to retrieve info from a table.
  - Syntax: SELECT *columns* FROM *table*



Selecting Data

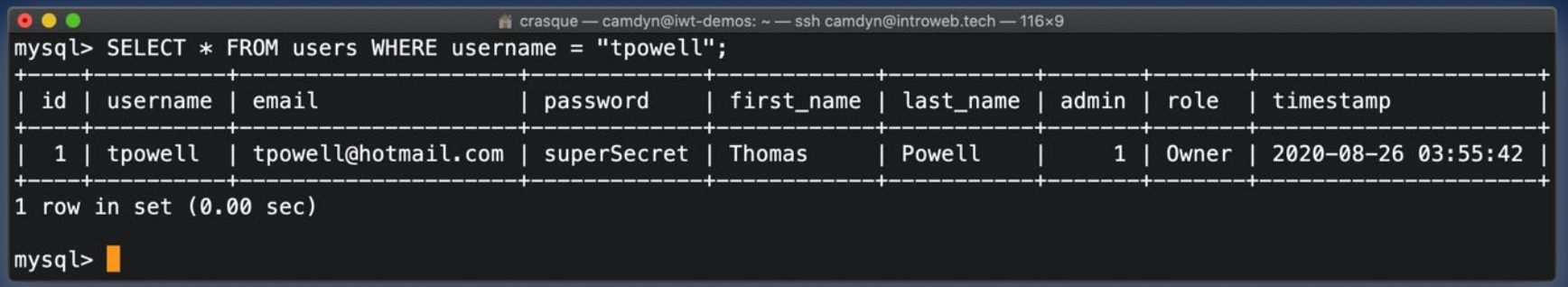
crasque — camdyn@iwt-demos: ~ — ssh camdyn@introweb.tech — 116×12

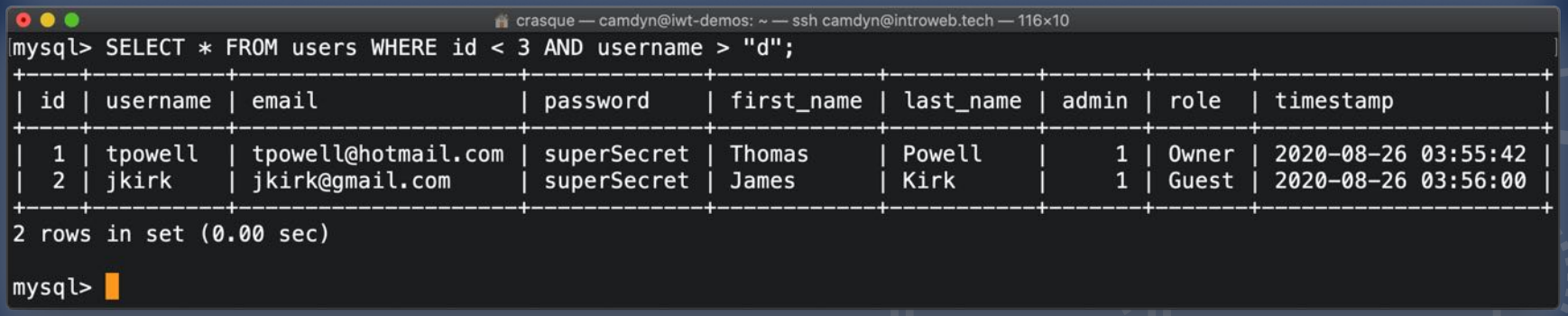
```
mysql> SELECT * FROM users;
+----+----------+---------------------+------------+------------+-----------+-------+-------+---------------------+
| id | username | email               | password   | first_name | last_name | admin | role  | timestamp           |
+----+----------+---------------------+------------+------------+-----------+-------+-------+---------------------+
|  1 | tpowell  | tpowell@hotmail.com | superSecret | Thomas     | Powell    |     1 | Owner | 2020-08-26 03:55:42 |
|  2 | jkirk    | jkirk@gmail.com     | superSecret | James      | Kirk      |     1 | Guest | 2020-08-26 03:56:00 |
|  3 | jkirk    | jkirk@gmail.com     | superSecret | James      | Kirk      |     1 | Guest | 2020-08-26 03:56:00 |
|  4 | mhoward  | mhoward@gmail.com   | superSecret | Moe        | Howard    |     0 | Guest | 2020-08-26 03:57:00 |
+----+----------+---------------------+------------+------------+-----------+-------+-------+---------------------+
4 rows in set (0.00 sec)

mysql>
```



More Selecting Data

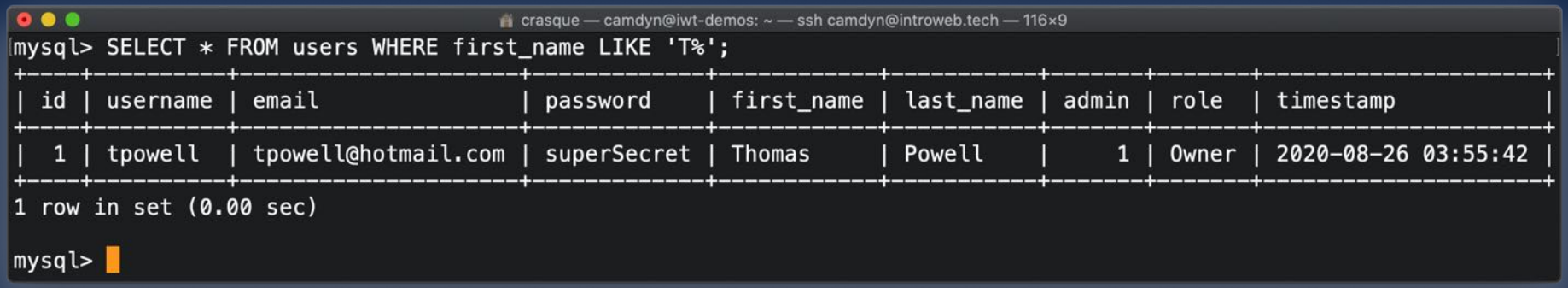
- Add the WHERE conditional to limit a selection. You can use typical operators like =,<,>,<=,>=, !=, AND (&&), OR (II), and NOT (!)
  - Special operators like IS NOT NULL, IS NULL, BETWEEN, and NOT BETWEEN are also available under mySQL

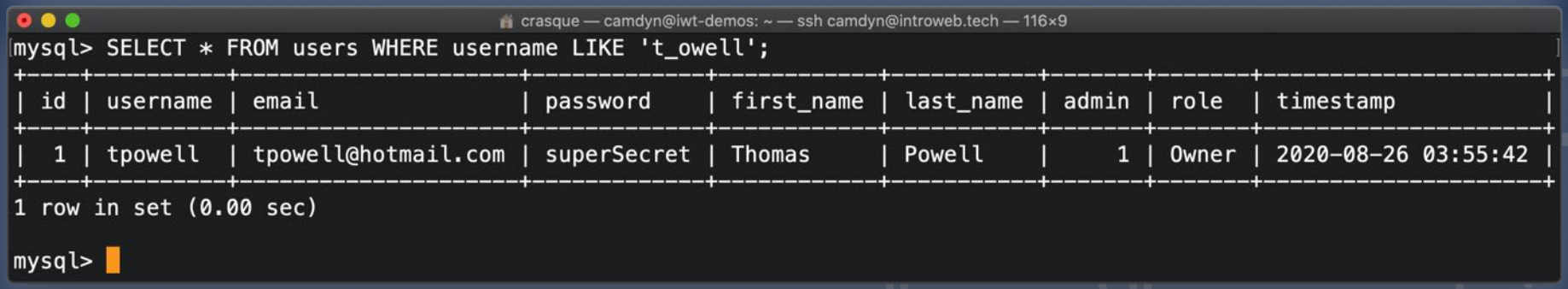




More Selecting Data

- In the case of strings you often need to use LIKE and NOT LIKE in conjunction with a pattern match of either _ (a single character match) or % which matches zero or more characters





Sorting Data

- Add ORDER BY to a SELECT statement
  - SELECT * from *table* ORDER BY *column*;
  - Attach ASC for ascending (default) or DESC for descending after the column
  - Join multiple columns together with commas

Updating Data

- **UPDATE** *table* **SET** *column*=‘*value*’;
- **UPDATE** *table* **SET** *column1*=‘*value1*’,…*columnN*=‘*valueN*’;
- **UPDATE** *table* **SET** *column*=‘*value*’ **WHERE** *column2*=‘*value2*’;

crasque — camdyn@iwt-demos: ~ — ssh camdyn@introweb.tech — 116×13

```
mysql> UPDATE users SET username="captain" WHERE id=2;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0

mysql> SELECT * FROM users WHERE id=2;
+----+----------+-----------------+-------------+------------+-----------+-------+-------+---------------------+
| id | username | email           | password    | first_name | last_name | admin | role  | timestamp           |
+----+----------+-----------------+-------------+------------+-----------+-------+-------+---------------------+
|  2 | captain  | jkirk@gmail.com | superSecret | James      | Kirk      |     1 | Guest | 2020-08-26 04:24:50 |
+----+----------+-----------------+-------------+------------+-----------+-------+-------+---------------------+
1 row in set (0.00 sec)

mysql>
```

Deleting Data

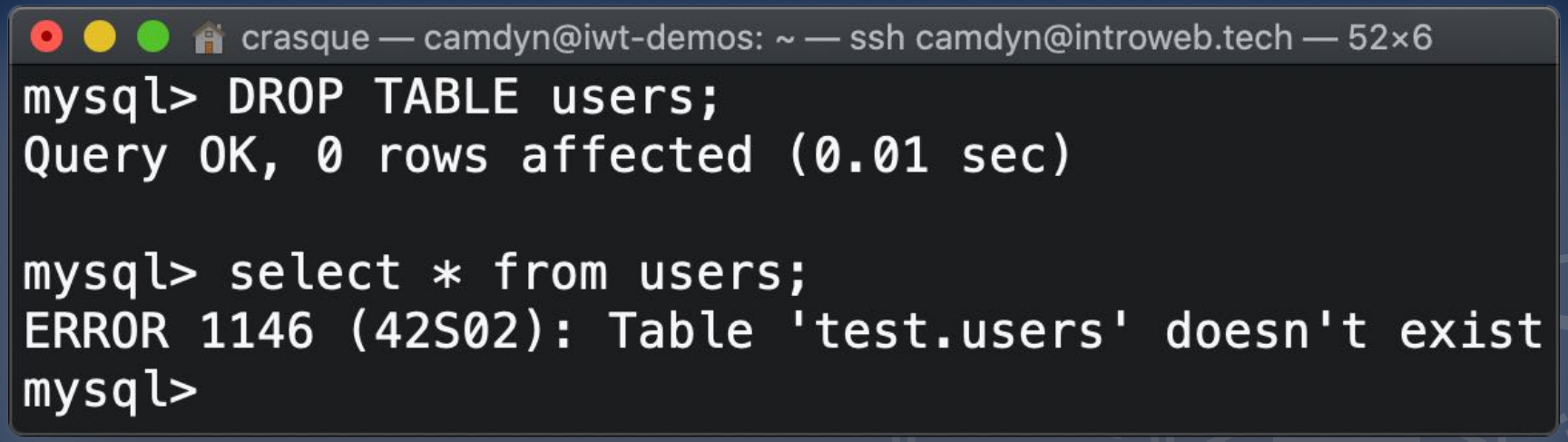
- DELETE FROM *table* WHERE *column*='*value*';
  - Avoid **DELETE FROM** *table*; as it would blow out the whole table

crasque — camdyn@iwt-demos: ~ — ssh camdyn@introweb.tech — 116×14

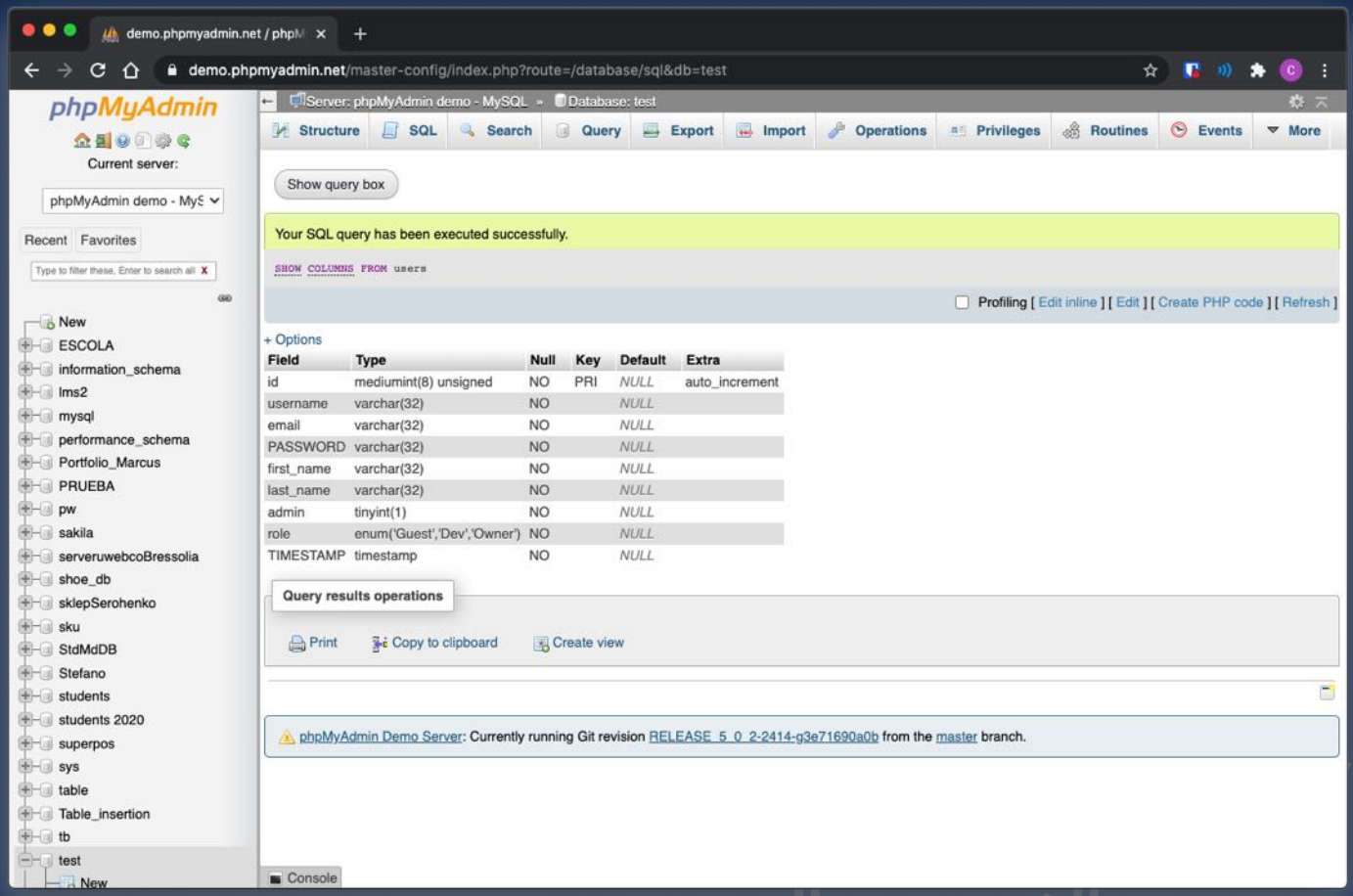
```
mysql> DELETE FROM users WHERE id=3;
Query OK, 1 row affected (0.01 sec)

mysql> SELECT * FROM users;
+----+----------+---------------------+-------------+------------+-----------+-------+-------+---------------------+
| id | username | email               | password    | first_name | last_name | admin | role  | timestamp           |
+----+----------+---------------------+-------------+------------+-----------+-------+-------+---------------------+
|  1 | tpowell  | tpowell@hotmail.com | superSecret | Thomas     | Powell    |     1 | Owner | 2020-08-26 03:55:42 |
|  2 | captain  | jkirk@gmail.com     | superSecret | James      | Kirk      |     1 | Guest | 2020-08-26 04:24:50 |
|  4 | mhoward  | mhoward@gmail.com   | superSecret | Moe        | Howard    |     0 | Guest | 2020-08-26 03:57:00 |
+----+----------+---------------------+-------------+------------+-----------+-------+-------+---------------------+
3 rows in set (0.00 sec)

mysql>
```

Deleting Data – Big Changes

- DROP TABLE tablename; is the appropriate way to destroy a table and its contents
- DROP DATABASE databasename; allows you to drop a whole DB (tables and all) – be careful in some shared environments as you may not have permissions to make a replacement database

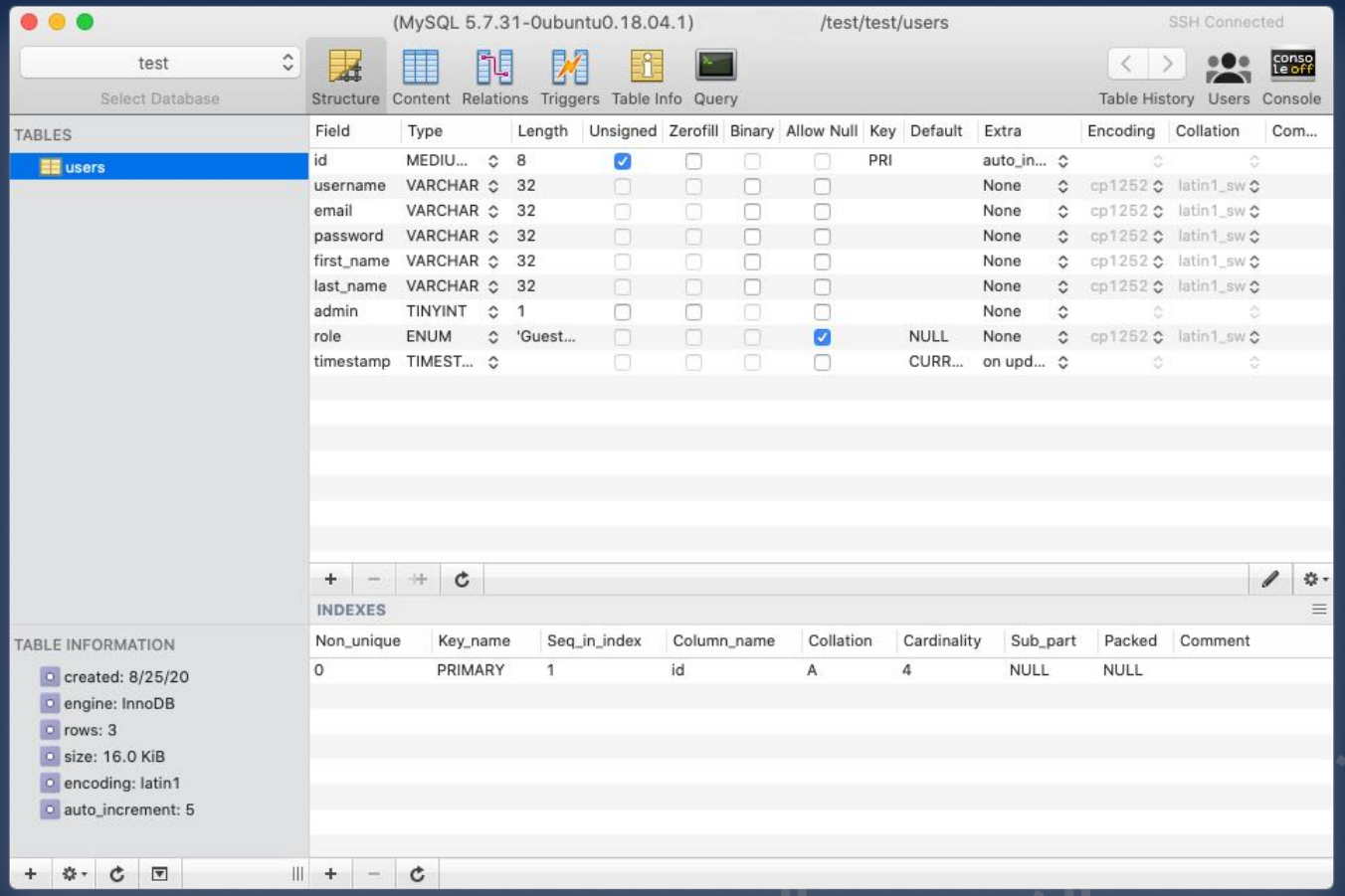


An Easier Way to Admin MySQL



http://www.phpmyadmin.net

Easier Way - GUI Client



https://sequelpro.com/download

Appropriate Meme?

Talking to mySQL from PHP

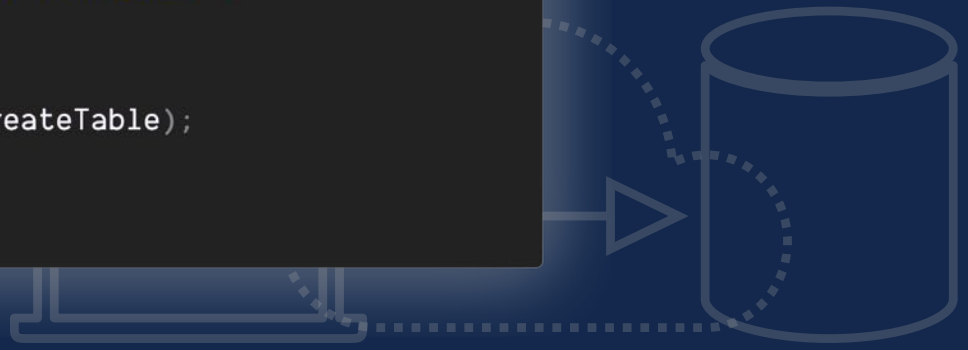
- First you need to connect with **`mysqli_connect`** and the select the database using **`mysqli_select_db`**
  - Add in error checking to be on the safe side

```php
<?php
$db_user = "root";
$db_password = "superSecret";
$db_host = "localhost";
$db_name = "mydatabase";

$dbc = mysqli_connect($db_host, $db_user, $db_password);

if (!$dbc) {
    die("Could not connect to MySQL: " . mysqli_connect_error());
}

mysqli_select_db($dbc, $db_name);

/* now you can do various statements to create tables, insert data, preform queries */
?>
```

Talking to mySQL from PHP

- Following our last example in the mySQL monitor let's create a table

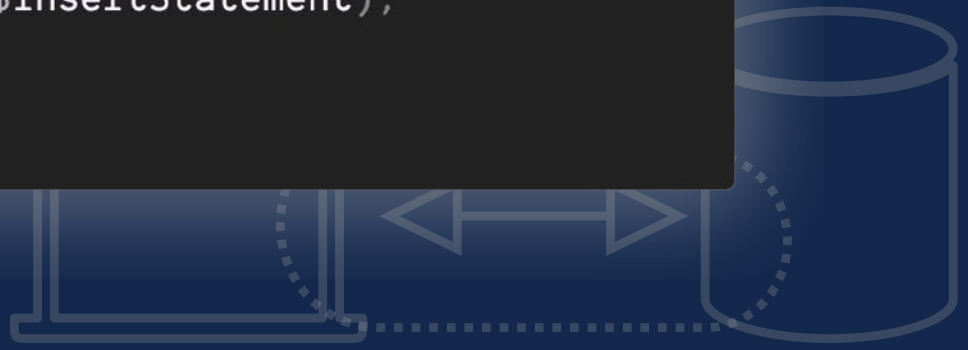
```php
<?php
$createTable = "CREATE TABLE users (" .
    "id MEDIUMINT UNSIGNED NOT NULL AUTO_INCREMENT PRIMARY KEY," .
    "username VARCHAR(32) NOT NULL," .
    "email VARCHAR(32) NOT NULL," .
    "password VARCHAR(32) NOT NULL," .
    "first_name VARCHAR(32) NOT NULL," .
    "last_name VARCHAR(32) NOT NULL," .
    "admin TINYINT(1) NOT NULL," .
    "role ENUM('Guest', 'Dev', 'Owner') NOT NULL," .
    "timestamp TIMESTAMP NOT NULL);";

$createResult = mysqli_query($dbc, $createTable);
echo "Database create: $results";
?>
```



Talking to mySQL from PHP

- Now we can insert some data in a similar fashion

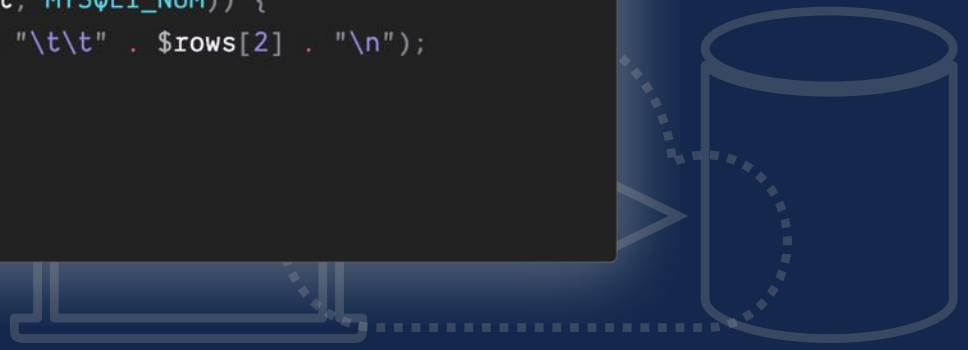
```php
<?php
$insertStatement = "INSERT INTO users" .
    "(username,email,password,first_name,last_name,admin,role)" .
    "VALUES ('tpowell','tpowell@hotmail.com','superSecret'," .
    "'Thomas','Powell',1,'Owner');";

$insertResult = mysqli_query($dbc, $insertStatement);
echo "Database insert: $results";
?>
```



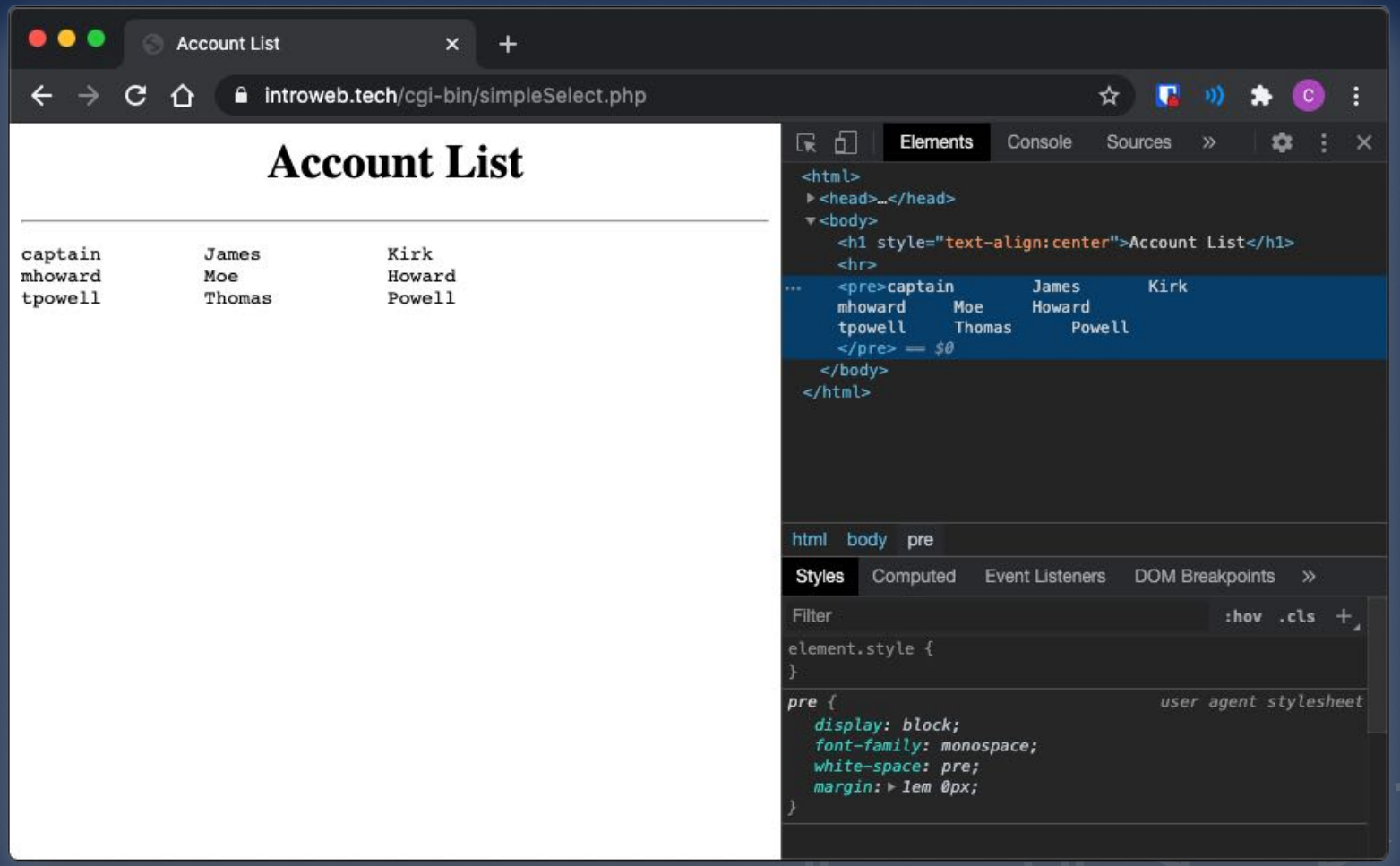
Running a Select Query from PHP

- Once you have done this, you can run a query or other SQL statement

```php
<?php
$query = "SELECT username, first_name, last_name FROM users ORDER BY username;";
$result = mysqli_query($dbc, $query);

// Loop through the resulting rows,
// but make sure to account for empty sets
$numAccts = mysqli_num_rows($result);
print "<pre>";
if ($numAccts == 0) {
    print "No accounts";
} else {
    while ($rows = mysqli_fetch_array($result, MYSQLI_NUM)) {
        print($rows[0] . "\t\t" . $rows[1] . "\t\t" . $rows[2] . "\n");
    }
}
print "</pre>";
?>
```



Running a Select Query from PHP

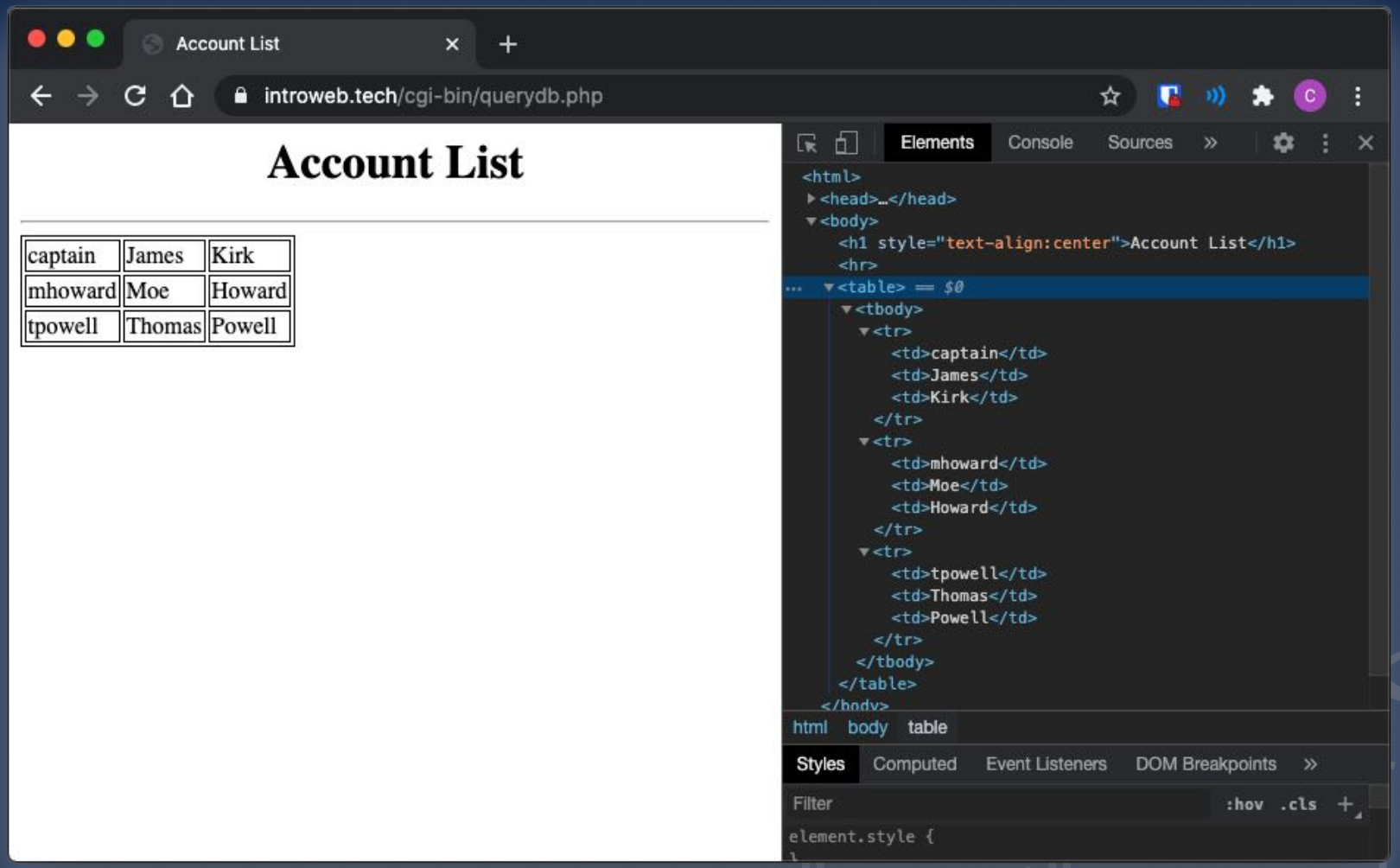


Running a Select Query from PHP

- Using tables for query results is often appropriate

```php
<?php
$query = "SELECT username, first_name, last_name FROM users ORDER BY username;";
$result = mysqli_query($dbc, $query);

// Loop through the resulting rows,
// but make sure to account for empty sets
$numAccts = mysqli_num_rows($result);
print "<table>";
if ($numAccts == 0) {
    print "<tr><td>No accounts</td></tr>";
} else {
    while ($rows = mysqli_fetch_array($result, MYSQLI_NUM)) {
        print("<tr><td>" . $rows[0] . "</td><td>" . $rows[1] . "</td><td>" . $rows[2] . "</td></tr>");
    }
}
print "</table>";
?>
```

Running a Select Query from PHP



Delete Items

- An easy pattern to follow is to add delete and edit links next to item with the unique-id for the object embedded. Then we can tie the delete link to call a page that runs a SQL statement to delete

```php
<?php
$query = "SELECT username, first_name, last_name, id FROM users ORDER BY username;";
$result = mysqli_query($dbc, $query);

// Loop through the resulting rows,
// but make sure to account for empty sets
$numAccts = mysqli_num_rows($result);
print "<table>";
if ($numAccts == 0) {
    print "<tr><td>No accounts</td></tr>";
} else {
    while ($rows = mysqli_fetch_array($result, MYSQLI_NUM)) {
        print("<tr><td>" . $rows[0] . "</td><td>" . $rows[1] . "</td><td>" . $rows[2] . "</td>");
        // Should a Delete function be a GET request? Ease over Correctness?
        print("<td><a href='/cgi-bin/deleteitem.php?id=" . $rows[3] . "'>Delete</a></td></tr>");
    }
}
print "</table>";
?>
```

Delete Items

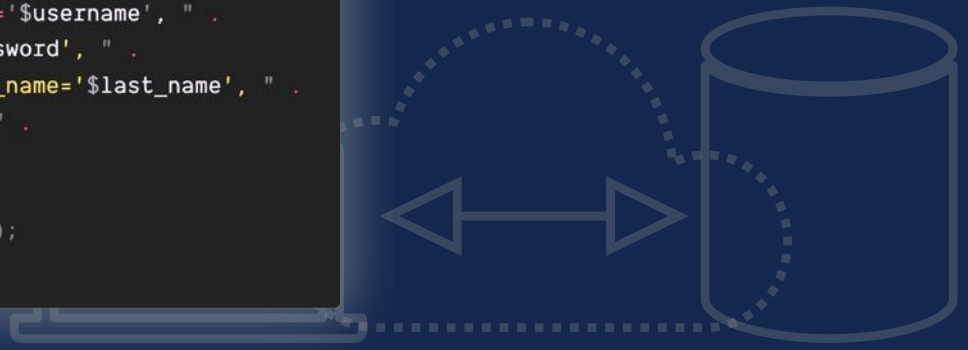
- Then, over in deleteitem.php you'd connect to the db, and then have something like this

```php
<?php
$query = "DELETE FROM users WHERE id='" . $_GET['id'] . "';";
$result = mysqli_query($dbc, $query);

// Makes the user do a u-turn of sorts once the delete has been executed
function goback()
{
    header("Location: {$_SERVER['HTTP_REFERER']}");
    exit;
}

goback();
?>
```

Update Item

- Similar to the delete pattern we add a link to trigger the edit. This link sends us to a form pre-populated with the DB entry we will update
- We then take this updated text and send to a PHP file that performs the update and bounces us back to the main page

```php
<?php
$id = $_GET['id'];
$username = $_GET['username'];
$email = $_GET['email'];
$password = $_GET['password'];
$first_name = $_GET['first_name'];
$last_name = $_GET['last_name'];
$admin = $_GET['admin'];
$role = $_GET['role'];

$query = "UPDATE users SET username='$username', " .
    "email='$email', password='$password', " .
    "first_name='$first_name', last_name='$last_name', " .
    "admin='$admin', role='$role', " .
    "WHERE id='$id';";

$result = mysqli_query($dbc, $query);
?>
```



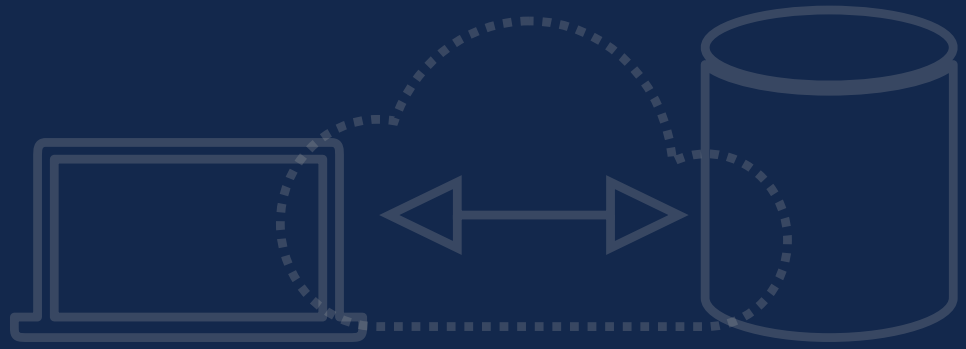
Add Item

- Similar to the update example we now create a link “Add item” which then points to a blank form and runs an insert command. Given the similarity of add and update you could imagine combining them

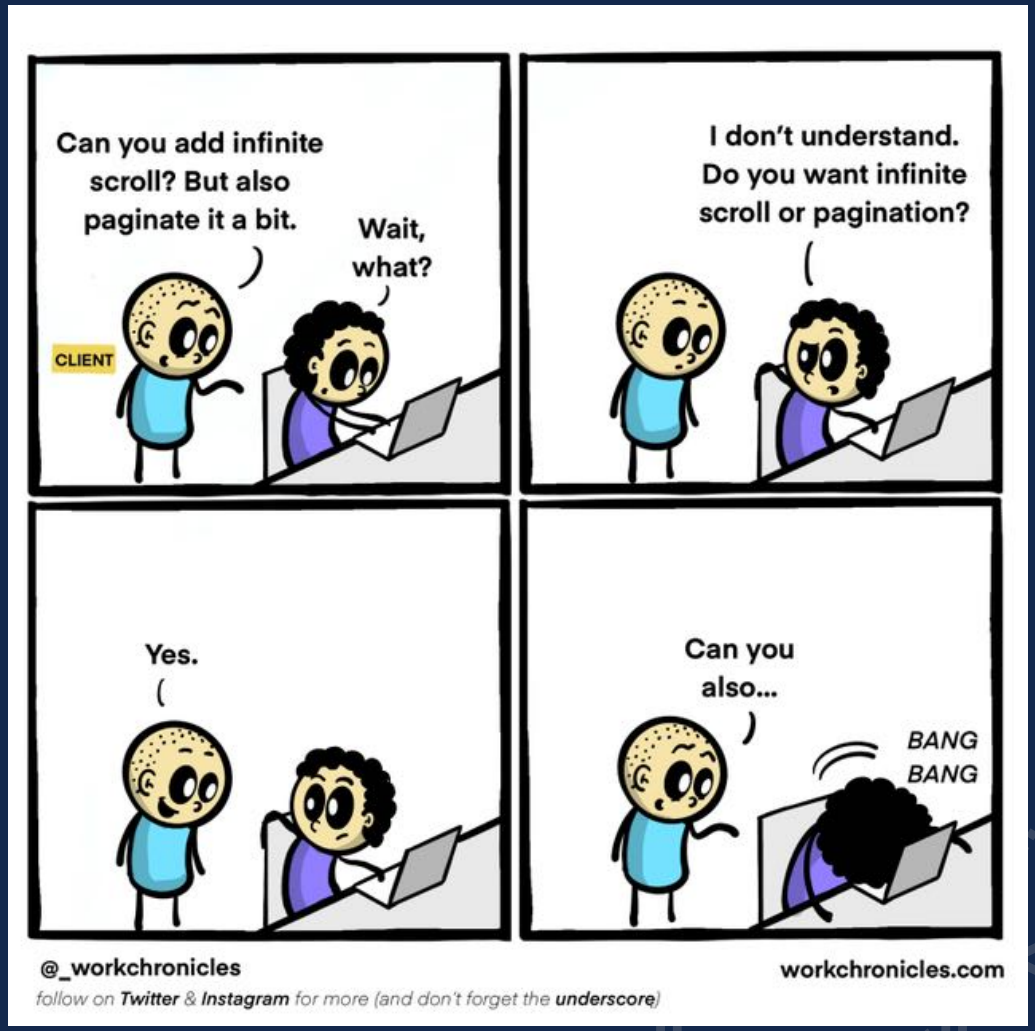
```php
<?php
$username = $_GET['username'];
$email = $_GET['email'];
$password = $_GET['password'];
$first_name = $_GET['first_name'];
$last_name = $_GET['last_name'];
$admin = $_GET['admin'];
$role = $_GET['role'];

$query = "INSERT INTO users (username, email, password, first_name," .
    "last_name, admin, role) VALUES ('$username', '$email', '$password'," .
    "'$first_name', '$last_name', '$admin', '$role')";

$result = mysqli_query($dbc, $query);
?>
```

Adding More

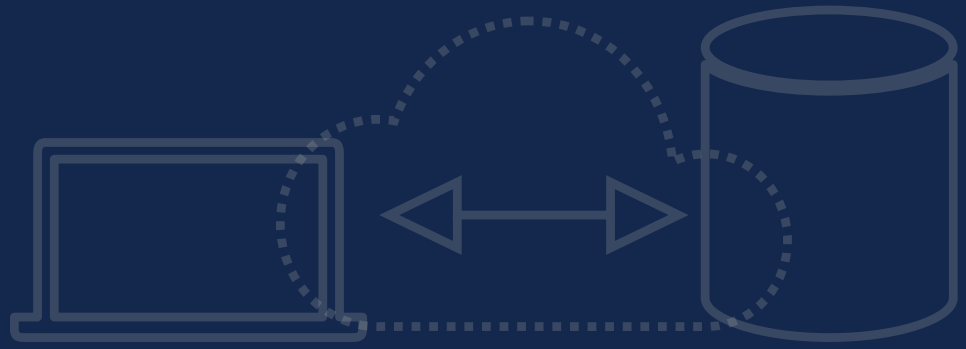
- One common feature is to add sorting by columns
- Another possibility is to add paging of results
  - Show by 5, 10, etc.
- Visually you might do zebra stripping for readability – alternate row colors
- You also might want to integrate validation and the various JavaScript usability improvements we have been discussing along the way



Features to Add - Pagination, Filtering, Scrolling, Sorting, …

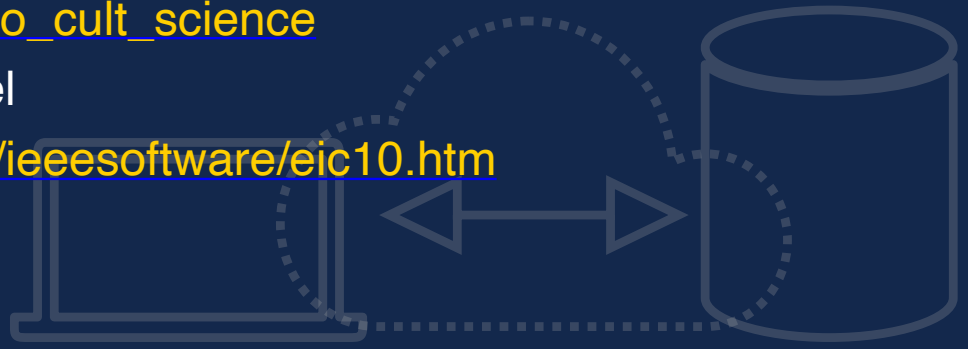


Web Site Template Systems

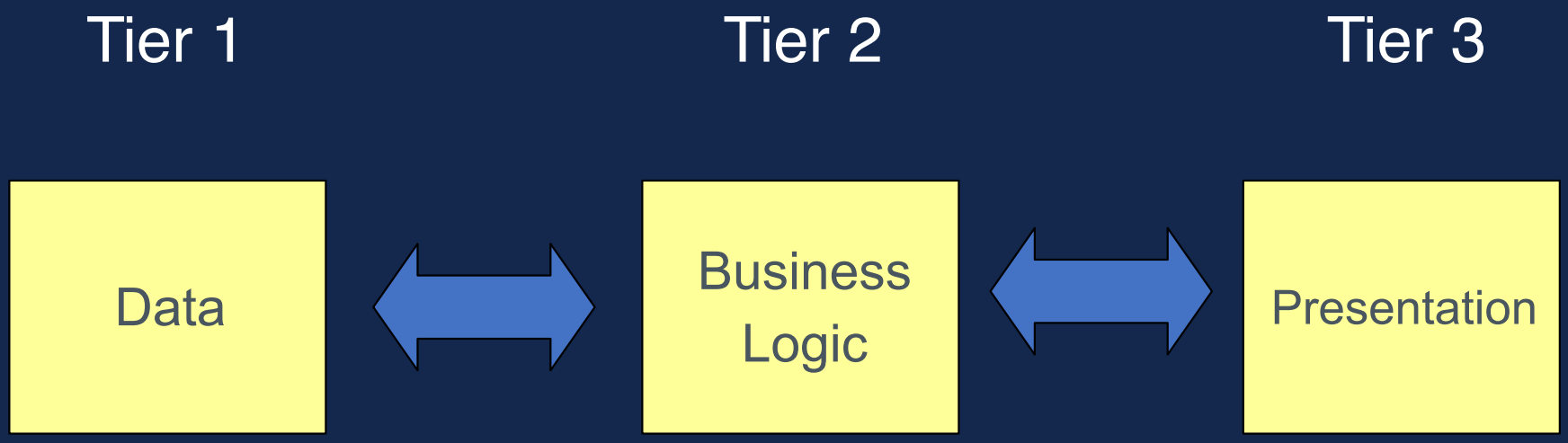


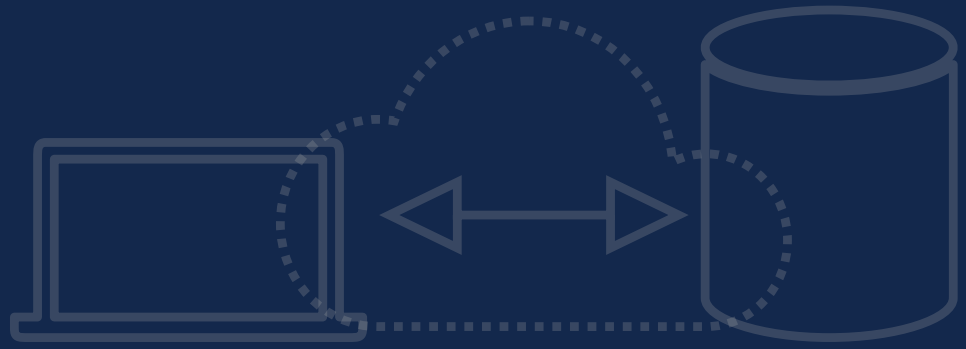
Goals of a Three Tier Web Architecture

- There can be numerous benefits to a well designed n-tier Web site/application
  - Each tier could be run on a separate machine(s) leading to better performance?
  - Better structure leading to better maintenance
  - Better division of labor due to structure also leading to improved maintenance
- Of course like any design pattern it is very possible to cause yourself great trouble if it is poorly applied or shudder simply not as necessary as some would have you believe
  - A “cargo cult” problem often seems to abound in programming at times
    - Feynman speech
      - http://en.wikipedia.org/wiki/Cargo_cult_science
    - Read application to SE by McConnel
      - http://www.stevemcconnell.com/ieeesoftware/eic10.htm

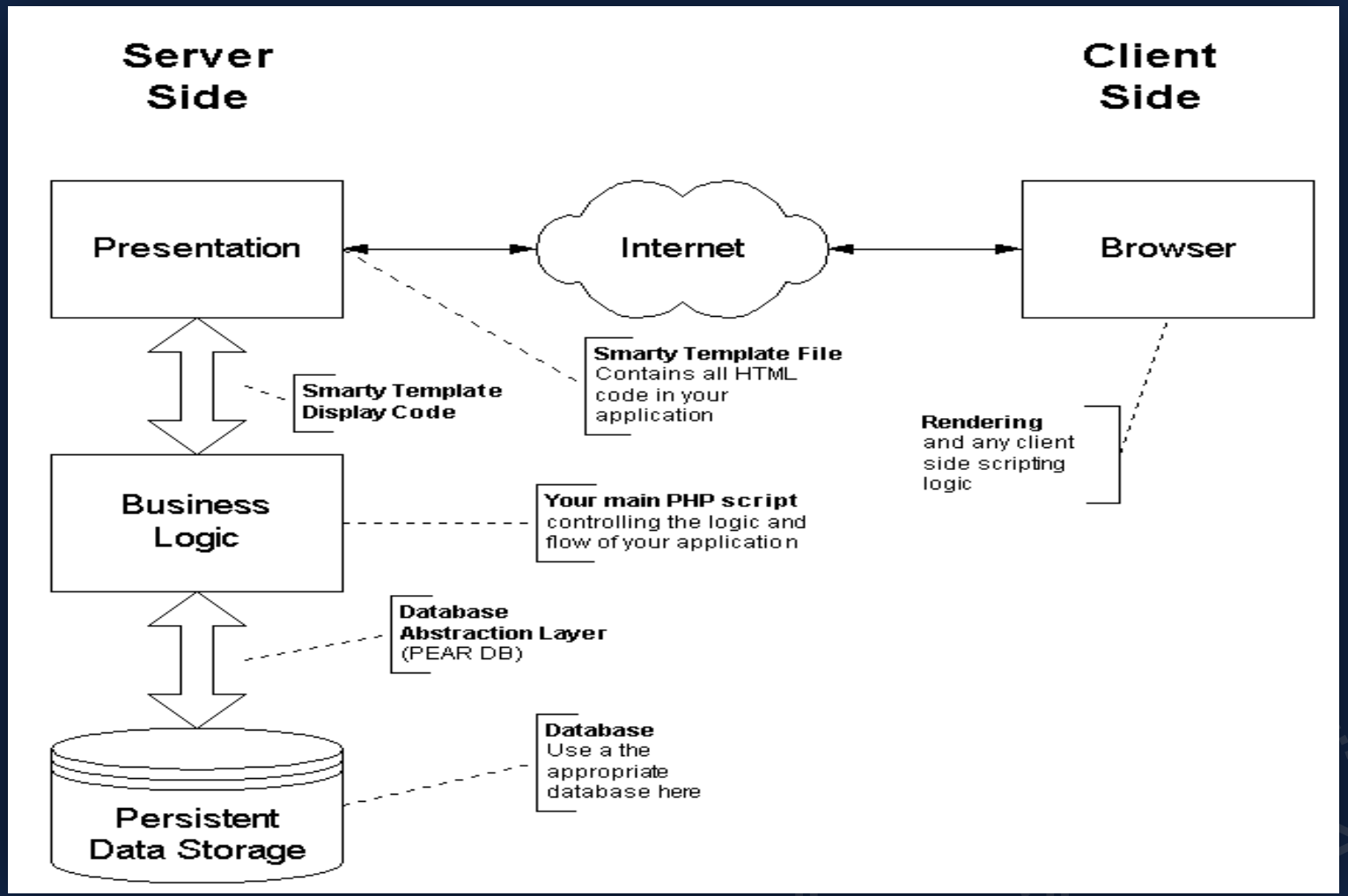


Typical 3-Tier Separation



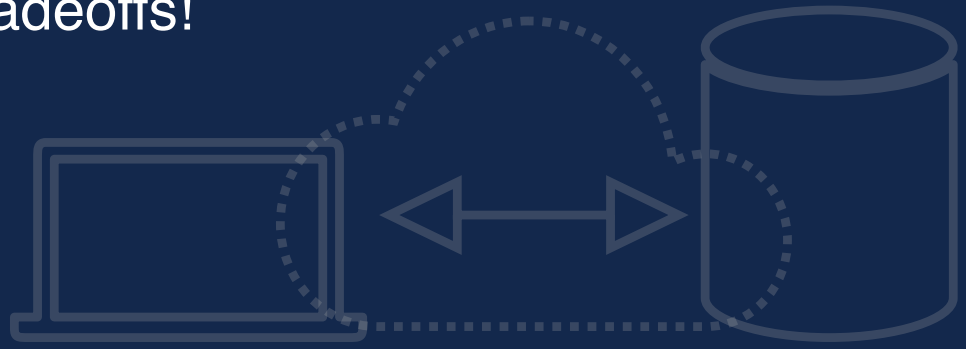


Typical 3-Tier Separation in PHP



Database Abstraction Layer

- In the case of PHP there are lots of choices to abstract for example
  - http://pear.php.net/package/MDB2
- The advantage is to abstract away certain details of an underlying database
  - Portability
    - Allow for easy switching of DB
    - Provide for easier code distribution
  - Provide more of an OO approach to the DB connections in PHP
  - At what cost? Remember your tradeoffs!



Template Example: Smarty

- Numerous template systems out there for PHP with the most notable being Smarty http://www.smarty.net/

- Reasons promoted for using template langs like Smarty
  - Big one: Separate code from template
  - Caching of templates, white space striping, other delivery prepping facilities
  - Avoid using PHP for presentation use a less powerful language and thus improve security (really?)

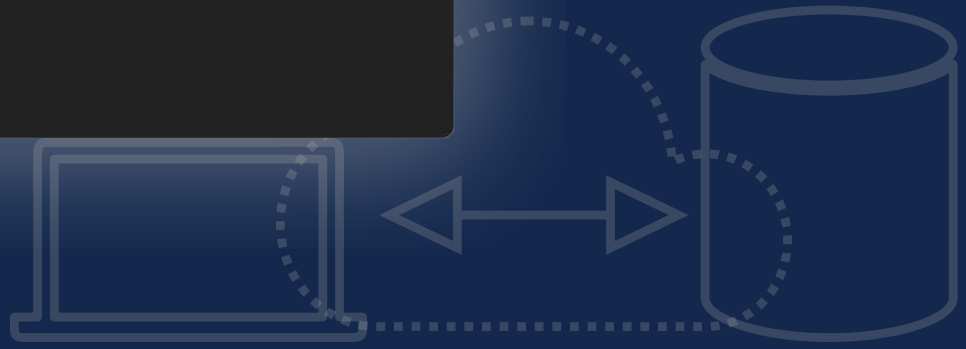
- Note: There are also libraries for specialized HTML output that you might want to take a look

Smarty Hello World

```php
<?php
// My path to Smarty
require ('./smarty/libs/Smart.class.php');

$smarty = new Smarty;

// Set a path to your templates
$smarty->template_dir = './templates/';

$smarty->assign('Username', 'Mr. Smarty');
$smart->display('index.tpl');
?>
```

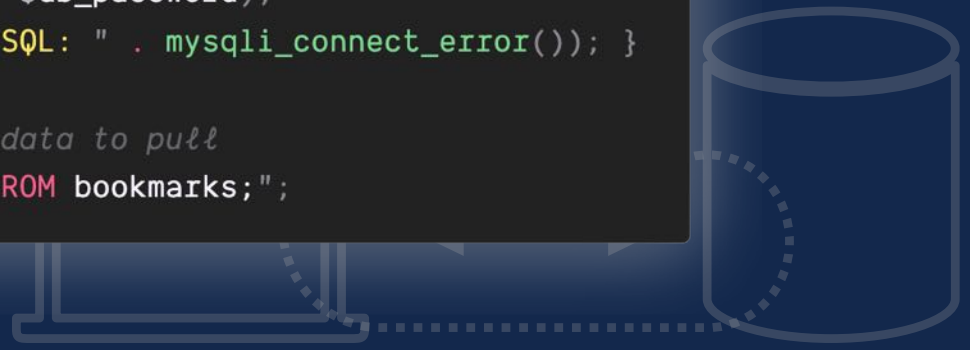
Smarty Hello World Template

```
1  <html>
2    <head>
3      <title>Hello Smarty</title>
4    </head>
5    <body>
6      Hello world {$Username}!
7    </body>
8  </html>
```



Smarty DB Example

```php
<?php
// Include the smarty file, create an object, and set the directory
require ('./smarty/libs/Smarty.class.php');
$smarty = new Smarty;
$smarty->template_dir = './templates/';

$db_user = 'root';
$db_password = 'superSecret';
$db_host = 'localhost';
$db_name = 'mydatabase';

// Connect to the mysql database
$dbc = mysqli_connect($db_host, $db_user, $db_password);
if (!$dbc) { die("Could not connect to MySQL: " . mysqli_connect_error()); }
mysqli_select_db($dbc, $db_name);
// The SQL query that will specify which data to pull
$query = "SELECT name, url, description FROM bookmarks;";
```

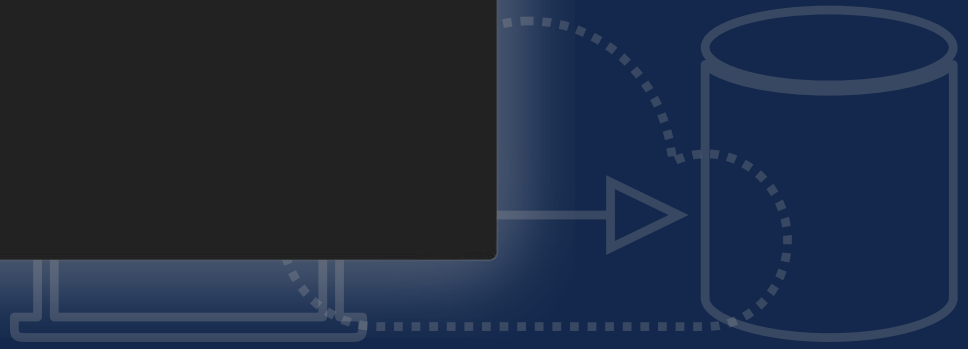


Smarty DB Example Contd.

```php
$result = mysqli_query($dbc, $query);
$bookmarks = array();
$i=0;
// Loop through all of the query rows
while($row = mysqli_fetch_array($results)) {
    // Create a bookmark, which will be an array of keys & values.
    // The values are pulled from the database.
    $aBookmark = array('name' => $row['name'],'url'
    => $row['url'],'description' => $row['description']);
    // Add that bookmark to the end of an array
    $bookmarks[$i++] = $aBoomark;
}
// Populate the template and display it
$smarty->assign('results', $bookmarks);
$smarty->display('displaybookmarks.tpl');
?>
```

Displaybookmarks.tpl

```
<!DOCTYPE html>
<html>
  <head>
    <title>List Bookmarks Smarty Style</title>
    <link rel="stylesheet" href="/styles/main.css" />
  </head>
</html>
<body>
  <h1>Bookmarks Smarty Style</h1>
  <hr />
  <table>
    <tr>
      <th>Name</th>
      <th>URL</th>
      <th>Description</th>
    </tr>
```

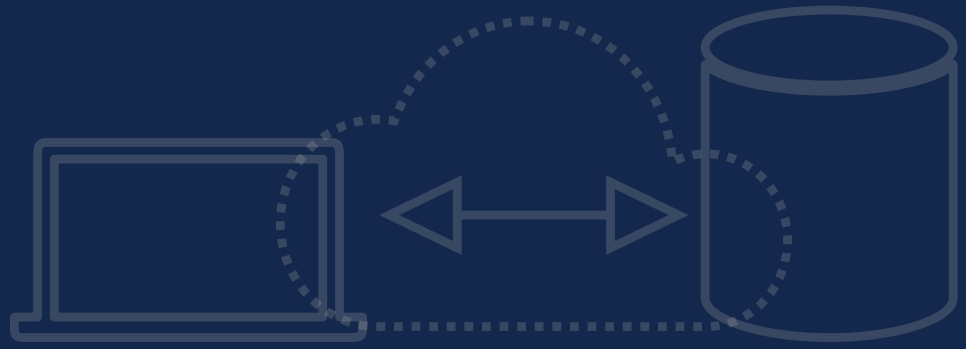


Displaybookmarks.tpl Contd.

```
{section name=nr loop=$results}
    <tr {if $smarty.section.nr.iteration is odd}  style="background-color:lightgray" {/if}>
      <td>
        {$results[nr].name}
      </td>
      <td>
        {$results[nr].url}
      </td>
      <td>
        {$results[nr].description}
      </td>
    </tr>
    {/section}
  </table>
</body>
</html>
```

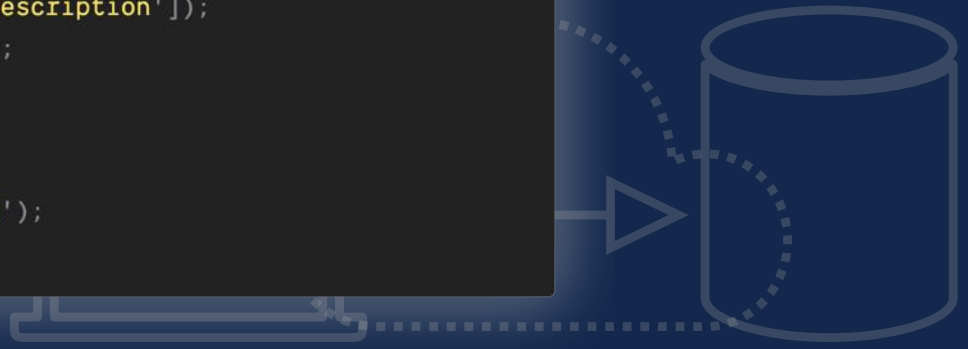
Pretty cool…but

- Some developers are skeptical of template systems like Smarty
  - Have to learn a new language
    - Is it that much different than just plain PHP?
      - `{ }` instead of `<? ?>`
  - Can't you just do it in PHP anyway?
    - Maybe you can



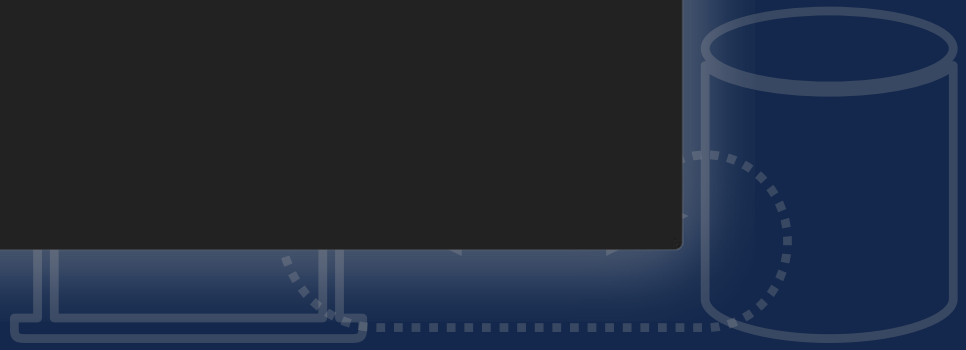
Notsosmarty.php

```php
<?php
$db_user = 'root';
$db_password = 'superSecret';
$db_host = 'localhost';
$db_name = 'test';
// Connect to the mysql database
$dbc = mysqli_connect($db_host, $db_user, $db_password);
if (!$dbc) { die("Could not connect to MySQL: " . mysqli_connect_error()); }
// Construct and send database query
$query = "SELECT name, url, description FROM bookmarks";
$result = mysqli_query($dbc, $query);
$bookmarks = array();
$numbookmarks = 0;
// Loop through database entries and format them for the template
while ($row = mysqli_fetch_array($result)) {
  $aBookmark = array('name' => $row['name'], 'url'
  => $row['url'], 'description' => $row['description']);
  $bookmarks[$numbookmarks++] = $abookmark;
}

// Now pass the results to the template
include('./templates/displaybookmarks2.tpl');
?>
```



A Template without Smarty

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>List Bookmarks Not Smarty Style</title>
    <link rel="stylesheet" href="style.css" />
  </head>
  <body>
    <h1>Bookmarks Not So Smarty Style</h1>
    <hr />
    <table>
      <tr>
        <th>Name</th>
        <th>URL</th>
        <th>Description</th>
      </tr>
```

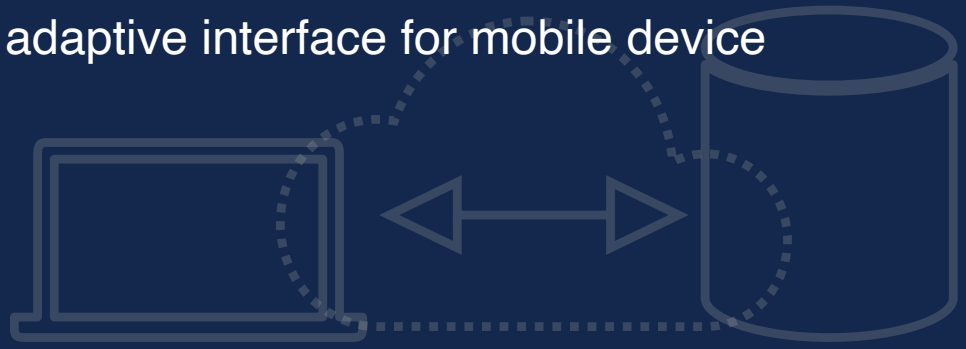


A Template without Smarty Contd.

```php
<?php for ($nr = 1; $nr < $numbookmarks; $nr++) { ?>
        <tr <?php if ($nr % 2 == 0) print('style="background-color:lightgray"'); ?> >
          <td>
            <?=$bookmarks[$nr]['name']?>
          </td>
          <td>
            <?=$bookmarks[$nr]['url']?>
          </td>
          <td>
            <?=$bookmarks[$nr]['description']?>
          </td>
      <?php } ?>
    </table>
  </body>
</html>
```

Template Conclusions

- "… the point of template engines should be to separate your business logic from your presentation logic, not separate your PHP code from your HTML code."
- Still the idea of templates is still a good idea, despite how it may be aggressively implemented
  - Easier to "re-skin" an application
  - Helps divide work from programmers and designers to a degree
- Don't get carried away with the separation of presentation and logic
  - Presentation may have logic associated with it
    - Particularly if you are trying to do an adaptive interface for mobile device



Switching to High Gear

- Since we build so many CRUD applications a number of efforts have been made to build them faster and easier
  - Examples: Rails framework for Ruby, Cake for PHP, Sails for NodeJS, Meteor, etc.
- These frameworks often contain tremendous demo deception and hide the tradeoffs made
  - Don’t interpret this as me not wanting to use them – interpret this as knowing what it is being provided and what you will have to address a scaffold is the frame not the finished product is all I want you to acknowledge

From Table to Object

- ORM
  - From the all knowing Wikipedia - "**Object-Relational Mapping** (aka **ORM**, **O/RM**, and **O/R mapping**) is a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages."
  - Lots of implementations exist
    - http://en.wikipedia.org/wiki/List_of_object-relational_mapping_software
- Example:
  - Take a Database table named "Pets" that has columns like name, type, breed, color and a primary key and an ORM system will create a relation to an object called "Pet" with similar properties
  - Question: What's the primary key? Answer: Convention for most systems is "id"

Scaffolding

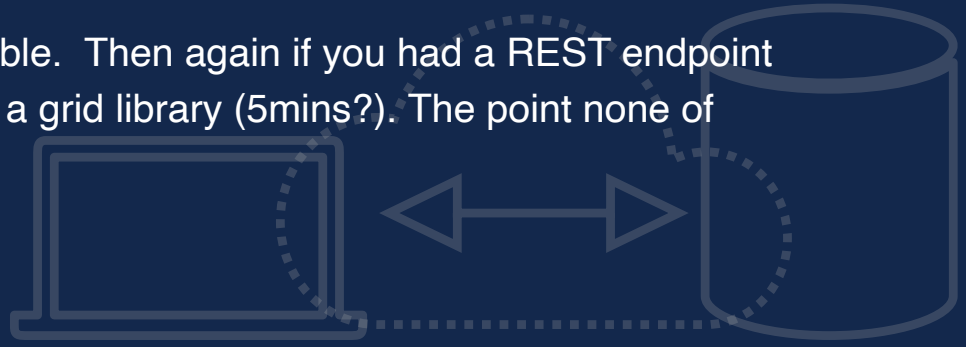
- In the real world
  - "**Scaffolding** is a temporary framework used to support people and material in the construction or repair of buildings and other large structures."

- In the Web and programming world
  - Generally term given to a quick and dirty framework built by inspecting database tables or various config files
  - Example: In Ruby you might create the scaffolds with some CLI command like

```
ruby script\generate model Pet
ruby script\generate controller Pet
```

or just `ruby script\generate scaffold Pet`

Out of this pops a full fledged CRUD app for that table. Then again if you had a REST endpoint how long would it take you to set CRUD actions on a grid library (5mins?). The point none of this is done it is a scaffold!



Scaffolding Part 2 - Quick CRUD

- With some quick CRUD scaffold in hand you likely have a list, edit, add, and delete page that allows you to make changes to the DB
- You also will find that a change to the DB can be quickly recognized in the app
  - Sometimes automatically sometimes via a trigger to re-scaffold

- Now you have to go and improve the views to make it look pretty and add any other customized business logic desired

- Scaffold is a good thing it allows you to prototype things but even when functional assume that security or performance may not be baked in

Summary

- MySQL is a very popular SQL database associated with PHP but generally widely used regardless of language
- SQL databases like MySQL use a common syntax called SQL which as commands for creating tables, adding data, managing data and selecting data
- It is important to plan your tables properly
- Consider doing things manually to understand the ideas of a database before you write code
- Code based solutions for database follow an MVC pattern.
  - The view layer of MVC tends to be the most important since it is part of the UI and changes the most. It also can be messy
- Leverage systems where appropriate to scaffold your CRUD apps but do not confuse such scaffolds as finished products there are usually many details not addressed.