COMP721 Web Development

AUT UNIVERSITY

Week 4: CSS, Regular Expressions, SQL and NoSQL DB

CSS

Regular expressions

Database basics

MySQL DB

NoSQL DB: MongoDB

Week 3 Review

- In Week 3 we have discussed:
- Arrarys: multi-dimensional arrays, associative arrays
- Binary search and quicksort
- Superglobals
- String functions: test, construct, compare, parse, convert

$_GET

Keys used to retrieve user input in $_GET

```
<form action="process.php" method="get">
   <label>User Name:  <input type="text" name="name"> </label>
   <label><br>Password: <input type="text" name="pwd"></label>
   <input type="submit" value="Submit">
</form>
```

CSS: Cascading Style Sheets

The style language of the Web

Used to present the content of Web Pages

Bootstrap

The most popular CSS framework

Cascading Style Sheets

- A single piece of CSS formatting information, such as text alignment, is referred to as a **style**
- The term **cascading** refers to the ability for Web pages to use CSS information from a series of sources – but which source to choose depends on the cascading order
- CSS is close to SQL, which is a query language. It use query to select a set of HTML elements, and define the styling rules on this selected set

Cascading Style Sheets (continued)

- **Style sheets** contain style information as a collection of style "**rules**"
- **Rules** start with a **selector** and then contain style **properties** and **values**.

```
selector { property1: value1; property2: value2; …}
```

CSS Rule

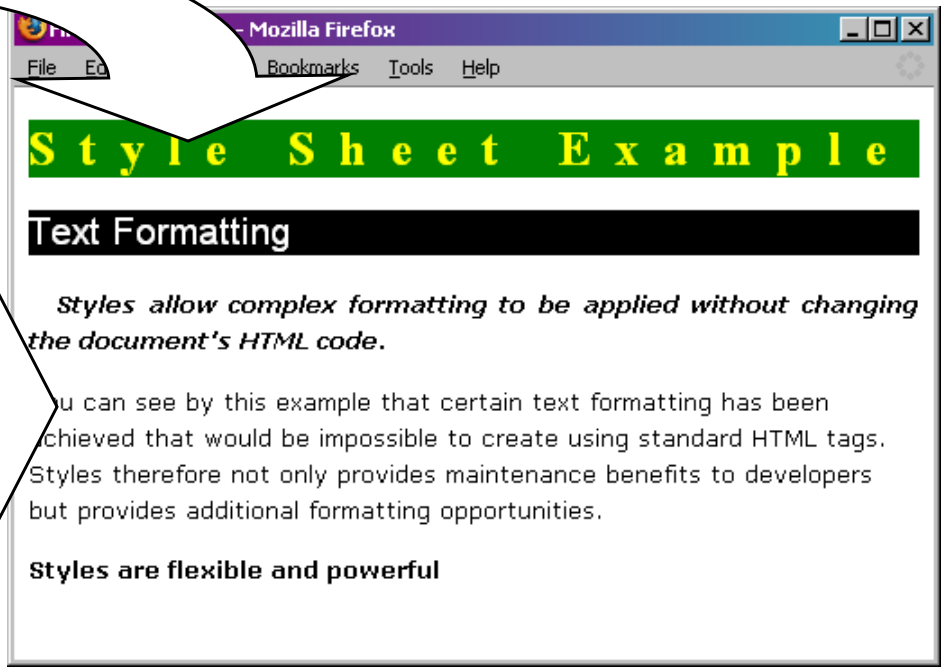
A **selector** identifies the **markup elements** that the style property values will be applied to.

Cascading Style Sheets (continued)

Remember the simple structure of HTML documents!

```
<!DOCTYPE ...>
<html>
  <head>
    <title>...</title>
    <link rel="stylesheet"
    type="text/css"
    href= "mystyle.css" />
  </head>
  <body>
    <!-- body content here -->
  </body>
</html>
```

Same external stylesheet can be linked and be re-used for many webpages



Cascading Style Sheets (continued)

- Here is a quick start list of some example style rules

```
h1, h2    { font-family: sans-serif; }
th        { color: #3366CC;}
div p     { border: 1px solid #FF0000; }
    (Selects descendant)
a:hover   { font-weight: bold;}
li        { font-size: 12px; }
a         { text-decoration: underline overline; }
h3        { border-bottom: 2px dashed green; }
#danger   { text-align: justify; color: red }
  (selects element with id="danger")
p.indent { text-indent: 20px; }
.upper    { text-transform: uppercase; }
  (Selects all elements with class="upper")
img       { float: right; }
ol        { list-style-type: upper-roman; }
```

Cascading Style Sheets (continued)

AUT UNIVERSITY

Inline Styles

- Allow you to add style information to a single element in a document

```
<p>This paragraph does not use CSS. </p>

<p style="font-family: Verdana; color: blue; text-align: center">
Paragraph formatted with inline styles. </p>
```

Note that based on cascading order, inline style will override css in other sources, as it is more specific!

Cascading Style Sheets (continued)

Internal Style Sheets

- Create styles that apply to the entire document

```
<html>
 <head>
 <title> Toner Cartridge Sales </title>
 <style type="text/css">
  body { color: blue; font-size: .8em; font-family: Arial, sans-serif; font-weight:  normal}
  h1 { color: navy; font-size: 2em; font-family: serif}
  h2 { color: red; font-size: 1.5em; font-family: Arial, sans-serif}
</style>
 </head>
 <body>
 <h1> Toner Cartridge Sales </h1>
 <hr />
 <h2> Lexmark Toner Cartridges </h2>
 ……
 </body>
 </html>
```

Cascading Style Sheets (continued)

External Style Sheets

- A separate text document containing style declarations that are used by multiple documents on a Web site
- Copy and paste the style declarations within the <style> element into a new file, say my_styles.css. Be certain not to copy the <style> tags.
- Replace the <style> element and the style declarations it contains with the following <link> element

  <link rel="stylesheet" href="my_styles.css" type="text/css" />

Best to use External Stylesheets.
Avoid inline styles, avoid internal stylesheets

Cascading Order

- How CSS rules cascade
  - Highest priority:
    1. Inline style (inside an HTML element)
    2. External and internal style sheets (in the head section)
    3. Browser default
  - Last rule
    - If two selectors are identical, the latter of the two will take precedence
  - Specific

    H1{} is more specific than *{}
  - !important;
    - Indicates that previous assign property is more important then other rule assigned to the same element

```
body { color: blue !important; font-size: .8em}
```

CSS: W3C References

- World Web Web Consortium (W3C):

  http://www.w3.org

  Home Page

  http://www.w3.org/Style/CSS/

  Cascading Style Sheets (CSS) Home Page

  http://www.w3.org/Style/CSS/learning

  W3C Learning CSS Page

  http://jigsaw.w3.org/css-validator/

  W3C CSS Validator

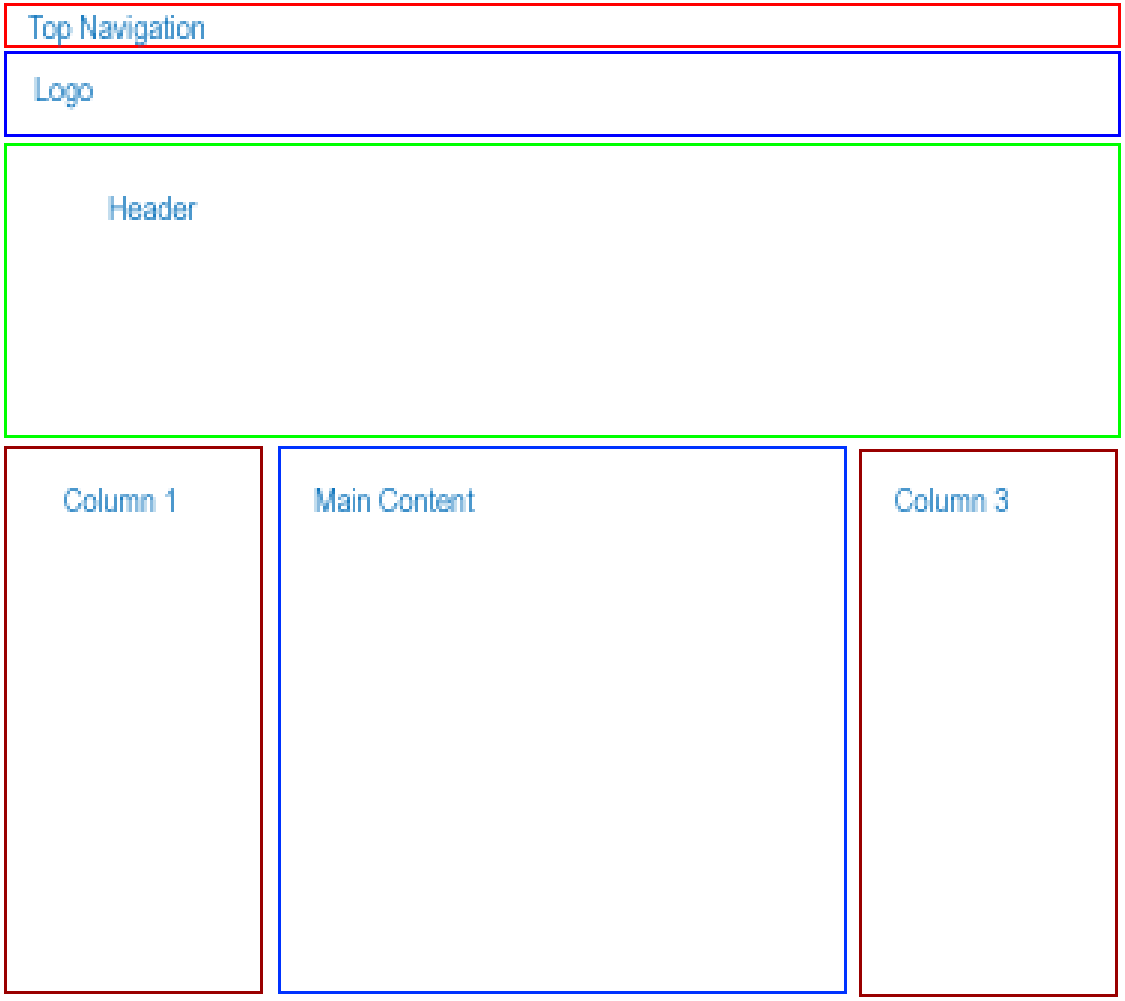
Useful CSS tools

- CSS3 pie: several of the most useful CSS3 decoration features. http://css3pie.com/
- CSS3 Click Chart: demo and code; http://www.impressivewebs.com/css3-click-chart/
- CCS3 button maker: https://css-tricks.com/examples/ButtonMaker/#
- CSS3 generator: http://css3generator.com/

Bootstrap

- A freely available design framework for websites and web applications
- Based upon HTML5, CSS and JavaScript
- Tool support
  - Such as LayoutIt!: a visual drag and drop interface builder (http://www.layoutit.com/)
- Fluid GUI: adaptable to any screen size: desktop, tablets, smart phones

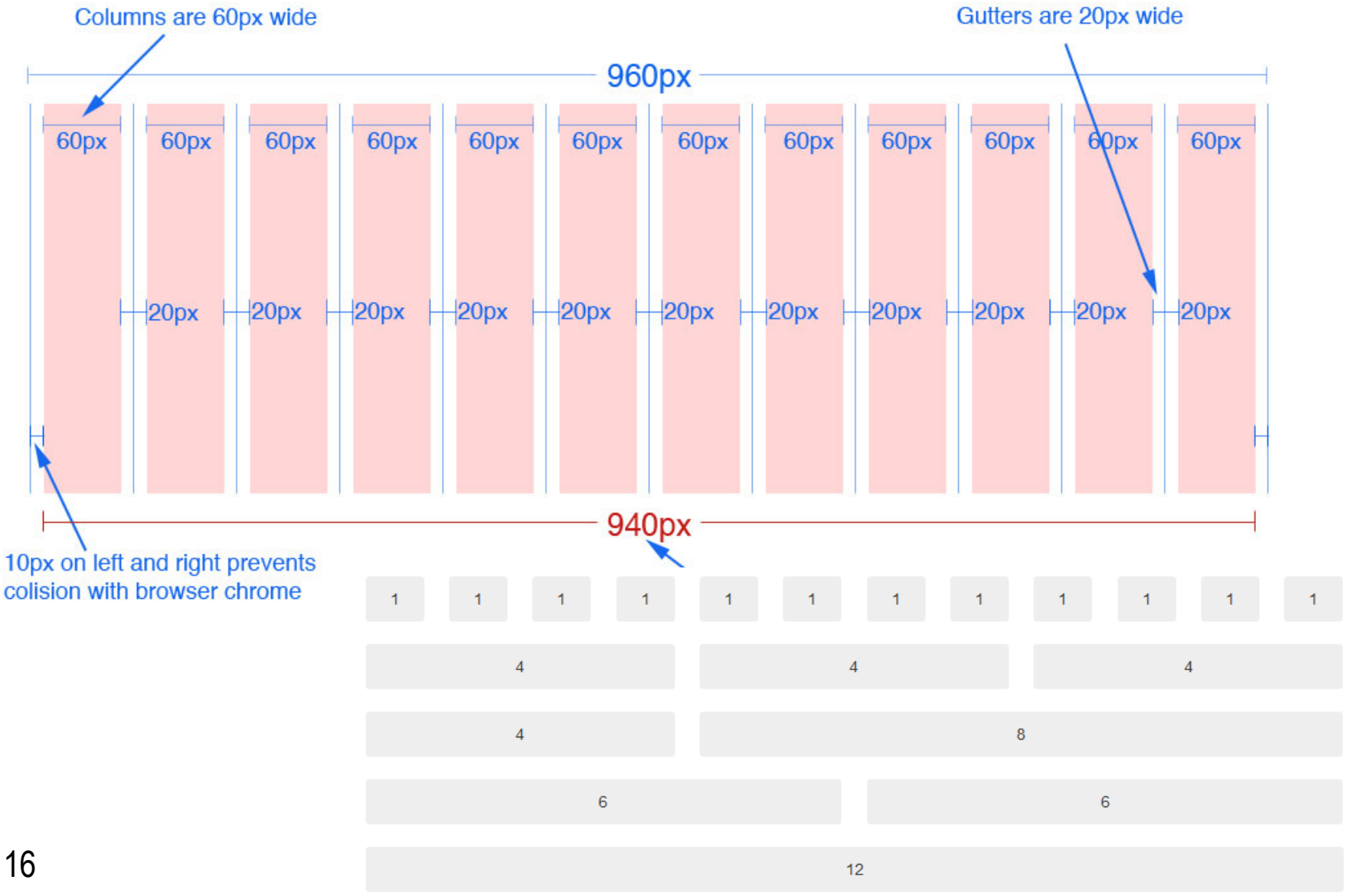
Layout: the first thing of web page/application design



Let's check AUT's website…

Bootstrap grid system for fluid layout

- 12-column version



Rich components

- Buttons

- Tabs

- Breadcrumb

- Pagination

- Alerts

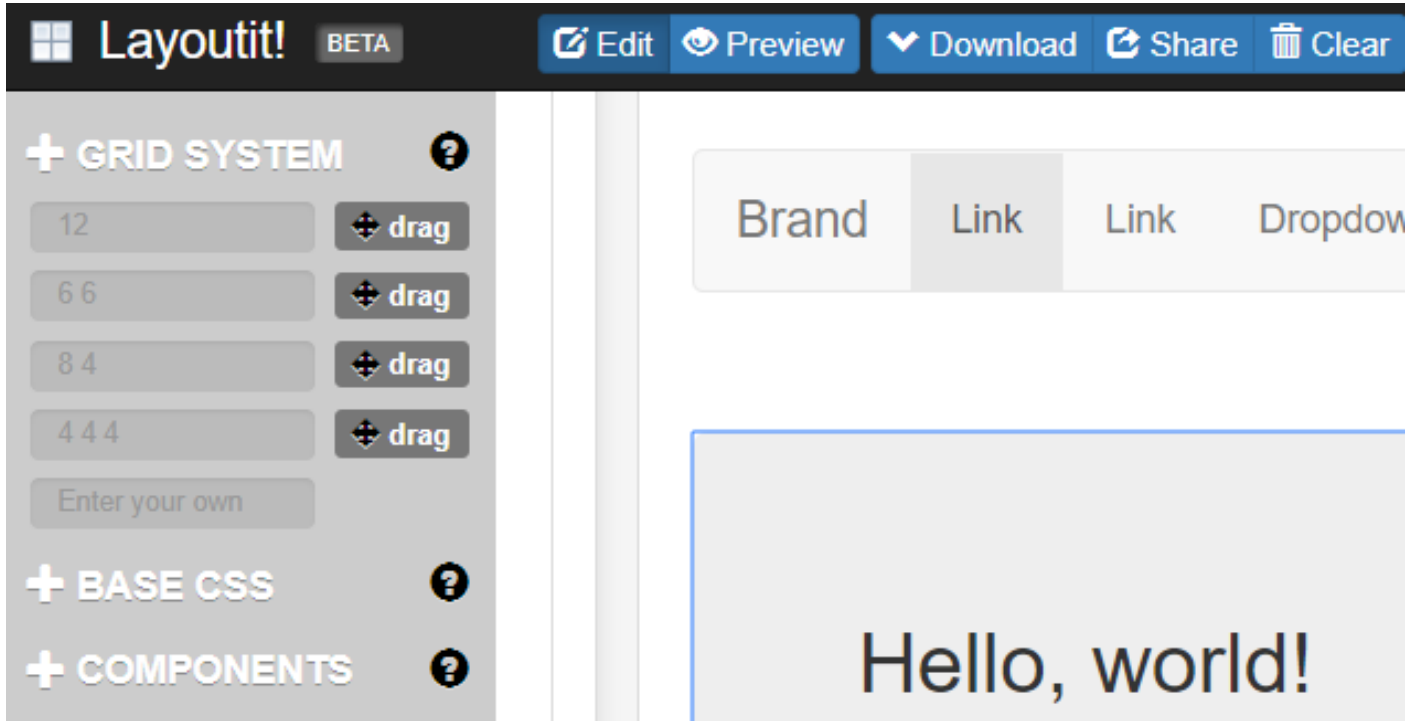
- Progress bar

Javascript/jQuery components

- [x] Transitions (required for any animation)
- [x] Modals
- [x] Dropdowns
- [x] Scrollspy
- [x] Togglable tabs
- [x] Tooltips
- [x] Popovers (requires Tooltips)
- [x] Affix
- [x] Alert messages
- [x] Buttons
- [x] Collapse
- [x] Carousel
- [x] Typeahead

Layoutit! and the generated code

- http://www.layoutit.com



CSS and trigonometry

- 19 March 2019: W3C CSS working group announced to include trigonometry functions in CSS
- CSS 3D animation demo: https://keithclark.co.uk/labs/css-fps/nojs/

Sine - sin()
Cosine - cos()
Tangent - tan()
Arccosine - acos()
Arcsine - asin()
Arctangent - atan()
Arctangent (of two numbers x and y) - atan2()
Square root - sqrt()
Square root of the sum of squares of its arguments - hypot()
Power of - pow()

REGULAR EXPRESSIONS

Using Regular Expressions

Web Developers should understand the concepts and value of using Regular Expressions

- Regular Expressions are a useful way to concisely define the syntax and 'pattern' of textural data.
- Simple functions can be used to test or 'match' data against the 'pattern'.
- Regular Expressions can be used in both client-side (HTML, CSS, JavaScript) and server-side scripts, so the same 'pattern' can be consistently applied to verify data formats.

And in particular be able to:

- Use Regular Expressions to check values entered in HTML forms.
- Example: https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_pattern

What are Regular Expressions?

- are strings that describe the ‘pattern’ or ‘rules’ for strings
- are strings that follow a set of syntax rules
- can be used as a concise and consistent way to test for matching patterns
- ***are great for checking form values!***

Regular Expressions in PHP

- PHP uses Perl Compatible Regular Expressions (PCRE) and has a range of pre-defined PCRE functions

  http://www.php.net/manual/en/ref.pcre.php

- Common functions

  `preg_match(),preg_replace(),preg_split()`

- Initialise a Regular Expression pattern, and test string

```
 $pattern = "/(chapter \d+(\.\d)*)/i";
$str = "For more information, see Chapter 3.4.5.1";

if (preg_match($pattern, $str)) {
 echo "A match was found.";
}
else{
 echo "A match was not found.";
}
```

Test/debug in PHP SandBox…

Regular Expressions in PHP

- A simple regular expression can be the equivalent of many lines of code.
- Simply define the ‘pattern’ of the Regular Expression, e.g. ‘pattern’ for a phone number such as

  (03) 9214-8000

  you could use

  $pattern = "/^\(\d\d\) \d\d\d\d-\d\d\d\d$/" ;
- Then ‘match’ the input string against the ‘pattern’

  preg_match($pattern, $inputString) // true if OK

Regular Expressions in PHP - Example

- Simple check for a phone number using **`preg_match()`**

```
function checkPhoneNumber($phoneNo) {
 $phoneRE = "/^\(\d\d\) \d\d\d\d-\d\d\d\d$/";
 if (preg_match($phoneRE, $phoneNo)) {
   return true;
 } else {
   return false;
 }
}
```

```
<form action="..." >
<p><label ...>Enter phone number (e.g.(03) 3456-7890):
  </label>
  <input type="text" name="phone" /></p>
<p><input type="submit" value="Send" /></p>
</form>
```

http://php.net/manual/en/reference.pcre.pattern.syntax.php

Regular Expressions - Basic Examples

/WebDev/	matches "Isn't WebDev great?"
/^ WebDev/	matches " WebDev rules!", not "What is WebDev?"
/ WebDev$/	matches "I love WebDev", not " WebDev is great!"
/^ WebDev$/	matches " WebDev", and nothing else (exact match)
/bana?na/	matches "banana" and "banna", but not "banaana".
/bana+na/	matches "banana" and "banaana", but not "banna".
/bana*na/	matches "banna", "banana", and "banaaana", but not "bnana"
/^[a-zA-z]+$/	matches any string of one or more letters and nothing else.

Regular Expressions - Basic Syntax

`/pattern/modifiers`

http://php.net/manual/en/reference.pcre.pattern.modifiers.php

Pattern Basics

^	Start of string
$	End of string
.	Match any single character
(a|b)	a or b
(…)	Group section
[abc]	match any character in the set
[^abc]	not match in the set
[a-z]	match the range
\d	match a single digit from 0 to 9 *shortcut for [0-9]*
\s	match space or tab

Pattern Quantifiers

a?	0 or 1 of a
a*	0 or more of a
a+	one or more instance of a
a{3}	exactly 3 a's = aaa
a{3,}	3 or more a's
a{3,6}	between 3 to 6 a's
!(pattern)	"not" pattern

[\ ^ $. | ? * + ( ) are the 11 meta-characters, or special characters, used in the syntax.
If you want to include these, you need to escape them with \. eg. \(

\w: shortcut for [A-Za-z0-9_]

Some Sample Patterns

/[A-Za-z0-9-]+/ = Letters, numbers and hyphens

/\d{1,2}\/\d{1,2}\/\d{4}/ = date as 19/9/2006

/#?([A-Fa-f0-9]){3}(([A-Fa-f0-9]){3})?/ = valid hex colour code

/^.+@.+\..{2,3}$/ = email address

/\w+@[a-zA-Z_]+\.[a-zA-Z]{2,6}/ = email address

/\<(/[^\>]+)\>/ = HTML tags

Regular Expressions - Basic Syntax

```
/pattern/modifiers
```

- **Pattern Modifiers**

/g	global matching (find all the matches)
/i	case insensitive
/s	single line mode
/m	multiple-line mode
/x	allow comments and white space in pattern
/e	evaluate replacement
/U	ungreedy replacement

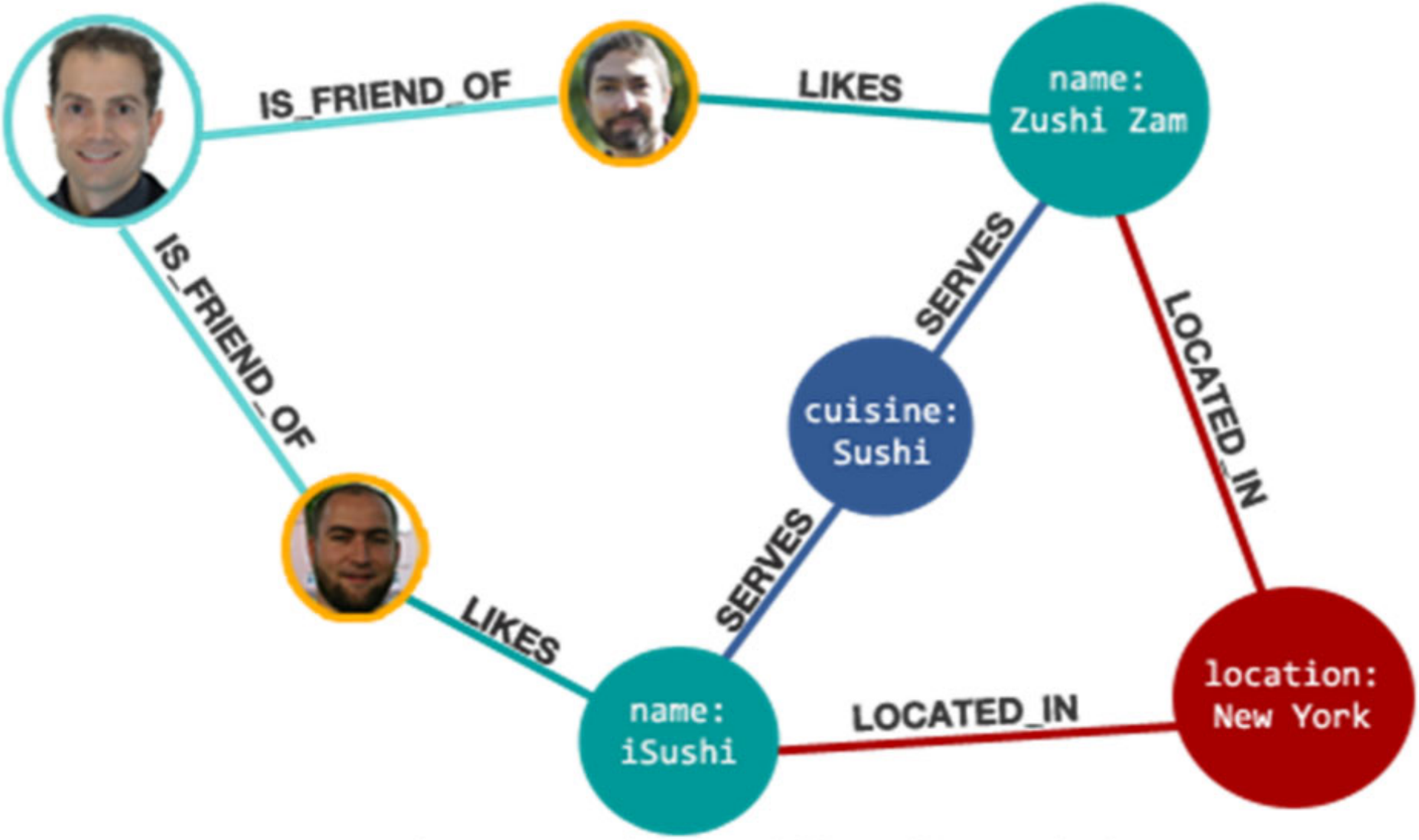
There are many useful online syntax references about Regular Expressions, such as:
http://www.regular-expressions.info/

Database basics

Introduction to Databases

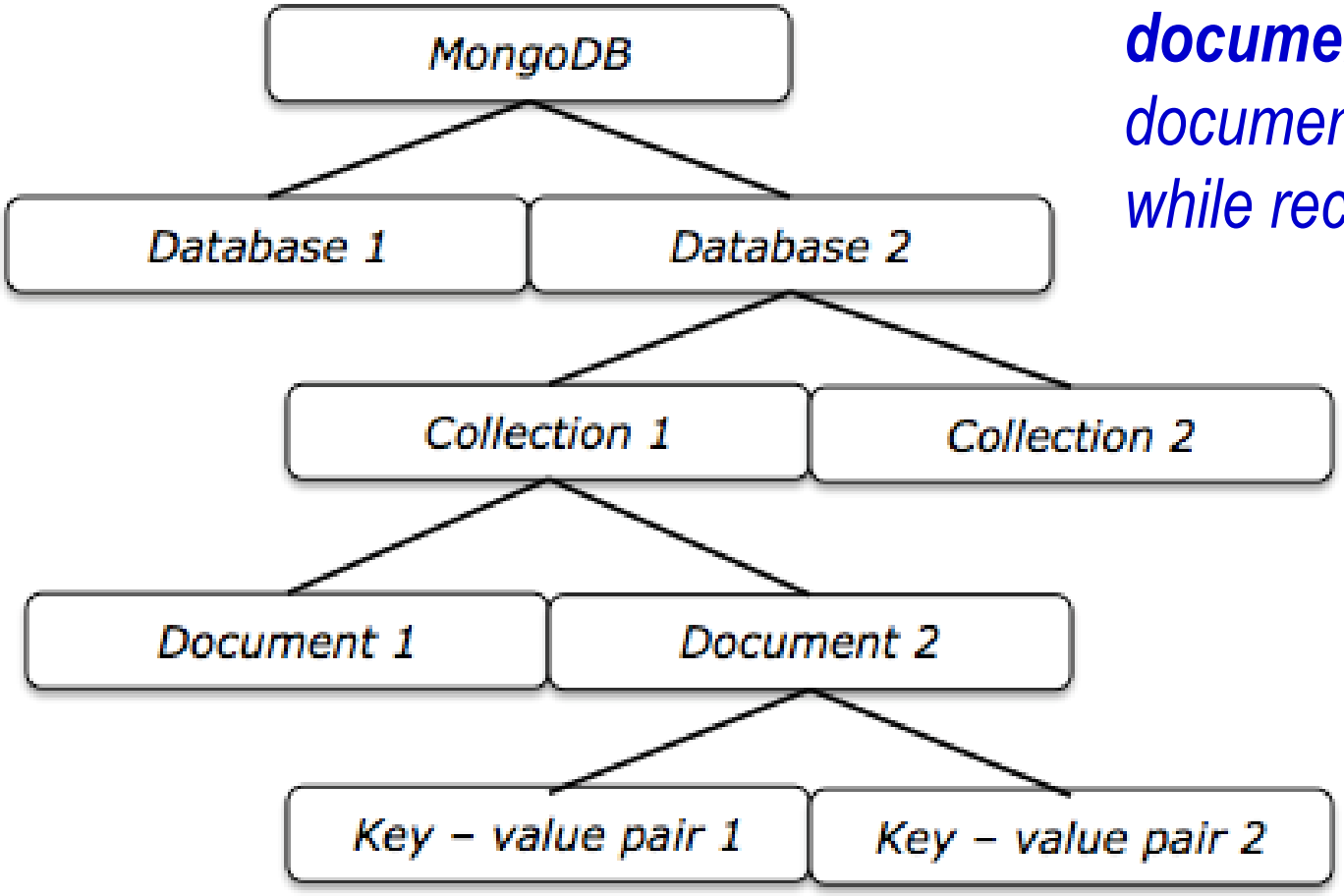
- A **database** is a collection of information from which a computer program can quickly access information
- Each row in a relational database table is called a **record**
- A **record** is a set of key-value pairs about a specific instance
- Each **column** in a database table is called a **field/attribute**
- **Fields** are the individual categories of information stored in a record

Graph DB



Graphic Source: Emil Eifrem and Philip Rathle, Neotechnology.com, 2013

Document-based DB

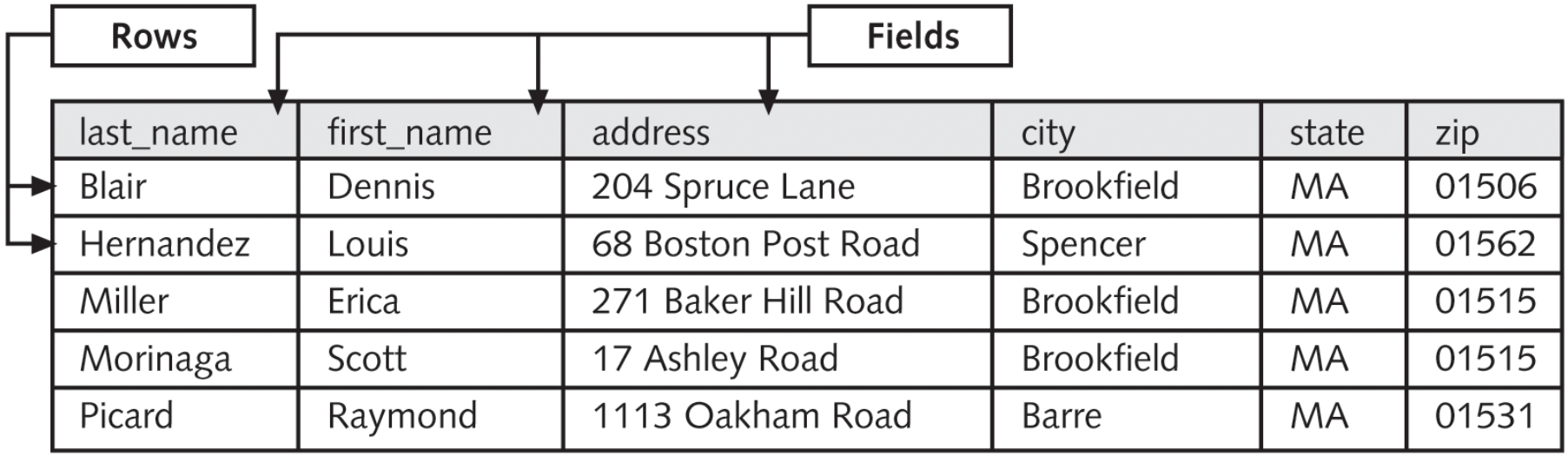


*The key difference between **document** and **record** is document can be nested while record must be flat (2D)*

*So every document is an **object** with properties and values…*

*Every collection is a **class**/group of objects*

Relational Databases



last_name	first_name	address	city	state	zip
Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Employee directory database

- A **flat-file database** stores information in a ***single*** table
- A **relational database** stores information across ***multiple*** related tables

Understanding Relational Databases

- **Relational databases** consist of one or more related tables
- A **primary table** is the main table in a relationship that is referenced by another table
- A **related table** (or “child table”) references a primary table in a relational database

Understanding Relational Databases (continued)

- A **primary key** is a field that contains a unique identifier for each record in a primary table
- A **primary key** is a type of index, which identifies records in a database to make retrievals and sorting faster
- A **foreign key** is a field in a related table that refers to the primary key in a primary table
- Primary and foreign keys link records across multiple tables in a relational database

One-to-One Relationships

- A **one-to-one relationship** exists between two tables when a related table contains exactly one record for each record in the primary table
- Create one-to-one relationships to break information into multiple, logical sets
- Information in the tables in a one-to-one relationship can be placed within a single table
- Make the information in one of the tables confidential and accessible only by certain individuals

One-to-One Relationships (continued)

Primary key

`Employees` table

employee_id	last_name	first_name	address	city	state	zip
101	Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
102	Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
103	Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
104	Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
105	Picard	Raymond	1113 Oakham Road	Barre	MA	01531

Foreign key

`Payroll` table

employee_id	start_date	pay_rate	health_coverage	year_vested	401k
101	2002	$21.25	none	na	no
102	1999	$28.00	Family Plan	2001	yes
103	1997	$24.50	Individual	na	yes
104	1994	$36.00	Family Plan	1996	yes
105	1995	$31.00	Individual	1997	yes

One-to-one relationship

One-to-Many Relationship

- A **one-to-many relationship** exists in a relational database when one record in a primary table has many related records in a related table
- Breaking tables into multiple related tables to reduce redundant and duplicate information is called **normalization**
- Provides a more efficient and less redundant method of storing this information in a database

One-to-Many Relationship (continued)

employee_id	last_name	first_name	language
101	Blair	Dennis	JavaScript
101	Blair	Dennis	ASP.NET
102	Hernandez	Louis	JavaScript
102	Hernandez	Louis	ASP.NET
102	Hernandez	Louis	Java
103	Miller	Erica	JavaScript
103	Miller	Erica	ASP.NET
103	Miller	Erica	Java
103	Miller	Erica	C++
104	Morinaga	Scott	JavaScript
104	Morinaga	Scott	ASP.NET
104	Morinaga	Scott	Java
105	Picard	Raymond	JavaScript
105	Picard	Raymond	ASP.NET

Table with redundant information

One-to-Many Relationship (continued)

`Employees` table

employee_id	last_name	first_name	address	city	state	zip
101	Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
102	Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
103	Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
104	Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
105	Picard	Raymond	1113 Oakham Road	Barre	MA	01531

`Languages` table ("many" side)

employee_id	language
101	JavaScript
101	ASP.NET
102	JavaScript
102	ASP.NET
102	Java
103	JavaScript
103	ASP.NET
103	Java
103	C++
104	JavaScript
104	ASP.NET
104	Java
105	JavaScript
105	ASP.NET

One record on the top table is linked to many records in the bottom table

Many-to-Many Relationship

- A **many-to-many relationship** exists in a relational database when many records in one table are related to many records in another table
  e.g. relationship between programmers and languages
- Must use a **junction table** which creates a one-to-many relationship for each of the two tables in a many-to-many relationship
- A junction table contains foreign keys from the two tables

Many-to-Many Relationship (continued)

`Employees` table

employee_id	last_name	first_name	address	city	state	zip
101	Blair	Dennis	204 Spruce Lane	Brookfield	MA	01506
102	Hernandez	Louis	68 Boston Post Road	Spencer	MA	01562
103	Miller	Erica	271 Baker Hill Road	Brookfield	MA	01515
104	Morinaga	Scott	17 Ashley Road	Brookfield	MA	01515
105	Picard	Raymond	1113 Oakham Road	Barre	MA	01531

`Languages` table

language_id	language
10	JavaScript
11	ASP.NET
12	Java
13	C++

*..1

*..1

`Experience` junction table

employee_id	language_id
101	10
101	11
102	10
102	11
102	12
103	10
103	11
103	12
103	13
104	10
104	11
104	12
105	10
105	11

Many-to-many relationship

Working with Database Management Systems

- A **database management system** (or DBMS) is an application or collection of applications used to access and manage a database
- A **schema** is the structure of a database including its tables, fields, and relationships
- A **flat-file database management system** is a system that stores data in a flat-file format
- A **relational database management system** (or RDBMS) is a system that stores data in a relational format

Examples of RDBMS

Working with Database Management Systems (continued)

Important aspects of database management systems:

- The structuring and preservation of the database file
- Ensuring that data is stored correctly in a database's tables, regardless of the database format
- Querying capability
- (also security)

Working with Database Management Systems (continued)

- A **query** is a structured set of instructions and criteria for retrieving, adding, modifying, and deleting database information
- **Structured query language** (or SQL – pronounced as sequel) is a standard data manipulation language used among many database management systems
- **Open database connectivity** (or ODBC) allows ODBC-compliant applications to access any data source for which there is an ODBC driver

Querying Databases with Structured Query Language

Common SQL keywords

Keyword	Description
DELETE	Deletes a row from a table
FROM	Specifies the tables from which to retrieve or delete records
INSERT	Inserts a new row into a table
INTO	Determines the table into which records should be inserted
ORDER BY	Sorts the records returned from a table
SELECT	Returns information from a table
UPDATE	Saves changes to fields in a record
WHERE	Specifies the conditions that must be met for records to be returned from a query

e.g. `select * from Employees`

WORKING WITH MYSQL DATABASES

Getting Started with MySQL

- MySQL is an open source database server, and it is fast and reliable.
- There are several ways to interface with a MySQL database server:
  - Using **MySQL Monitor**, a command-line program;

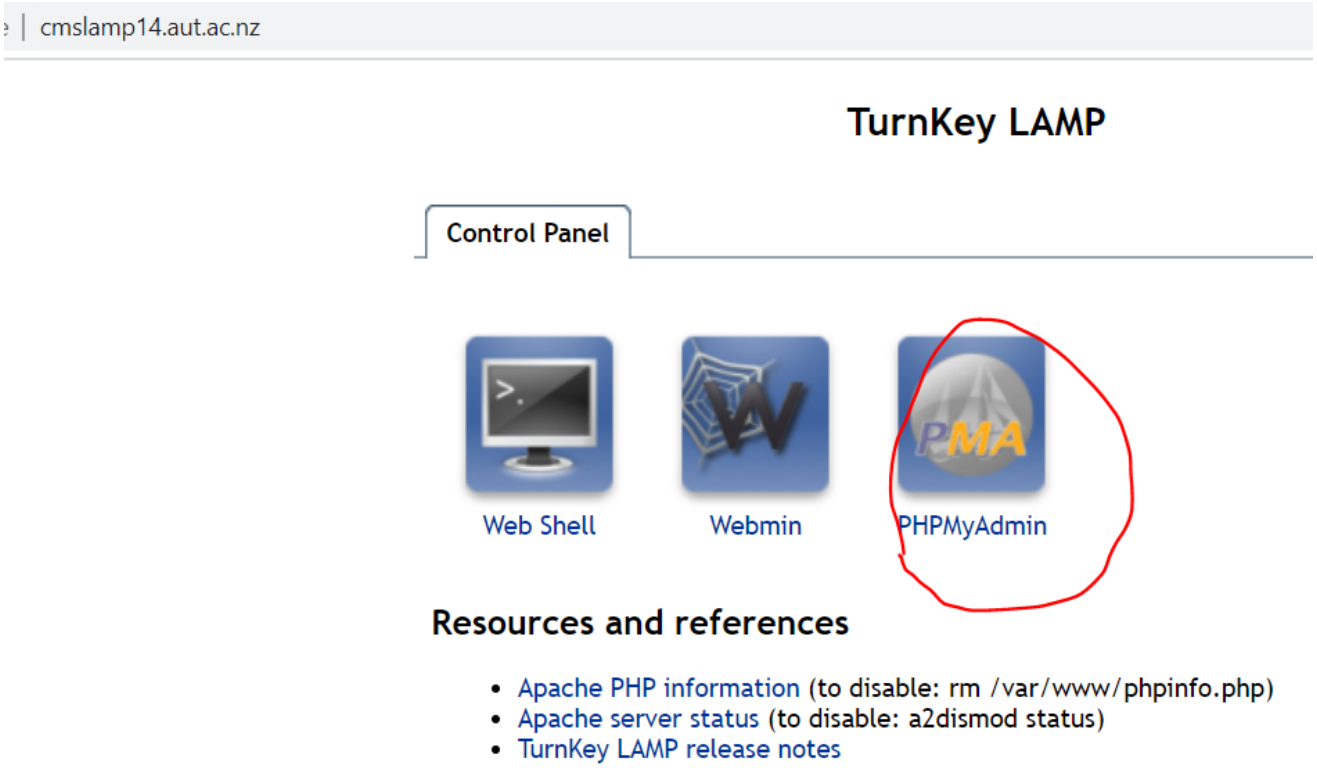
    or third part **GUI** client software, e.g., HeidiSQL
  - Using **phpMyAdmin**, a web interface program
  - Using **PHP database functions** within PHP scripts

*Right now we will access the MySQL database server using the web client "**phpMyAdmin**", in your lab, you might use HeidiSQL client.*

Next lecture we will begin accessing MySQL through PHP scripts.

Logging in to PHPMyAdmin

- *We will be accessing the MySQL database server at* **`cmslamp14.aut.ac.nz`** *Your acount and database will already be created*



- *Then login with your credential…*

Working with the MySQL Monitor

- In the `SQL` Tab, we can input any SQL statements

```
show databases;
use <your databse>;
show tables;
SELECT * FROM inventory;
```

- The SQL keywords entered are *not* case sensitive

Single quotes? Double quotes?

- In ANSI SQL
  - Single quotes are for strings.
  - double quotes quote object names (e.g. tables) which allows them to contain characters not otherwise permitted, or be the same as reserved words (Avoid this, really).
- So usually we use **single quotes** although MySQL allows them to be used interchangably for strings.

Understanding MySQL Identifiers

- Identifiers for databases, tables, fields, indexes, and aliases
- The case sensitivity of database and table identifiers depends on the operating system
  - Not case sensitive on Windows platforms
  - Case sensitive on UNIX/Linux systems
- MySQL stores each database in a directory of the same name as the database identifier
- Field and index identifiers are case insensitive on all platforms

Managing Databases and Tables

Creating Databases

Note: we do not have permission to create DB on cmslamp14…

- Use the `CREATE DATABASE` statement to create a new database:

```
CREATE DATABASE guitars;
Query OK, 1 row affected (0.02 sec)
```

- To use a new database, select it by executing the `use` *`database`* statement
- Before adding records to a new database, first define the tables and fields that will store the data

Deleting Databases

- You must be logged in as the root user or have privileges to delete a database
- Use the `DROP DATABASE` statement to remove all tables from the database and to delete the database
- The syntax for the `DROP DATABASE` statement is:

```
DROP DATABASE database;
```

Selecting Databases

- Use the **`SHOW DATABASES`** statement to view the databases that are available
- Use the **`USE DATABASE`** statement to select the database to work with
- Use the **`SELECT DATABASE()`** statement to display the name of the currently selected database
- The `mysql` database is installed to contain user accounts and information that is required for installation of the MySQL database server
- The `test` database is installed to ensure that the database server is working properly

Creating Tables

- The `CREATE TABLE` statement specifies the table and column names and the data type for each column
- The syntax for the `CREATE TABLE` statement is:

```
CREATE TABLE table_name (column_name TYPE, ...);
```

- Execute the `USE` statement to select a database before executing the `CREATE TABLE` statement

Creating Tables (continued)

Common MySQL field data types

Type	Range	Storage
BOOL	-128 to 127; 0 is considered false	1 byte
INT or INTEGER	-2147483648 to 2147483647	4 bytes
FLOAT	-3.402823466E+38 to -1.175494351E-38, 0, and 1.175494351E-38 to 3.402823466E+38	4 bytes
DOUBLE	-1.7976931348623157E+308 to -2.2250738585072014E-308, 0, and 2.2250738585072014E-308 to 1.7976931348623157E+308	8 bytes
DATE	'1000-01-01' to '9999-12-31'	Varies
TIME	'-838:59:59' to '838:59:59'	Varies
CHAR(*m*)	Fixed length string between 0 to 255 characters	Number of bytes specified by *m*
VARCHAR(*m*)	Variable length string between 1 to 65,535 characters	Varies according to the number of bytes specified by *m*

DECIMAL(M, p) : stores an exact representation of the number; there is no approximation of the stored value. M: total digits; P: precision.

Deleting Tables

- The `DROP TABLE` statement removes all data and the table definition
- The syntax for the `DROP TABLE` statement is:

```
DROP TABLE table;
```

Showing Tables

- Use the `SHOW TABLES` statement to show the non temporary tables of the selected database

WORKING WITH DATA RECORDS

Adding Records

- Use the `INSERT` statement to add individual records to a table
- The syntax for the `INSERT` statement is:

```
INSERT INTO table_name VALUES(value1, value2, ...);
```

- The values entered in the `VALUES` list must be in the same order in which you defined the table fields
- Specify `NULL` in any fields for which you do not have a value
- Add multiple records, use the `LOAD DATA` statement

```
LOAD DATA LOCAL INFILE 'file_path_name' INTO
TABLE table_name;
```

Updating Records

- To update records in a table, use the `UPDATE` statement
- The syntax for the `UPDATE` statement is:

```
UPDATE table_name
SET column_name=value
WHERE condition;
```

  - The `UPDATE` keyword specifies the name of the table to update
  - The `SET` keyword specifies the value to assign to the fields in the records that match the condition in the `WHERE` keyword

Deleting Records

- Use the `DELETE` statement to delete records in a table
- The syntax for the `DELETE` statement is:

```
DELETE FROM table_name
WHERE condition;
```

- The `DELETE` statement deletes all records that match the condition
- To delete all the records in a table, leave off the `WHERE` keyword

Retrieving Records

- Use the `SELECT` statement to retrieve records from a table:

```
SELECT criteria FROM table_name;
```

- Use the asterisk (*) wildcard with the `SELECT` statement to retrieve all fields from a table
- To return multiple fields, separate field names with a comma

```
SELECT model, quantity FROM inventory;
```

Retrieving Records – Sorting

- Use the `ORDER BY` keyword with the `SELECT` statement
  to perform an alphanumeric sort of the results returned from a query

```
SELECT make, model FROM inventory ORDER BY make, model;
```

- To perform a reverse sort, add the `DESC` keyword after the
  name of the field by which you want to perform the sort

```
SELECT make, model FROM inventory ORDER BY make DESC, model;
```

Retrieving Records – Filter

- The **criteria** portion of the `SELECT` statement determines which fields to retrieve from a table
- You can also specify which records to return by using the `WHERE` keyword

```
SELECT * FROM inventory WHERE make='Martin';
```

- Use the keywords `AND` and `OR` to specify more detailed conditions about the records you want to return

```
SELECT * FROM inventory WHERE make='Washburn` AND
price<400;
```

MongoDB

MongoDB

- A document-oriented NoSQL DB
- Terms

RDBMS		MongoDB
Database		Database
Table, View		Collection
Row		Document (JSON, BSON/Binary JSON)
Column		Field
Index		Index
Join		Embedded Document
Foreign Key		Reference
Partition		Shard

MongoDB is also a cloud DB

- Local DB
  - Start the local MongoDB:
    - In terminal go to "`C:\Program Files\MongoDB\Server\3.6\bin`" dir

      and type `mongod.exe`
- Cloud access: using the connection string copied from your https://cloud.mongodb.com account; remember to add client machine IP to the whitelist

e.g.

```
mongodb://admin:<PASSWORD>@cluster0-shard-00-00-
syy3n.mongodb.net:27017,cluster0-shard-00-01-
syy3n.mongodb.net:27017,cluster0-shard-00-02-
syy3n.mongodb.net:27017/test?ssl=true&replicaSet=Cluster0-shard-
0&authSource=admin
```

Clients

- MongoDB Shell: Terminal (mongo.exe)
- MongoDB Compass: a graphical client
- Various APIs
  - JavaScript, PHP, Node.js, Python, Ruby, Perl, Java, C#, C++, Haskell

A document/row

```
> db.user.findOne({age:39})
{
        "_id" : ObjectId("5114e0bd42…"),
        "first" : "John",
        "last" : "Doe",
        "age" : 39,
        "interests" : [
                "Reading",
                "Mountain Biking ]
        "favorites": {
                "color": "Blue",
                "sport": "Soccer"}
}
```

`_id` is automatically generated by MongoDB when the doc is created

Show db and collections

- In Mogodb terminal
  - “show dbs”
  - “show collections”

CRUD

- Create
  - db.collection.insert(<document>)
  - db.collection.save(<document>)
  - db.collection.update(<query>, <update>, { upsert: true })
- Read
  - db.collection.find(<query>, <projection>)
  - db.collection.findOne(<query>, <projection>)
- Update
  - db.collection.update(<query>, <update>, <options>)
- Delete
  - db.collection.remove(<query>, <justOne>)

CRUD example

```
> db.user.insert({
        first: "John",
        last : "Doe",
        age: 39
})
```

```
> db.user.find (
{
        "first" : "John",
        "last" : "Doe",
        "age" : 39
})
```

```
> db.user.update(
        {"_id" :
ObjectId("51…")},
        {
            $set: {
                age: 40,
                salary: 7000}
        })
```

```
> db.user.remove({
        "first": /^J/
})
```

References

- http://people.inf.elte.hu/kiss/13kor/Mongodb.ppt