Finishing SQL Join
PHP
Server-side Programming
cs380

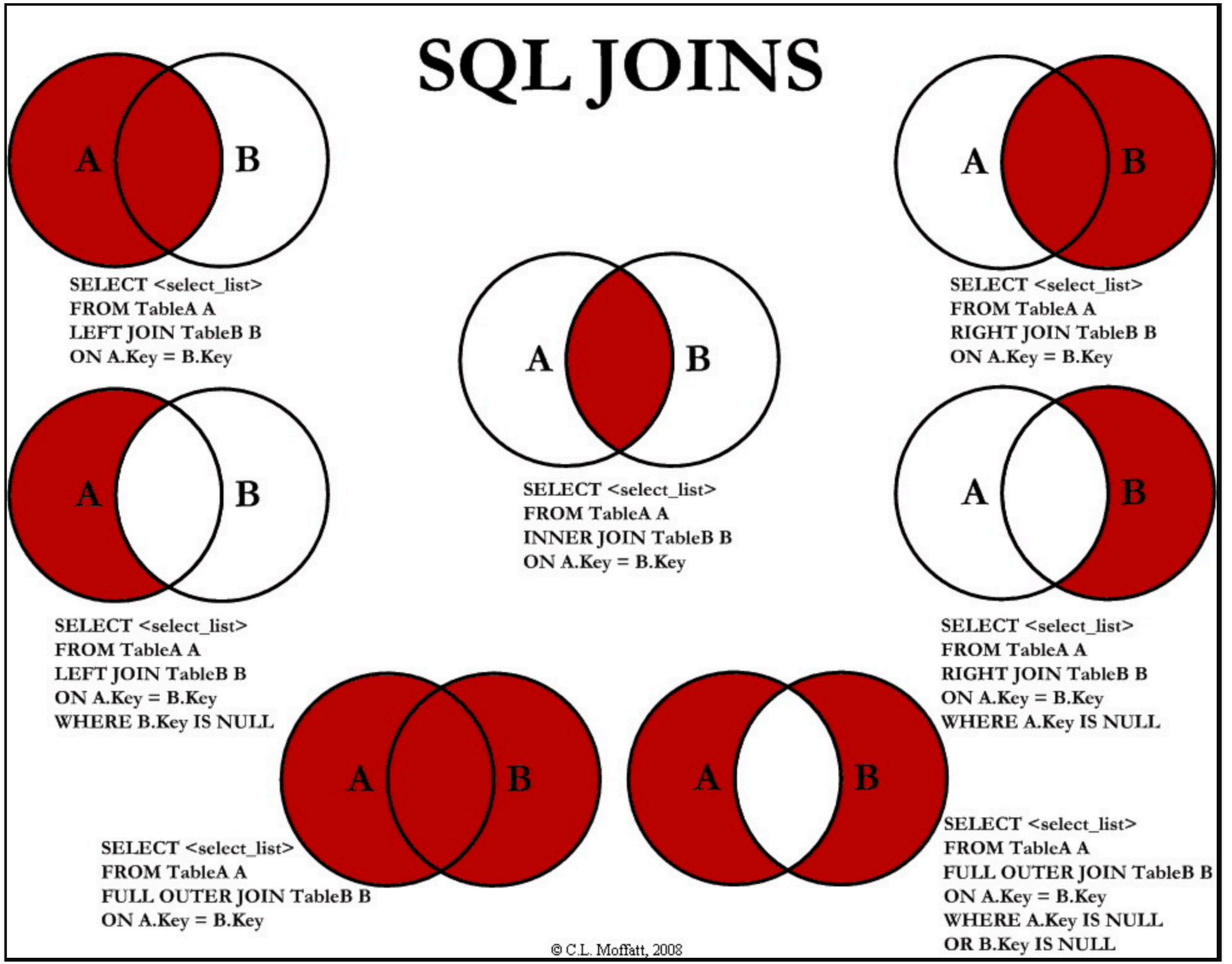
G Towell
Sep 21

URLs

Millicent

https://docs.google.com/presentation/d/1P1vytAbK4eO0FMimBn8lxz3XL025Mbnys3mv6uXO5RU/edit#slide=id.g97a46420fb_0_0

- Inner
- Left
- Right
- Full Outer
- “”
  - same as inner



https://www.codeproject.com/Articles/33052/Visual-Representation-of-SQL-Joins

Joins

```
Databases
— t1
id  name
1   Mary
2   Alis

-- t2
id  name
1   Mary
3   Katy
```

```
select * from t1 left join t2 on t1.name=t2.name;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
|    1 | mary |    1 | mary |
|    2 | alis | NULL | NULL |
+------+------+------+------+
```

```
select * from t1 inner join t2 on t1.name=t2.name;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
|    1 | mary |    1 | mary |
+------+------+------+------+
```

```
select * from t1 right join t2 on t1.name=t2.name;
+------+------+------+------+
| id   | name | id   | name |
+------+------+------+------+
|    1 | mary |    1 | mary |
| NULL | NULL |    1 | katy |
+------+------+------+------+
```

More Selecting

Hurricanes

```
show tables;
+--------------------+
| Tables_in_hurricane |
+--------------------+
| hurricane          |
| observation        |
| typename           |
+--------------------+
```

```
describe typename;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| type  | varchar(2)  | NO   | PRI | NULL    |       |
| name  | varchar(30) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

```
describe hurricane;
+-------+-------------+------+-----+---------+-------+
| Field | Type        | Null | Key | Default | Extra |
+-------+-------------+------+-----+---------+-------+
| HID   | char(8)     | NO   | PRI | NULL    |       |
| Name  | varchar(25) | YES  |     | NULL    |       |
+-------+-------------+------+-----+---------+-------+
```

```
describe observation;
+-------------+-----------------------------------------------------------------------------+------+-----+---------+-------+
| Field       | Type                                                                        | Null | Key | Default | Extra |
+-------------+-----------------------------------------------------------------------------+------+-----+---------+-------+
| HID         | char(8)                                                                     | YES  | MUL | NULL    |       |
| date        | date                                                                        | YES  |     | NULL    |       |
| time        | time                                                                        | YES  |     | NULL    |       |
| type        | enum('','TY','PT','ET','ST','TD','TS','HU','EX','SD','SS','LO','WV','DB') | YES  |     | NULL    |       |
| latitude    | float                                                                       | YES  |     | NULL    |       |
| latitudehemi | char(1)                                                                    | YES  |     | NULL    |       |
| longitude   | float                                                                       | YES  |     | NULL    |       |
| longitudehemi | char(1)                                                                   | YES  |     | NULL    |       |
| maxSustained | int(11)                                                                    | YES  |     | NULL    |       |
+-------------+-----------------------------------------------------------------------------+------+-----+---------+-------+
```

Searching for Hurricanes

- Name of the hurricane and the Easternmost point (in the western hemisphere) for all hurricanes whose names start with A
- Name of Easternmost hurricane
  - Note this uses a join on a non-key
    - This can be very slow.
    - BUT there is only 1 observation on one side of join
- Name of first observed storm in database (with the full name of the type) and an actual name!
- First observation for each named storm in database (with the full name of the type)
- Last observation of each named hurricane when the storm was of type ‘db’.

- Be very careful when interpreting answers when you have grouped and maxed. The data in the row will not necessarily be form the max row

Thoughts about efficiency

- Avoid queries with multiple tables in “from”
  - This will generate a new temporary table with |M|*|N| rows
    - e.g. select count(*)from hurricane as hu, observation as ob;
      - 51840*1893 = 98133120 — the query took 3 seconds to run!
    - If you throw a where onto this, it will still generate the huge table before cutting it down
      - select hu.hid, hu.name, ob.type from hurricane as hu, observation as ob where hu.hid=ob.hid
  - better to use join
    - select hu.hid, hu.name, ob.type from hurricane as hu inner join observation as ob on hu.hid=ob.hid;

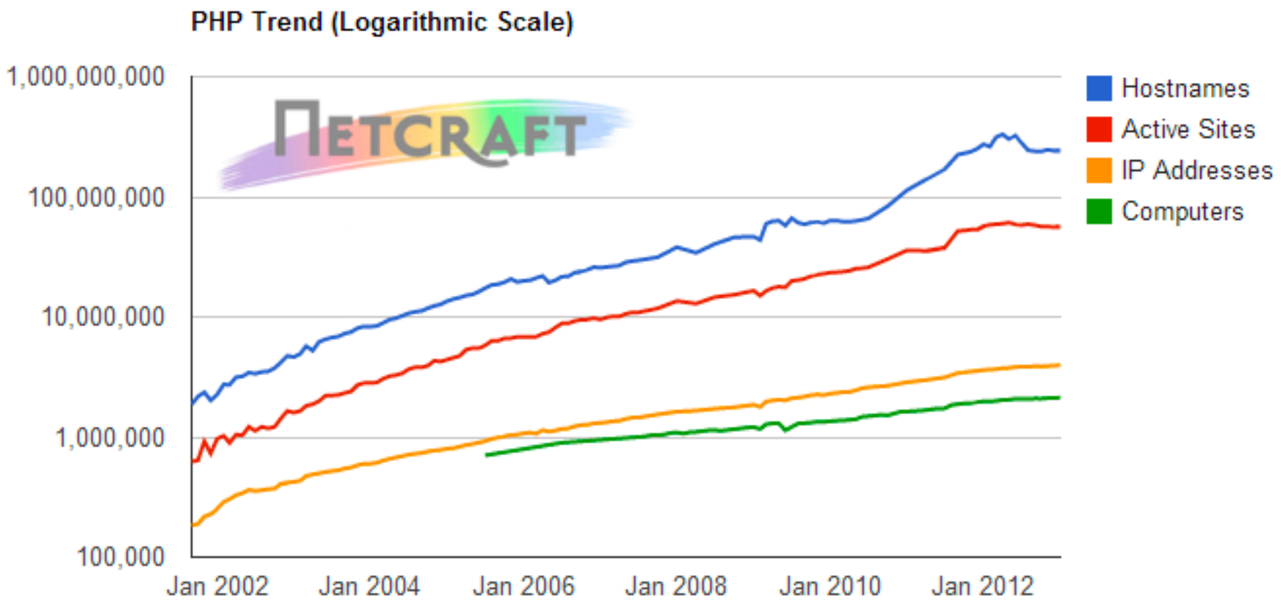
Joining efficiency

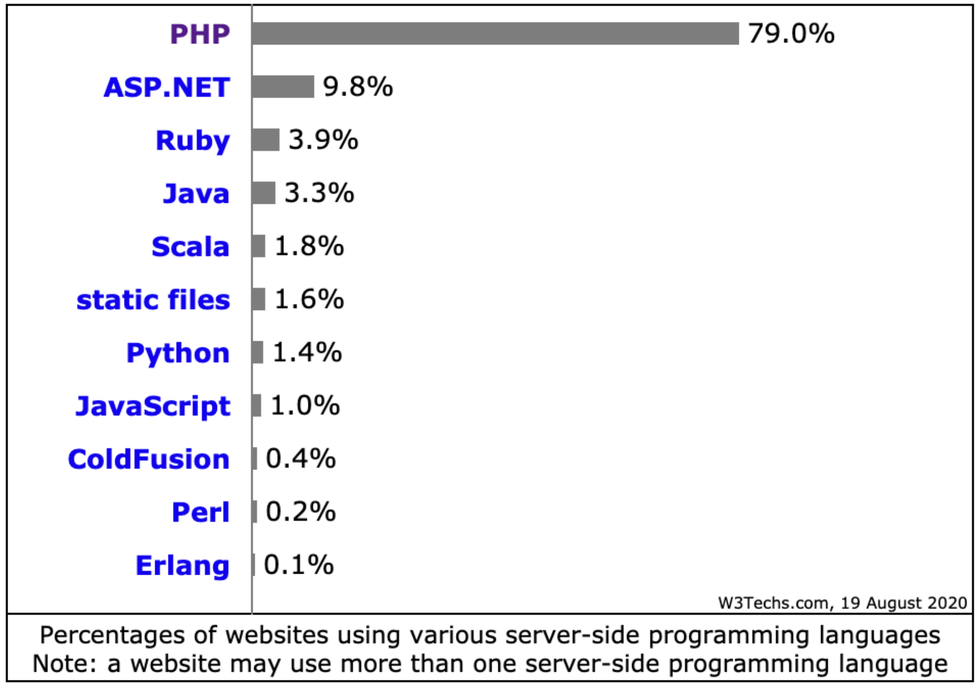
inner join only

- **join order matters!**
  - because if we can join two tables that will reduce the number of rows needed to be processed by subsequent steps, then our performance will improve.
- SO as a general rule:
  - Specify the largest table first.
  - Next, specify the smallest table. The contents of the second, third, and so on tables are all transmitted across the network. You want to minimize the size of the result set from each subsequent stage of the join query. The most likely approach involves joining a small table first, so that the result set remains small even as subsequent larger tables are processed.
  - Join the next smallest table, then the next smallest, and so on.
  - For example, if you had tables BIG, MEDIUM, SMALL, and TINY, the logical join order to try would be BIG, TINY, SMALL, MEDIUM.

The dominant server-side programming language

Module	July 2020 Count	July 2020 %	June 2020 Count	June 2020 %	Growth %
OpenSSL	1,710,836	6.69	1,662,363	6.48	3.28
PHP	1,074,998	4.20	1,086,952	4.23	-0.75
mod_ssl	343,105	1.34	364,458	1.42	-5.52
mod_bwlimited	278,983	1.09	290,458	1.13	-3.61
mod_jk	214,091	0.84	158,476	0.62	35.58
DAV	120,711	0.47	126,596	0.49	-4.31
perl	104,491	0.41	110,742	0.43	-5.31
Python	90,591	0.35	90,887	0.35	0.03





Intro to PHP

- PHP programs must have a .php extension
  - This tells the web server that the file contains PHP.
  - The web server then runs the PHP program, generates HTML and spits out the HTML
  - Users cannot see the source (if server is properly configured)
- Simplest PHP programs are just HTML! There is no PHP at all
- To tell server that there is PHP to execute
  - <?PHP ..... ?>
  - Can have as many <?php ... ?> as you want

On Server

```
file hw1.php
<html>
<body>
Hello World!
</body>
</html>
```

```
file  hw2.php
<html>
<body>
Hello World!
<?php
?>
</body>
</html>
```

in Browser

```
<html>
<body>
Hello World!
</body>
</html>
```

Running PHP

- Expectation is to use the browser
- PHP is installed only on comet

- Can also run from command line
  - COMET> php -f xxx.php

- LAMP

Echo and Print

- Comments
  - Java/C style
  - Also “# ...”
- PHPs echo and print functions are basically the same
- Generally will use these to put out well-formatted html from within PHP sections
- C-like printf and sprintf
  - printf???
- Concatenate strings with “.”

On Server

in Browser

file hw3.php

```php
<?PHP
print("<html>");
print("<body>");
print("Hello" . "World");
print("</body>");
print("</html>");
?>
```

Variables

- always preceded by $
- NOT typed
- CASe Sensitive
- Global variables exist across <? php ?> blocks

hw4.php

```
<html>
<body>
<?PHP
//ini_set('display_errors', 'Off');

$count = 0;
print("Hello" . $count);
$count += 10;
?>
World
<?PHP
print("Hello" . $count);
?>
<br>Again
<?PHP
print("Hello" . count);
?>
</body>
</html>
```

functions

- function name(parameters) {
  body
  }
- default is parameters are passed by value
- “return” for non-null return value
- GLOBAL indicates function should use global variable (rather than create a new local variable)

```
<html>
<body>
<?PHP
$count = 0;
function printer($strin) {
    $count = 5;
    GLOBAL $count;
    $count += 1;
    echo "$strin $count<br>";
}
printer("Hello");
?>
World
<?PHP
printer("Hello");
printer("World");
?>
</body>
</html>
```

Arrays/Maps and loops

- PHP does not distinguish between
  - $a[1]="43"
  - $a["43"]=1
  - arrays are really maps
- usual loops: for, while, do..while
- foreach(array as $value) {
  }
  - Java: for(int iii : arrayOfInt)
- foreach(array as $key=>$value)
- It is not an error to read off end of array
  - since arrays are maps there is no real end
  - just returns null

```
<html>
<body>
<?PHP
$cc = array(1,1,2,3,5,8,13,21,34);
$cq = array(1=>2, 2=>3, 4=>7, 7=>17);
for ($i=0; $i<count($cc); $i++) {
    print($i . " " . $cc[$i] . "<br>" );
}
?>
World
<?PHP
foreach($cq as $val) {
    print($val . "<br>");
}
for ($i=0; $i<8; $i++) {
    if ($cq[$i]!=Null) {
        print($i . " " . $cq[$i] . "<br>" );
    }
}
foreach ($cq as $ky=>$val) {
    print("$ky $val<br>");
}
?>
</body>
</html>
```

array.php

Foreach

be warned

- akin to java iterator
  - so changes to array do not change the keys and values

```
<html>
<body>
<?php
$a = array();
for ($i=0; $i<10; $i++) {
    $a[$i] = 2*$i;
}
foreach ($a as $k=>$v) {
    echo "$k  $v  $a[$k] <br>";
    if (array_key_exists($v, $a)) {
        $a[$v] = 5*$v;
    }
}
?>
</body>
</html>
```

foreach.php

Super Global variables

- $_SERVER
  - where the script is running
  - and lots of other stuff
- getallheaders()
  - a subset of $_SERVER
  - visible in browser developer console
- $_COOKIE
  - the cookies set for this page
- $_GET
- $_POST
  - submissions from forms
  - $_REQUEST

Server.php

```
<html><body>
<?PHP
print_r($_SERVER);
?>
</body></html>
```

headers.php

```
<html>
<head>
<style>
table, tr, td, th { border:5px solid #dd0000; border-collapse:
</style>
</head>
<body>
<table>
<?PHP
foreach (getallheaders() as $ky=>$val) {
    print("<tr><td>$ky</td><td>$val</td></tr>");
}
?>
</table>
</body>
</html>
```

PHP from the command line

- invoke
- UNIX>php something.php
- UNIX>php -f sometings.php

- php args.php now=then then=name name=geoff

args.php

```
<?php
if(defined('STDIN') ) {
  echo "Running from CLI\n";
  $_GET=array();
  }
if ($argc>1) {
   for ($i=1; $i<$argc; $i++) {
       echo $argv[$i] . "\n";
       $av = explode("=", $argv[$i]);
       $_GET[$av[0]]=$av[1];
   }
}
print_r($_GET);
```

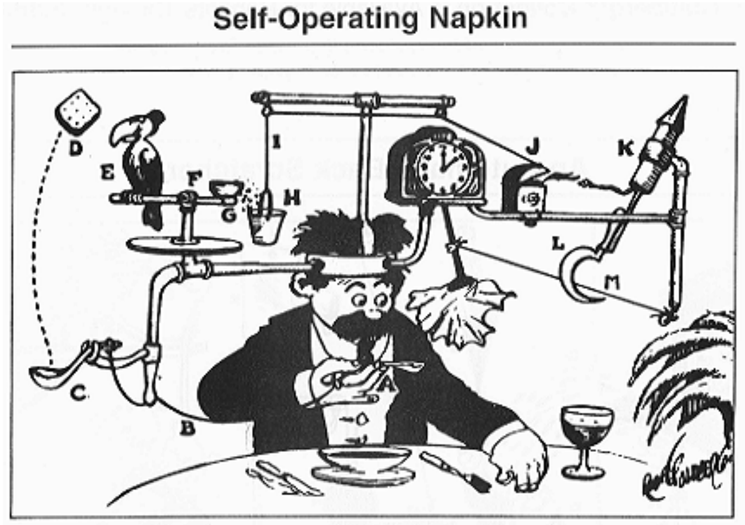
Running PHP

- You must run PHP on machines onto which PHP has been installed
- To view php programs through a browser the web server must be configured for PHP
- Macs have php installed. You can configure the internal web server for PHP is you are so motivated

- comet.cs.brynmawr.edu is such a machine
  - Put files with .php extension into you public_html directory
  - http://comet.cs.brynmawr.edu/~gtowell/380/hw1.php
    - If you miss “comet” what happens will depend on your browser and the server. On my mac, the file download to my computer

PHP Practice

- Write PHP to create a table that prints the first 50 fibonacci numbers
- Write PHP to create a table that prints all prime numbers less than 1000.
  - Use the “sieve of Eratosthenes” algorithm
  - Use at least 1 function and 1 global array for this task.
  - https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes
  - rewrite your sieve using “map” rather than array in which initial map only stores potentials that are no multiple of 2 or 3.
- Write a “Rube Goldberg”-ish hello world program in PHP. Use arrays, functions, loops, ...
  - in the end it should just print “hello world”



Next Class: PHP for interactive web pages

- Get / Post