Database Basics — *We'll see some PHP code in practice*

MySQL Prepared Statements and a Benchmark
Code: https://github.com/ewbarnard/PreparedStatements
Slides: https://github.com/sketchings/beginning-php-training/tree/master/sessions/09-database

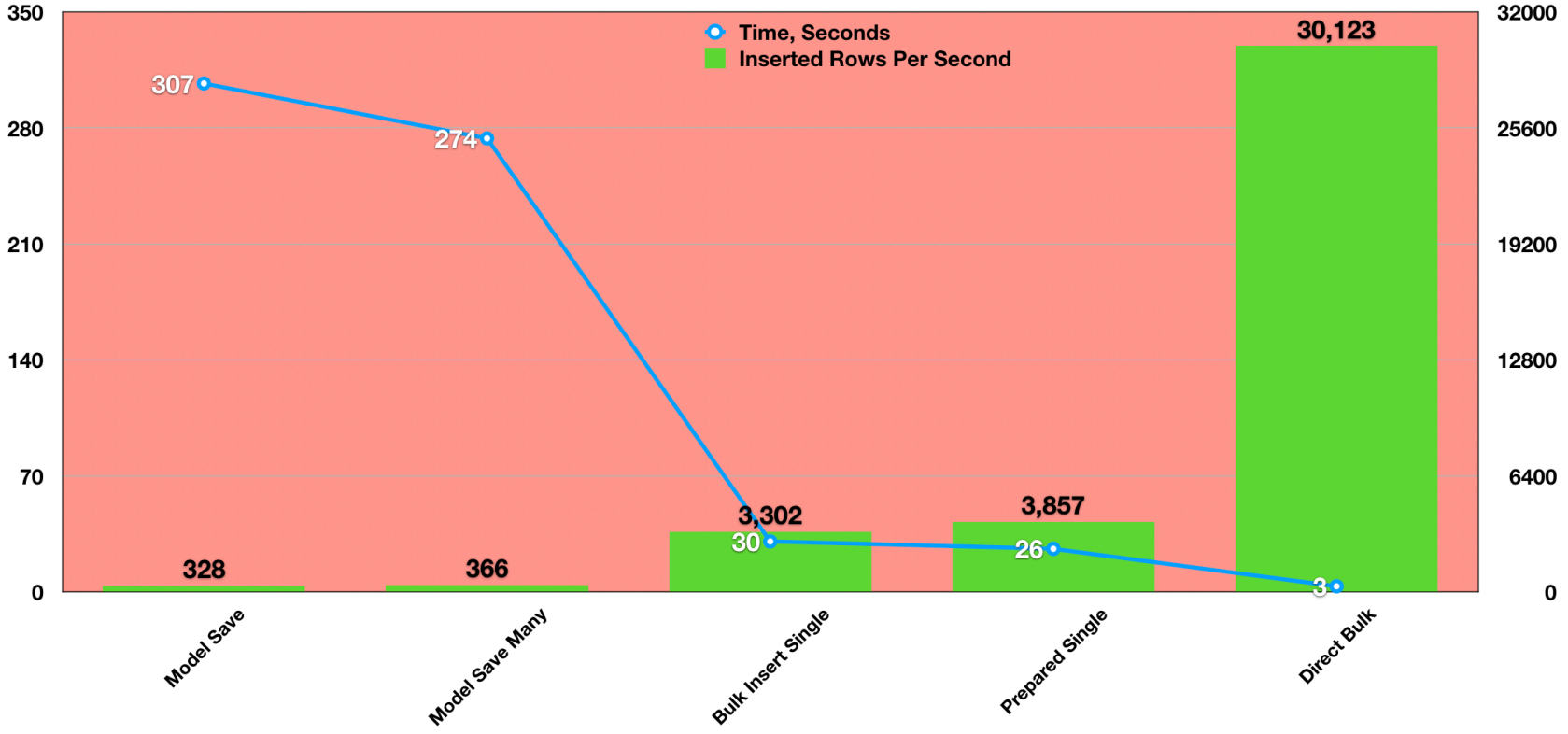
2

The Plan

- **Benchmark**
- Prepared Statements
- Table Design
- Prepared-Statement Utility

Benchmark: 100,000 Row Inserts

Benchmark on my MacBook Pro, PHP 5.6, CakePHP 3.5, MySQL 5.5



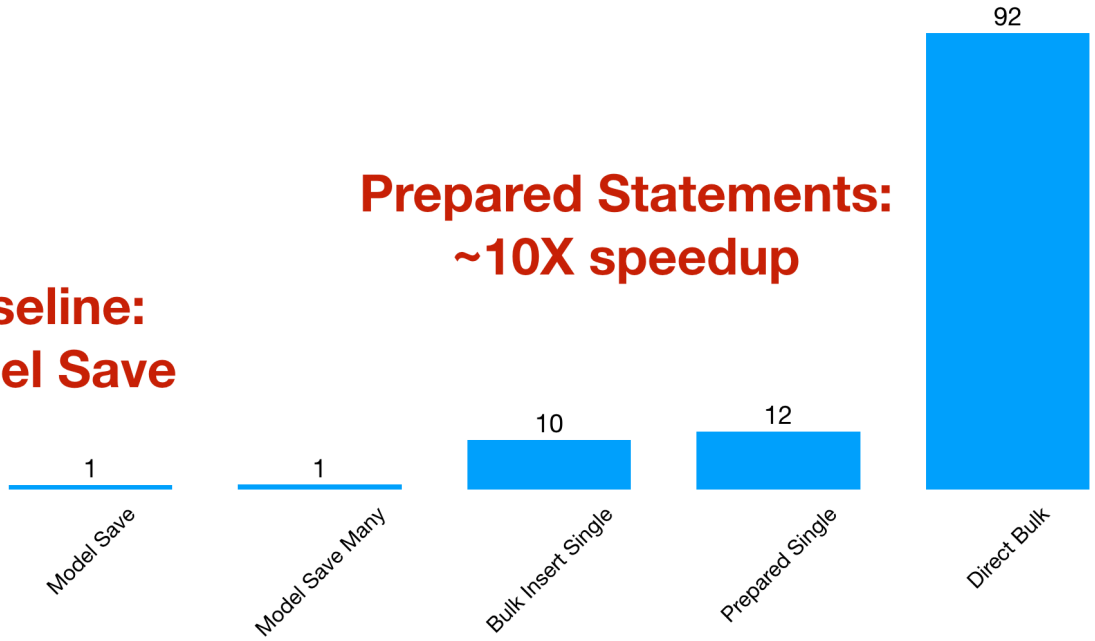
Benchmark: 100,000 Row Inserts

Same benchmark run, showing speedup factor

Baseline: Model Save

Prepared Statements: ~10X speedup

Hand-coded bulk insert: ~100X speedup



Benchmark Report

```php
public function main() {
    $this->prepareStatements();
    $this->loadExport();

    $passes = 0;
    while ($passes++ < static::$passes) {
        $this->benchmark();
    }

    $this->report();
    return 0;
}

private function report() {
    $this->resultQuery->execute([]);
    $rows = $this->resultQuery->fetchAll( type: 'assoc');
    $this->verbose(sprintf( format: '%-20s %-10s %-20s %-10s',
        args: 'Method', _: 'Seconds', 'Rows Per Second', 'Gain'));
    foreach ($rows as $row) {
        $this->verbose(sprintf( format: '%-20s %-10s %-20s %-10s',
            $row['method'], $row['seconds'], $row['rows_per_second'], $row['gain']));
    }
}
```

```
time bin/cake benchmark --verbose
Method               Seconds    Rows Per Second      Gain
modelSave            306.85     327.51               1.0X
modelSaveMany        273.67     366.40               1.1X
bulkInsertSingle     30.38      3,301.56             10.1X
preparedSingle       25.93      3,857.06             11.8X
preparedBulk         3.32       30,123.15            92.0X

real    53m39.946s
```

Input File as CSV Text Data

```
1       motion,lat,lon,ele,time,nearest,distance,feet,seconds,mph,climb
2       Drive,47.355681,-122.503923,110.09,"2017-09-04 12:37:00","Marshall House",60940,326,5,44.43,-3.06
3       Drive,47.353894,-122.503930,110.44,"2017-09-04 12:37:10","Marshall House",60848,652,10,44.45,0.35
4       Drive,47.352840,-122.503945,113.54,"2017-09-04 12:37:16","Marshall H    ,60799,385,6,43.69,3.10
5       Drive,47.351188,-122.504016,113.48,"2017-09-04 12:37:26","Marshall H    ,60740,603,10,41.11,-0.06
6       Drive,47.349998,-122.504038,109.54,"2017-09-04 12:37:33","Marshall H    ,60693,434,7,42.29,-3.94
7       Drive,47.348913,-122.504053,106.38,"2017-09-04 12:37:39","Marshall      396,6,44.98,-3.16
8       Drive,47.347870,-122.504062,103.11,"2017-09-04 12:37:45","Marshall      380,6,43.24,-3.27
9       Drive,47.346900,-122.504326,99.99,"2017-09-04 12:37:51","Marshall       60,6,40.89,-3.12
99996   Drive,46.769259,-121.714755,1395.78,"2017-09-05 12:08:09","Reflection  ,126,5,29.13,-3.50
99997   Drive,46.769789,-121.714475,1392.29,"2017-09-05 12:08:14","Reflection  3,205,5,28.01,-3.49
99998   Drive,46.770157,-121.714278,1389.23,"2017-09-05 12:08:18","Reflection  8,143,4,24.37,-3.06
99999   Drive,46.770527,-121.714059,1388.82,"2017-09-05 12:08:22","Reflection L 7,146,4,24.83,-0.41
100000  Drive,46.770837,-121.713384,1385.45,"2017-09-05 12:08:28","Reflection L 23,203,6,23.07,-3.37
100001  Drive,46.770657,-121.712603,1382.75,"2017-09-05 12:08:34","Reflection L 08,206,6,23.40,-2.70
100002
```

Imported Result

Destination Table

benchmark @prepared_statem...

id	motion	lat	lon	ele	time	nearest	distance	feet	seconds	mph	climb
919	Drive	46.758021	-122.051264	499.72	2017-09-04 16:14:32	Rainier Mountaineering Inc	3967	462	6	52.55	3.07
920	Drive	46.758338	-122.050151	502.75	2017-09-04 16:14:36	Rainier Mountaineering Inc	3673	301	4	51.35	3.03
921	Drive	46.758722	-122.048481	506.12	2017-09-04 16:14:42	Rainier Mountaineering Inc	3239	440	6	50.03	3.37
922	Drive	46.759055	-122.046171	509.13	2017-09-04 16:14:50	Rainier Mountaineering Inc	2650	590	8	50.28	3.01
923	Drive	46.759201	-122.045079	511.29	2017-09-04 16:14:54	Rainier Mountaineering Inc	2373	278	4	47.39	2.16
924	Drive	46.759404	-122.043518	514.38	2017-09-04 16:15:00	Rainier Mountaineering Inc	1976	397	6	45.12	3.09
925	Drive	46.759617	-122.041816	517.59	2017-09-04 16:15:07	Rainier Mountaineering Inc	1545	432	7	42.12	3.21
926	Drive	46.759646	-122.041580	517.85	2017-09-04 16:15:08	Rainier Mountaineering Inc	1485	60	1	40.85	0.26
927	Drive	46.759935	-122.039251	521.09	2017-09-04 16:15:18	Rainier Mountaineering Inc	897	592	10	40.33	3.24
928	Drive	46.759987	-122.036713	522.72	2017-09-04 16:15:28	Rainier Mountaineering Inc	263	635	10	43.26	1.63
929	Drive	46.759668	-122.034737	524.75	2017-09-04 16:15:37	Rainier Mountaineering Inc	268	507	9	38.43	2.03
930	Drive	46.759019	-122.032146	528.12	2017-09-04 16:15:49	Rainier Mountaineering Inc	954	689	12	39.17	3.37
931	Drive	46.758261	-122.029086	530.26	2017-09-04 16:16:04	Rainier Mountaineering Inc	1767	813	15	36.96	2.14
932	Drive	46.757861	-122.027485	527.09	2017-09-04 16:16:11	Rainier Mountaineering Inc	2193	426	7	41.48	-3.17
933	Drive	46.757041	-122.024016	525.59	2017-09-04 16:16:24	Rainier Mountaineering Inc	3109	917	13	48.10	-1.50
934	Drive	46.756895	-122.021811	526.98	2017-09-04 16:16:31	Rainier Mountaineering Inc	3647	554	7	53.93	1.39

Load CSV Data

```
95      private function loadExport() {
96          $page = file_get_contents(static::$path);
97          $this->lines = explode( delimiter: "\n", $page);
98          foreach ($this->lines as $line) {
99              if ($line !== '') {
100                 if (!count($this->header)) {
101                     $this->header = str_getcsv($line);
102                 } else {
103                     $this->import[] = array_combine($this->header, str_getcsv($line));
104                 }
105             }
106         }
107         $this->count = (float)count($this->import);
108     }
```

Run the Benchmark Steps

```
110     private function benchmark() {
111         $this->modelSave();
112         $this->modelSaveMany();
113         $this->bulkInsertSingle();
114         $this->preparedSingle();
115         $this->preparedBulk();
116     }
```

Baseline: Model Save

Framework Model saves one row at a time (baseline 1X performance gain)

```
118     private function modelSave() {
119         $this->beginRun();
120
121         $table = TableRegistry::get( alias: 'Benchmark');
122         foreach ($this->import as $item) {
123             $entity = $table->newEntity($item);
124             $table->save($entity);
125         }
126         $this->flushTables();
127
128         $this->endRun( caller: __FUNCTION__);
129     }
```

Housekeeping

Begin Run, End Run

```
131     private function beginRun() {
132         $this->truncate();
133         $this->begin = microtime( get_as_float: true);
134     }
135
136     private function truncate() {
137         $this->connection->query( sql: 'truncate table prepared_statements.benchmark');
138     }
139
140     private function flushTables() {
141         $this->connection->query( sql: 'flush tables prepared_statements.benchmark');
142     }
143
144     private function endRun($caller) {
145         $interval = microtime( get_as_float: true) - $this->begin;
146         $per = sprintf( format: '%.3f',  args: $this->count / $interval);
147         $elapsed = sprintf( format: '%.6f', $interval);
148         $this->resultInsert->execute([$caller, (int)$this->count, $elapsed, $per]);
149     }
```

Model SaveMany

Framework Model saves all rows (10% performance gain, 1.1X)

```
151     private function modelSaveMany() {
152         $this->beginRun();
153
154         $table = TableRegistry::get( alias: 'Benchmark');
155         $entities = $table->newEntities($this->import);
156         $table->saveMany($entities);
157         $this->flushTables();
158
159         $this->endRun( caller: __FUNCTION__);
160     }
```

Model Bulk Insert

Framework Model's bulk-insert ability (10X performance gain)

```
162     private function bulkInsertSingle() {
163         $this->beginRun();
164
165         $table = TableRegistry::get( alias: 'Benchmark');
166         $query = $table->query();
167         $query->insert(array_keys($this->import[0]));
168         foreach ($this->import as $item) {
169             $query->values($item);
170         }
171         $query->execute();
172         $this->flushTables();
173
174         $this->endRun( caller: __FUNCTION__);
175     }
```

Prepared Statement

PHP's PDO Prepared Statement
(12X performance gain)

```
177     private function preparedSingle() {
178         $this->beginRun();
179
180         $sql = 'INSERT INTO prepared_statements.benchmark
181           (motion, lat, lon, ele, time, nearest, distance, feet, seconds, mph, climb)
182           VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)';
183         $this->insert = $this->connection->prepare($sql);
184
185         foreach ($this->import as $item) {
186             $this->insert->execute(array_values($item));
187         }
188         $this->flushTables();
189
190         $this->endRun( caller: __FUNCTION__);
191     }
```

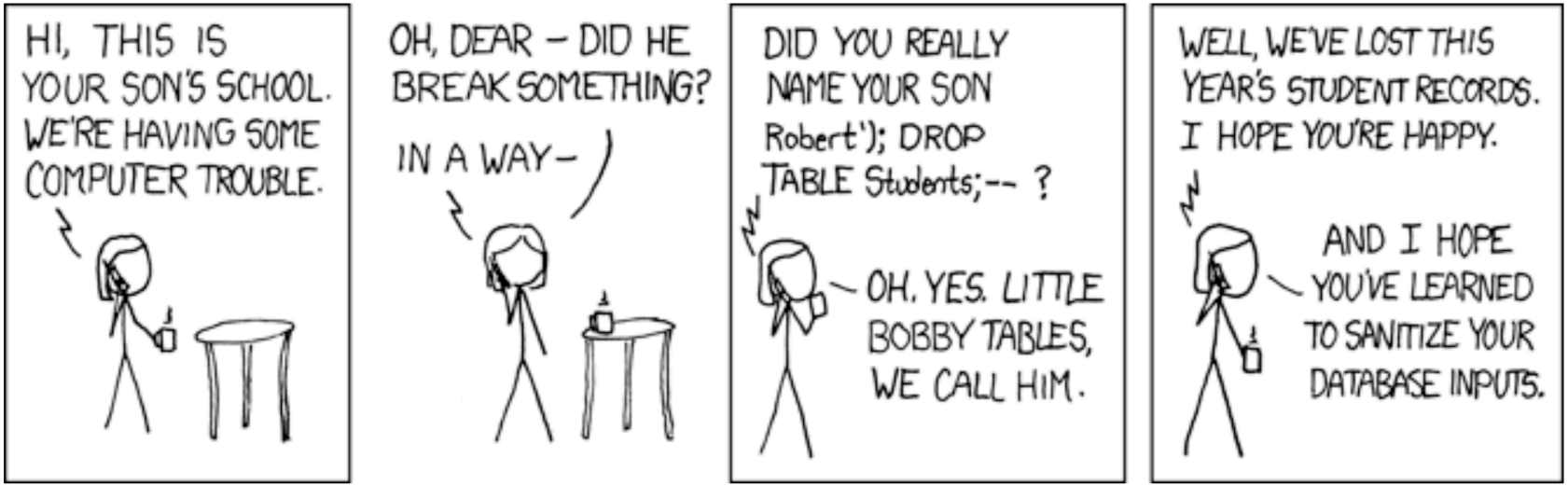
Sanitized Data Only!

```
193     private function preparedBulk() {
194         $this->beginRun();
195
196         $sql = 'INSERT INTO prepared_statements.benchmark
197           (motion, lat, lon, ele, time, nearest, distance, feet, seconds, mph, climb)
198           VALUES ';
199         $rows = [];
200         foreach ($this->import as $item) {
201             $row = [];
202             foreach ($item as $value) {
203                 $row[] = "'$value'";
204             }
205             $rows[] = '(' . implode( glue: ',', $row) . ')';
206         }
207         $sql .= implode( glue: ',', $rows);
208         $this->connection->query($sql);
209         $this->flushTables();
210
211         $this->endRun( caller: __FUNCTION__);
212     }
213 }
```

Hand-Coded Bulk Insert
(92X performance gain)

Exploits of a Mom (XKCD Comic)

https://xkcd.com/327/



Her daughter is named Help I'm trapped in a driver's license factory.

The Plan

- Benchmark
- **Prepared Statements**
- Table Design
- Prepared-Statement Utility

What is a “Prepared Statement”?

- Database analyzes, compiles, optimizes its plan for executing the query
- Complex process is relatively time consuming
- Compile once, execute many
- Parameters need not be quoted; not susceptible to SQL injection

From the PHP Manual

http://php.net/manual/en/pdo.prepared-statements.php

```php
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (:name, :value)");
$stmt->bindParam(':name', $name);
$stmt->bindParam(':value', $value);

// insert one row
$name = 'one';
$value = 1;
$stmt->execute();

// insert another row with different values
$name = 'two';
$value = 2;
$stmt->execute();
```

Example #1: Using named placeholders

From the PHP Manual

http://php.net/manual/en/pdo.prepared-statements.php

```php
$stmt = $dbh->prepare("INSERT INTO REGISTRY (name, value) VALUES (?, ?)");
$stmt->bindParam(1, $name);
$stmt->bindParam(2, $value);

// insert one row
$name = 'one';
$value = 1;
$stmt->execute();

// insert another row with different values
$name = 'two';
$value = 2;
$stmt->execute();
```

Example #2: Using positional ? placeholders

From the PHP Manual

http://php.net/manual/en/pdo.prepared-statements.php

```
$stmt = $dbh->prepare("SELECT * FROM REGISTRY where name = ?");
if ($stmt->execute(array($_GET['name']))) {
  while ($row = $stmt->fetch()) {
    print_r($row);
  }
}
```

Example #3: Fetching results row by row

Parameters are passed to the database server separately from the query statement, making SQL injection impossible

When and Why I Use Prepared Statements

- Performance: Benchmark shows 10X improvement on inserts
- When I already hand-coded queries during table design
- I use the CakePHP wrappers (as shown in this project)
- **Warning: Hand-coded queries where you place your data in the query string (like our 92X benchmark) are subject to SQL Injection**
- Standard web page load is not a use case; batch processing is

Web Page Load

- Bulk insert: Not applicable
- Not a long-running process
- Does not do repetitive inserts during SAME page load
- So… Use your standard framework ORM tools

Long-Running Process 1

- Batch job
- Processing a queue
- Long-running (micro)service, worker, agent

Long-Running Process 2

- Compile once, execute many
- Efficiency means lower impact on production database
- Can incorporate caching strategies

Bulk Insert

- Very-large inserts can block other production database queries
- Only consider fully-sanitized data
- For data import consider MySQL's LOAD DATA INFILE

The Plan

- Benchmark
- Prepared Statements
- **Table Design**
- Prepared-Statement Utility

First Table Design

```
CREATE TABLE `first_import` (
  `id`       MEDIUMINT(8) UNSIGNED   NOT NULL AUTO_INCREMENT,
  `motion`   VARCHAR(255)            NOT NULL DEFAULT '',
  `lat`      DECIMAL(10, 6)          NOT NULL,
  `lon`      DECIMAL(10, 6)          NOT NULL DEFAULT '0.000000',
  `ele`      DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT '0.00',
  `time`     DATETIME                NOT NULL,
  `nearest`  VARCHAR(255)            NOT NULL DEFAULT '',
  `distance` MEDIUMINT(8) UNSIGNED   NOT NULL DEFAULT '0',
  `feet`     MEDIUMINT(8) UNSIGNED   NOT NULL DEFAULT '0',
  `seconds`  MEDIUMINT(8) UNSIGNED   NOT NULL DEFAULT '0',
  `mph`      DECIMAL(10, 2)          NOT NULL DEFAULT '0.00',
  `climb`    DECIMAL(10, 2)          NOT NULL DEFAULT '0.00',
  PRIMARY KEY (`id`),
  UNIQUE KEY `time` (`time`)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = utf8;
```

Identical to CSV data file with primary key added as `id`

Unique key for de-duplicating rows based on their timestamp

First Data Import

- Hike, Walk, Drive values repeated many times
- Waypoint values also repeated many times
- Report generated by query available in source code on GitHub

time bin/cake first_import import --verbose

```
Loaded export (CSV) file

Motion Counts:
Hike        1164
Walk        2009
Drive       6465

Nearest-Waypoint Counts:
Kautz Creek Trail                                   48
…
Grove of the Patriarchs Suspension Bridge           164
…
Kane Hall                                           211
Westside Road End                                   219
Gateway Inn                                         220
Twin Firs                                           317
Kautz Creek                                         345
Gateway Inn Cabins                                  407
Reflection Lakes Jct 1                              460
We Work Meetup                                      523
Reflection Lakes Parking                            579
Longmire Museum                                     655
Ohanapecosh Campground Picnic                       740
Paradise Visitor Center                             912
University Inn Pineapple                            971
Rainier Mountaineering Inc                          1015
```

Data Import Table

“ele” is elevation above Sea Level in meters (1620 meters === 5315 feet)

id	motion	lat	lon	ele	time	nearest	distance	feet	seconds	mph	climb
1957	Hike	46.776467	-121.730934	1618.33	2017-09-04 20:31:22	Reflection Lakes Jct 1	2715	35	16	1.49	1.20
1958	Hike	46.776521	-121.730816	1619.51	2017-09-04 20:31:40	Reflection Lakes Jct 1	2688	35	18	1.34	1.18
1959	Hike	46.776560	-121.730754	1620.32	2017-09-04 20:31:56	Reflection Lakes Jct 1	2674	21	16	0.90	0.81
1960	Hike	46.776632	-121.730671	1619.76	2017-09-04 20:32:14	Reflection Lakes Jct 1	2657	33	18	1.27	-0.56
1961	Hike	46.776728	-121.730603	1621.15	2017-09-04 20:32:32	Reflection Lakes Jct 1	2646	39	18	1.47	1.39
1962	Hike	46.776796	-121.730475	1620.13	2017-09-04 20:32:49	Reflection Lakes Jct 1	2618	40	17	1.62	-1.02
1963	Hike	46.776872	-121.730356	1621.11	2017-09-04 20:33:06	Reflection Lakes Jct 1	2594	41	17	1.63	0.98
1964	Hike	46.776933	-121.730277	1622.18	2017-09-04 20:33:22	Reflection Lakes Jct 1	2578	30	16	1.27	1.07
1965	Hike	46.777038	-121.730152	1622.96	2017-09-04 20:33:41	Reflection Lakes Jct 1	2555	49	19	1.77	0.78
1966	Hike	46.777099	-121.730008	1623.53	2017-09-04 20:33:59	Reflection Lakes Jct 1	2525	42	18	1.60	0.57
1967	Hike	46.777146	-121.729863	1623.49	2017-09-04 20:34:15	Reflection Lakes Jct 1	2493	40	16	1.71	-0.04
1968	Hike	46.777229	-121.729795	1624.79	2017-09-04 20:34:35	Reflection Lakes Jct 1	2484	35	20	1.18	1.30
1969	Hike	46.777303	-121.729701	1625.66	2017-09-04 20:34:51	Reflection Lakes Jct 1	2468	36	16	1.52	0.87
1970	Hike	46.777363	-121.729614	1625.82	2017-09-04 20:35:09	Reflection Lakes Jct 1	2452	31	18	1.17	0.16

First Observation

- Motion values repeated many times
- Move to separate small table
- Main table points to row ID of small table

```
Motion Counts:
Hike          1164
Walk          2009
Drive         6465
```

Second Observation

- Nearest-Waypoint repeated many times as well
- Move to separate small table
- Main table points to row ID of small table

```
Nearest-Waypoint Counts:
Kautz Creek Trail                                  48
Grove of the Patriarchs Suspension Bridge          164
Kane Hall                                          211
Westside Road End                                  219
Gateway Inn                                        220
Twin Firs                                          317
Kautz Creek                                        345
```

Third Observation

- These small tables have similar characteristics
- Recording a name (string)
- Very few rows relative to main table
- We can write a small utility handling all such similar small tables

Final Table Design 1

```sql
CREATE TABLE `motion` (
  `id`   TINYINT(3) UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255)        NOT NULL DEFAULT '',
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = utf8;
```

```sql
CREATE TABLE `waypoint` (
  `id`   SMALLINT(5) UNSIGNED NOT NULL AUTO_INCREMENT,
  `name` VARCHAR(255)         NOT NULL DEFAULT '',
  PRIMARY KEY (`id`),
  UNIQUE KEY `name` (`name`)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = utf8;
```

Final Table Design 2

```
CREATE TABLE `second_import` (
  `id`          SMALLINT(8) UNSIGNED    NOT NULL AUTO_INCREMENT,
  `motion_id`   TINYINT(3) UNSIGNED     NOT NULL DEFAULT '0',
  `waypoint_id` SMALLINT(5) UNSIGNED    NOT NULL DEFAULT '0',
  `lat`         DECIMAL(10, 6)          NOT NULL,
  `lon`         DECIMAL(10, 6)          NOT NULL DEFAULT '0.000000',
  `ele`         DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT '0.00',
  `time`        DATETIME                NOT NULL,
  `distance`    MEDIUMINT(8) UNSIGNED   NOT NULL DEFAULT '0',
  `feet`        MEDIUMINT(8) UNSIGNED   NOT NULL DEFAULT '0',
  `seconds`     MEDIUMINT(8) UNSIGNED   NOT NULL DEFAULT '0',
  `mph`         DECIMAL(10, 2)          NOT NULL DEFAULT '0.00',
  `climb`       DECIMAL(10, 2)          NOT NULL DEFAULT '0.00',
  PRIMARY KEY (`id`),
  UNIQUE KEY `time` (`time`)
)
  ENGINE = InnoDB
  DEFAULT CHARSET = utf8;
```

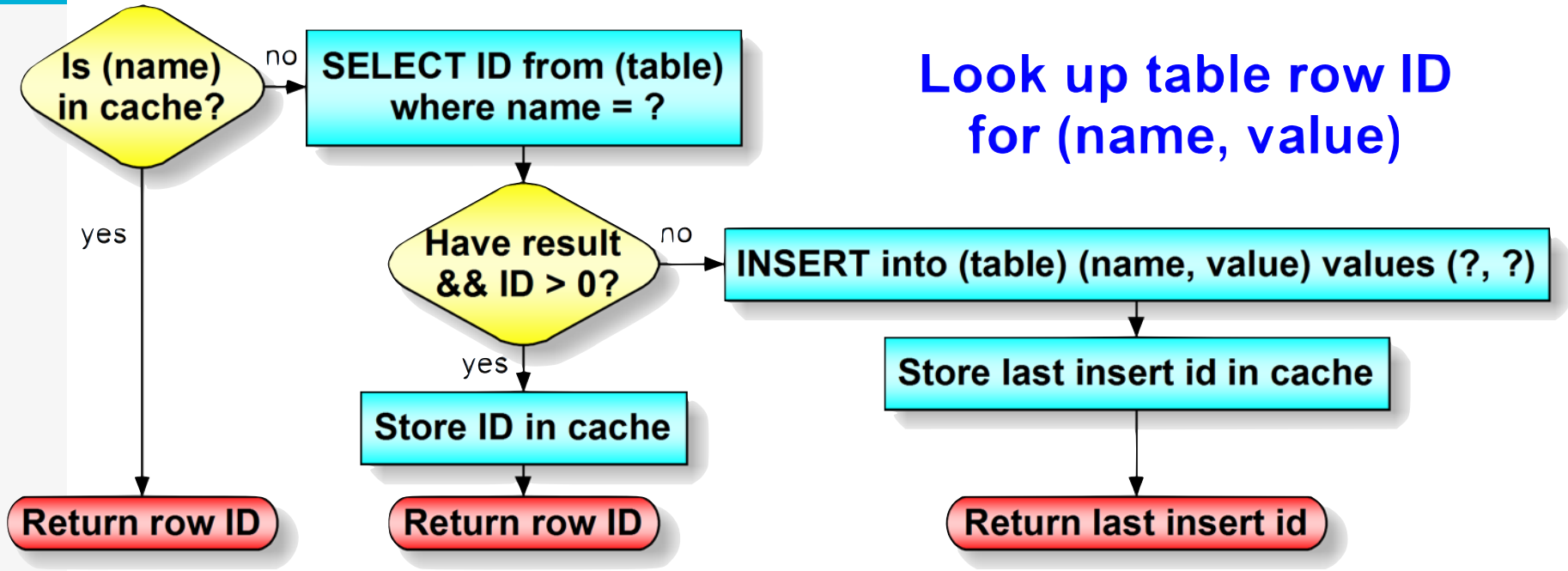
The Plan

- Benchmark
- Prepared Statements
- Table Design
- **Prepared-Statement Utility**

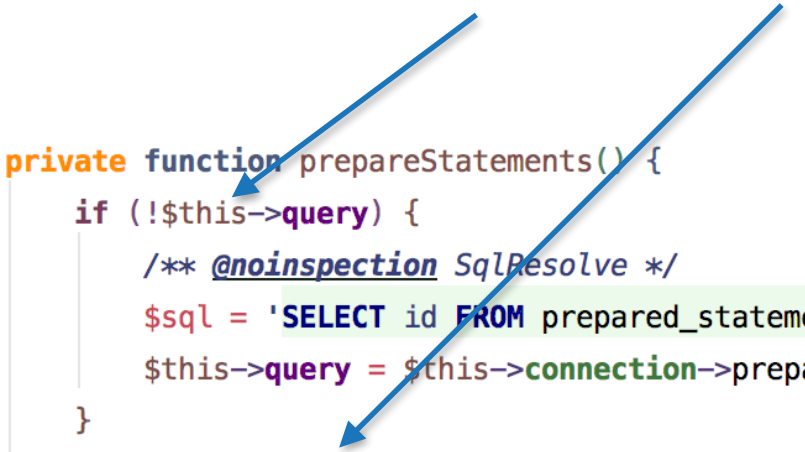
Code Design

- Internal cache (memoize)
- Small tables have identical structure
- Single utility class handling all such tables
- Utility implemented as one Singleton for each table

Lookup Utility Flow



LookupUtil 1



```
59      private function prepareStatements() {
60          if (!$this->query) {
61              /** @noinspection SqlResolve */
62              $sql = 'SELECT id FROM prepared_statements.' . $this->table . ' WHERE `name` = ?';
63              $this->query = $this->connection->prepare($sql);
64          }
65          if (!$this->insert) {
66              /** @noinspection SqlResolve */
67              $sql = 'INSERT INTO prepared_statements.' . $this->table . ' (`name`) VALUES (?)';
68              $this->insert = $this->connection->prepare($sql);
69          }
70      }
```

LookupUtil 2

```
72      public static function lookup(Connection $connection, $table, $value) {
73          $instance = static::getInstance($connection, $table);
74          return array_key_exists($value, $instance->cache) ?
75              $instance->cache[$value] : $instance->runLookup($value);
76      }
77
78      /**
79       * @param Connection $connection
80       * @param string $table
81       * @param array $dependencies
82       * @return LookupUtil
83       */
84      public static function getInstance(Connection $connection, $table, array $dependencies = []) {
85          if (!array_key_exists($table, static::$instances)) {
86              static::$instances[$table] = new static($connection, $table, $dependencies);
87          }
88          return static::$instances[$table];
89      }
```

LookupUtil 3

```
91      private function runLookup($value) {
92          if (count($this->cache) >= static::$cacheLimit) {
93              $this->cache = []; // Cache got too big; clear and start over
94          }
95          if (!$this->query) {
96              // Should only happen when developing unit tests
97              throw new \InvalidArgumentException('No query for ' . $this->table);
98          }
99          $parms = [substr($value, start: 0, length: 255)];
100         $this->query->execute($parms);
101         $row = $this->query->fetch( type: 'assoc');
102         if (is_array($row) && array_key_exists( key: 'id', $row)) {
103             $id = (int)$row['id'];
104         } else {
105             $this->insert->execute($parms);
106             $id = (int)$this->insert->lastInsertId();
107         }
108         $this->cache[$value] = $id;
109         return $id;
110     }
```

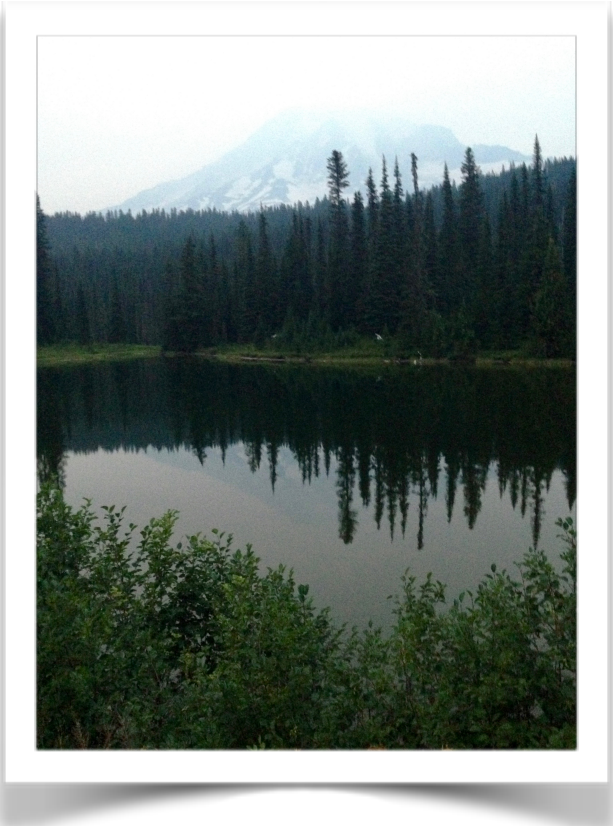
Second Import 1 of 2

```
48        $sql = 'INSERT INTO prepared_statements.second_import
49          (motion_id, waypoint_id, lat, lon, ele, time, distance, feet, seconds, mph, climb)
50          VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)';
51        $this->insert = $this->connection->prepare($sql);
52
53        $sql = 'SELECT id FROM prepared_statements.second_import WHERE time = ? LIMIT 1';
54        $this->queryTime = $this->connection->prepare($sql);
55
56        $sql = 'SELECT m.`name` motion, count(m.`name`) count
57          FROM prepared_statements.second_import si
58          INNER JOIN prepared_statements.motion m ON m.id = si.motion_id
59          GROUP BY m.`name` ORDER BY count(m.`name`)
60          LIMIT 10';
61        $this->queryMotion = $this->connection->prepare($sql);
62
63        $sql = 'SELECT w.`name` nearest, count(w.`name`) count
64          FROM prepared_statements.second_import si
65          INNER JOIN prepared_statements.waypoint w
66          ON w.id = si.waypoint_id
67          GROUP BY w.`name` ORDER BY count(w.`name`)
68          LIMIT 100';
69        $this->queryNearest = $this->connection->prepare($sql);
```

Second Import 2 of 2

```
88      private function populateTable() {
89          foreach ($this->import as $item) {
90              $this->queryTime->execute([$item['time']]);
91              $row = $this->queryTime->fetch( type: 'assoc');
92              if (!(is_array($row) && array_key_exists( key: 'id', $row))) {
93                  $this->populateImport($item);
94              }
95          }
96      }
97
98      private function populateImport(array $import) {
99          $motionId = LookupUtil::lookup($this->connection, table: 'motion', $import['motion']);
100         $waypointId = LookupUtil::lookup($this->connection, table: 'waypoint', $import['nearest']);
101         $parms = [
102             $motionId, $waypointId, $import['lat'], $import['lon'], $import['ele'], $import['time'],
103             $import['distance'], $import['feet'], $import['seconds'], $import['mph'], $import['climb'],
104         ];
105         $this->insert->execute($parms);
106     }
```

Same Result



time bin/cake second_import second --verbose

```
Loaded export (CSV) file

Motion Counts:
Hike        1164
Walk        2009
Drive       6465

Nearest-Waypoint Counts:
Kautz Creek Trail                                48
Grove of the Patriarchs 024                      79
Grove of the Patriarchs 025                      87
Grove of the Patriarchs 023                      125
Johnson Hall                                     147
Grove of the Patriarchs Suspension Bridge        164
Stevens Way Entrance                             173
Reflection Lakes Jct 2                           210
Kane Hall                                        211
Westside Road End                                219
Gateway Inn                                      220
Twin Firs                                        317
Kautz Creek                                      345
Gateway Inn Cabins                               407
Reflection Lakes Jct 1                           460
We Work Meetup                                   523
Reflection Lakes Parking                         579
Longmire Museum                                  655
Ohanapecosh Campground Picnic                    740
Paradise Visitor Center                          912
University Inn Pineapple                         971
Rainier Mountaineering Inc                       1015
Marshall House                                   1031

real 0m12.403s
user 0m5.163s
sys  0m1.792s
```

Summary

- Specific use case: Many queries/ inserts in one invocation
- Async/batch job, processing a queue, etc.
- Take advantage of database tables having repeating information
- Main PHP application: Take advantage of framework tools
- Highly data intensive: Use best tools for *this* job