

Re: [PHP] Arbitrary mathematical relations, not just hashes

Source: <http://coding.derkeiler.com/Archive/PHP/php.general/2008-04/msg00253.html>

- *From:* topnotcher@xxxxxxxxxx ("Greg Bowser")
 - *Date:* Sun, 6 Apr 2008 21:23:28 -0400
-

(sorry I just hit send on a blank email; I'm absent-minded)

First, in the strictest mathematical sense, a relation from a set A to a set B is a subset of the cross-product $A \times B$.

(obviously, the mathematical notation is not a great way to represent this in a program.)

Hence, a relation is a set of ordered pairs: `array(array('apple','red'), array('ruby','red'))` is a relation from the objects to the colors.

For a relation R from A to B , the inverse relation is defined by (b,a) for all a in A and B in b .

Thus, the inverse relation of objects \sim colors would be:

```
array(array('red','apple'), array('red','ruby')).
```

A function is a special case of a relation: functions have the property that if a in A is related to b in B by the relation R and a is related to c by the relation R , then $b = c$.

It follows that mathematically, there is no function $f: \text{color} \rightarrow \text{object}$.

As you suggest, each element of the relation $\text{colors} \sim \text{objects}$ relation must return a list in order to be a function.

So... How does SQL solve this problem?

SQL has indexes. You have a list of objects:

```
<?
$objects = array(
array(
'name'=>'apple',
'type' => 'fruit',
'color' => 'red'
),
```

Re: [PHP] Arbitrary mathematical relations, not just hashes

```
array(  
'name' => 'ruby',  
'type' => 'gem',  
'color' => 'red'  
)  
);  
>
```

Given a type, you would like to find all objects with that type. One way to accomplish this might be:

```
<?php  
function locate($property,$value,$objects) {  
    $return = array();  
    foreach ($objects as $object) {  
        if ($object[$property] == $value)  
            $return[] = $object;  
    }  
    return $return;  
}  
>
```

Obviously, the time increased to complete a search will increase with the size of the \$objects array. This is where the indexes come in: Suppose we would like to easily be able to locate an object given a color, a type, or its name:

```
<?php  
//array, indexed by $property. Each element is an array of numbers, n,  
//corresponding to $objects[n].  
$indexes = array();  
  
//properties to index  
$keys = array('color','type','name');  
  
//initialize  
foreach ($keys as $key)  
    $indexes[$key] = array();  
  
//this accomplishes the same thing as array_flip() ;  
foreach ($objects as $n => $object) foreach ($keys as $key) {  
    if (!isset($indexes[$key][$object[$key]]))  
        $indexes[$key][$object[$key]] = array();  
  
    $indexes[$key][$object[$key]][] = $n;  
}  
>
```

The above produces the following array:

```
Array  
(
```

Re: [PHP] Arbitrary mathematical relations, not just hashes

```
[color] => Array
(
  [red] => Array
  (
    [0] => 0
    [1] => 1
  )
)
```

```
[type] => Array
(
  [fruit] => Array
  (
    [0] => 0
  )
)
```

```
[gem] => Array
(
  [0] => 1
)
)
```

```
[name] => Array
(
  [apple] => Array
  (
    [0] => 0
  )
)
```

```
[ruby] => Array
(
  [0] => 1
)
)
)
```

Conceptually, this is all SQL does.

Anyway... I wouldn't actually do any of that in an application. I'd probably go with what Casey suggested.

— GREG

Disclaimer: The above is intended only to be conceptual. Any attempt at parsing may fail. Technical accuracy is not guaranteed.

P.S. Sorry for the math. I was bored :p

Re: [PHP] Arbitrary mathematical relations, not just hashes

Re: [PHP] Arbitrary mathematical relations, not just hashes

However, I can't easily find all items whose \$color is 'red', nor all items whose \$type is 'fruit'. In other words, color() and type() aren't full mathematical relations.

Of course, I could create the inverse function as I go along:

```
$inverse_color['red'] = ['apple', 'ruby']; # uglyish, assigning list to
```

value

By definition, not all relations are functions, nor all functions invertible.

The function inverse_color(object) specifies more than

What you're referring to sounds like an index: given an object, you can easily find it. On Sun, Apr 6, 2008 at 8:15 PM, Casey <heavyccasey@xxxxxxxx> wrote:

On Sun, Apr 6, 2008 at 4:52 PM, Kelly Jones <kelly.terry.jones@xxxxxxxx> wrote:

Many programming languages (including Perl, Ruby, and PHP) support

hashes:

```
$color['apple'] = 'red';  
$color['ruby'] = 'red';  
  
$type['apple'] = 'fruit';  
$type['ruby'] = 'gem';
```

This quickly lets me find the color or type of a given item.

In this sense, color() and type() are like mathematical functions.

However, I can't easily find all items whose \$color is 'red', nor all items whose \$type is 'fruit'. In other words, color() and type() aren't full mathematical relations.

Of course, I could create the inverse function as I go along:

```
$inverse_color['red'] = ['apple', 'ruby']; # uglyish, assigning list to
```

value

and there are many other ways to do this, but they all seem kludgy.

Re: [PHP] Arbitrary mathematical relations, not just hashes

Is there a clean way to add 'relation' support to Perl, Ruby, or PHP?

Is there a language that handles mathematical relations

naturally/natively?

I realize SQL does all this and more, but that seems like overkill for something this simple?

—
We're just a Bunch Of Regular Guys, a collective group that's trying to understand and assimilate technology. We feel that resistance to new ideas and technology is unwise and ultimately futile.

Something like this?

```
<?php
$objects = array(
  'apple' => array(
    'type' => 'fruit',
    'color' => 'red'
  ),
  'ruby' => array(
    'type' => 'gem',
    'color' => 'red'
  )
);

// Search for all red objects.
$red = array();
foreach ($objects as $name => $object) {
  if ($object['type'] == 'red')
    $red[] = $name;
}

?>
```

—
-Casey

—
PHP General Mailing List (<http://www.php.net/>)
To unsubscribe, visit: <http://www.php.net/ unsub.php>