

Tetraboy asks: “Hey Maggie,
do you have any bears?”

No, but here's a slow loris!



ORM in the PHP World

Maggie Nelson
php|tek 2009

About Me

<3 databases

<3 PHP

</3 how PHP and database play together



What is this “ORM” you speak of?

Objects

- Represent things in real life
- Have properties
- Have methods that access properties in some way
- Inherit from each other



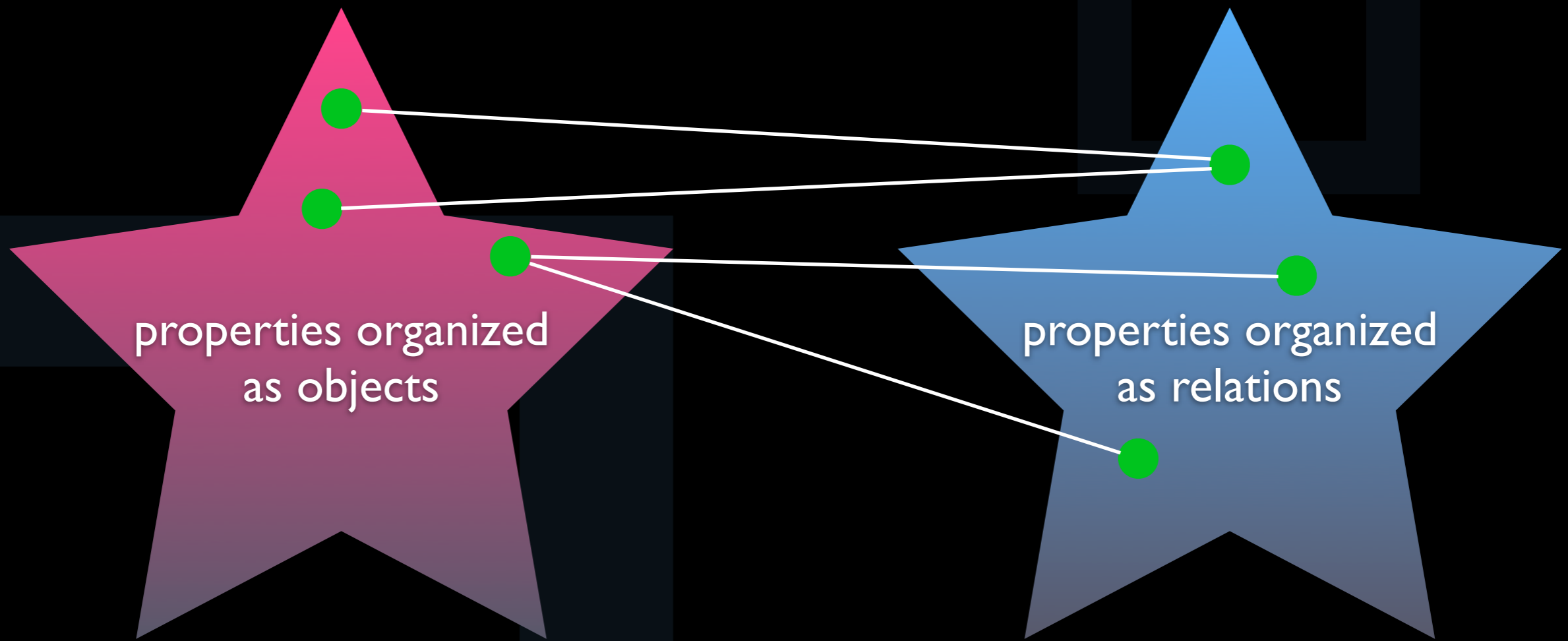
Relations

- Represent how things in real life connect to each other
- Also represent how properties of things connect to each other



properties organized
as relations

Map



Encapsulate this!



Encapsulation

- Hide the pesky details (“how does the car work again?”)
- Provide a common interface (“here’s the steering wheel, here’s the gas pedal”)

Objects

Public methods and properties provide a natural interface.

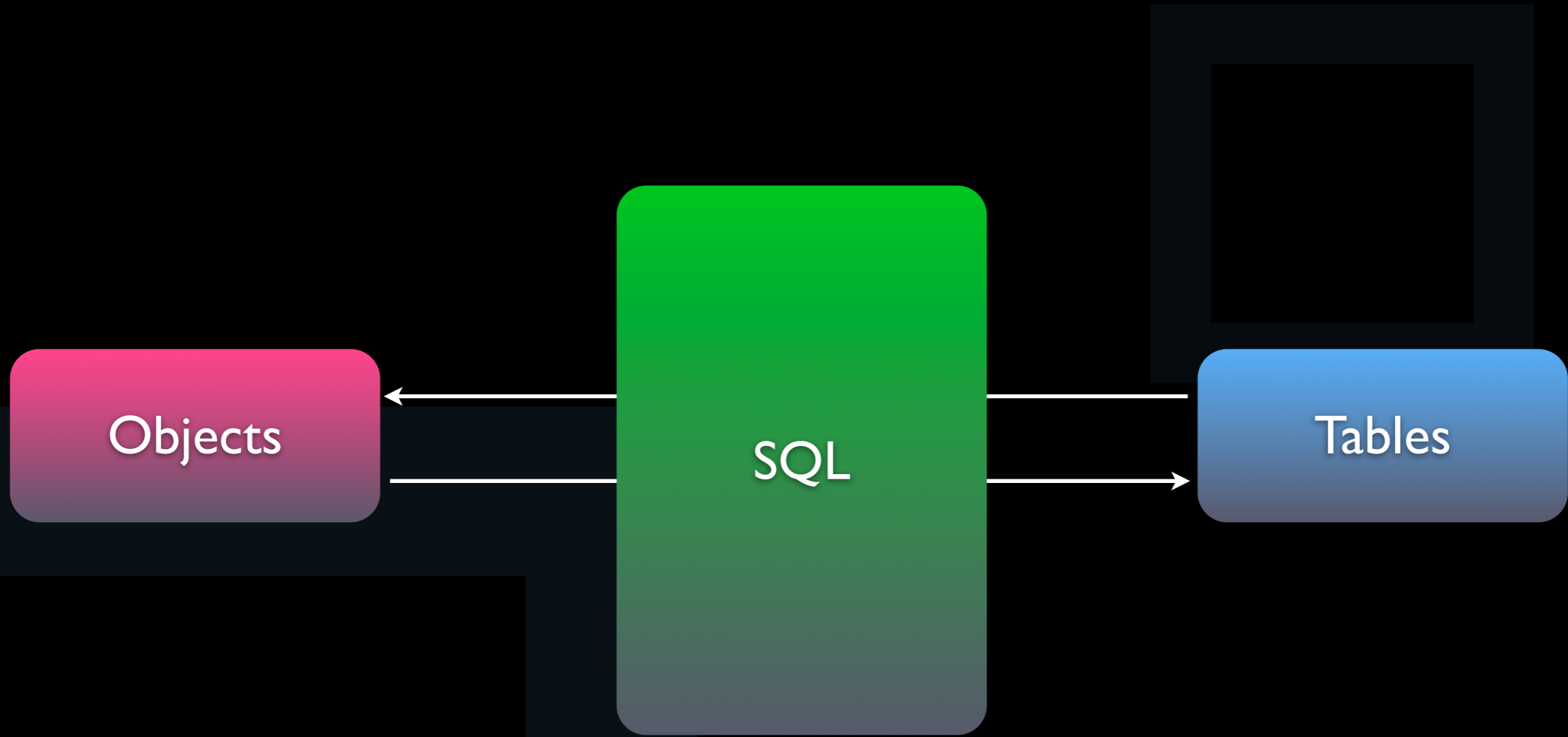
Restricted methods and properties hide away the details.

SQL is a type of interface

SQL as an interface

SQL provides public access to stored data and way to combine the data.

Database internals are completely hidden away.

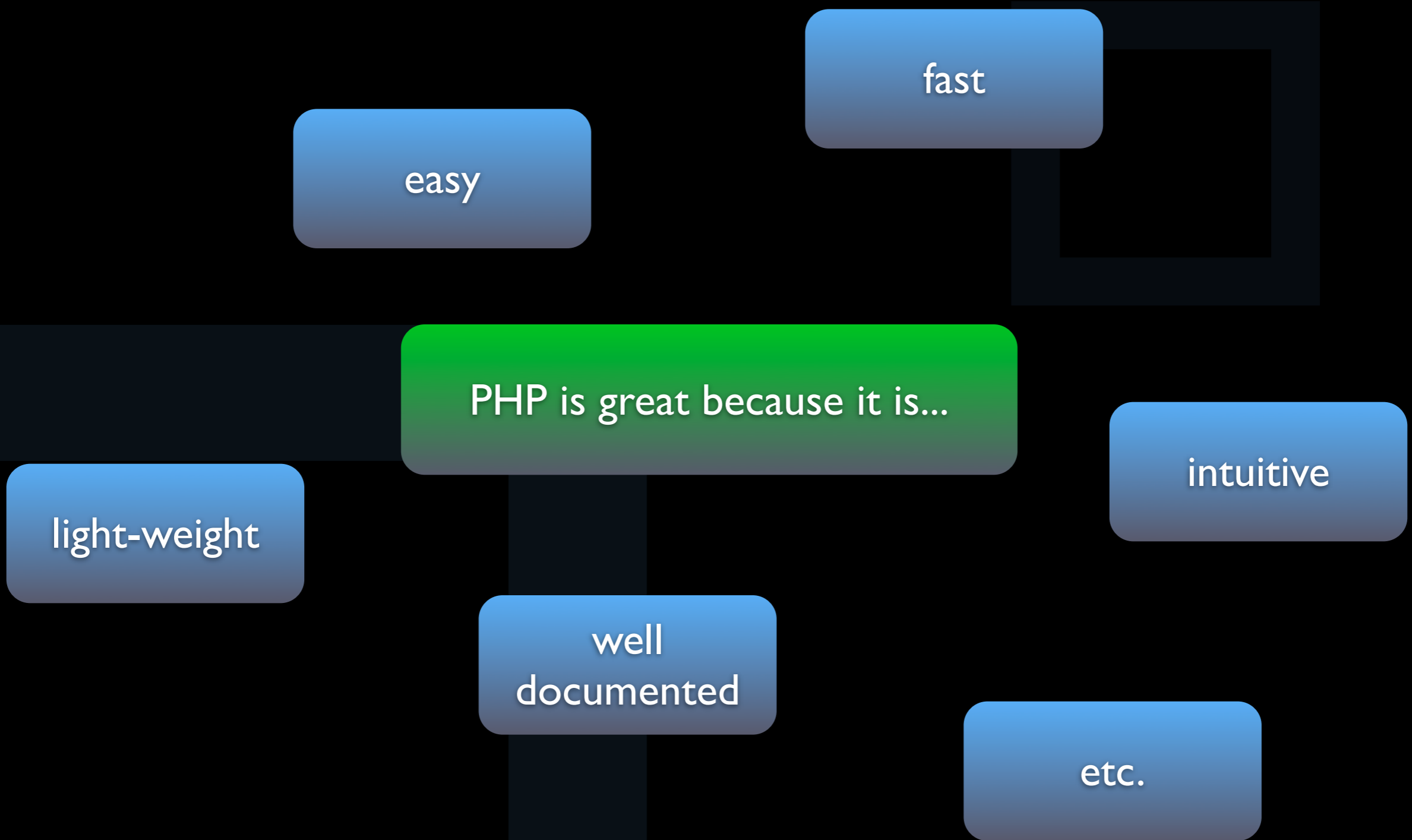


Oh, man, but SQL seems pretty detailed! We should totally hide it!

I agree, more encapsulation is needed, but...

Let's talk about PHP!





Objects aren't arrays

ORM (Object Relational Mapping)

not ARM (Array Relational Mapping?)

“The Database Class” != ORM

“The Database Class” and the persistence objects should not live in the same spot

The “Database Class”

Deals with database access.

“Hey, MySQL, execute this query, please, here are my credentials.”

ORM

Persists application data.

```
$kitten->save();
```



PHP is already a language

I hope this ORM won't make me learn all new syntax and stuff!

Let's get some ORM on!

In the ideal world...

Object (can) persist.

Object don't know HOW and WHERE they persist.

CRUD!



Persisting objects via CRUD

C – create (add)

R – read (select)

U – update (modify)

D – delete (remove)

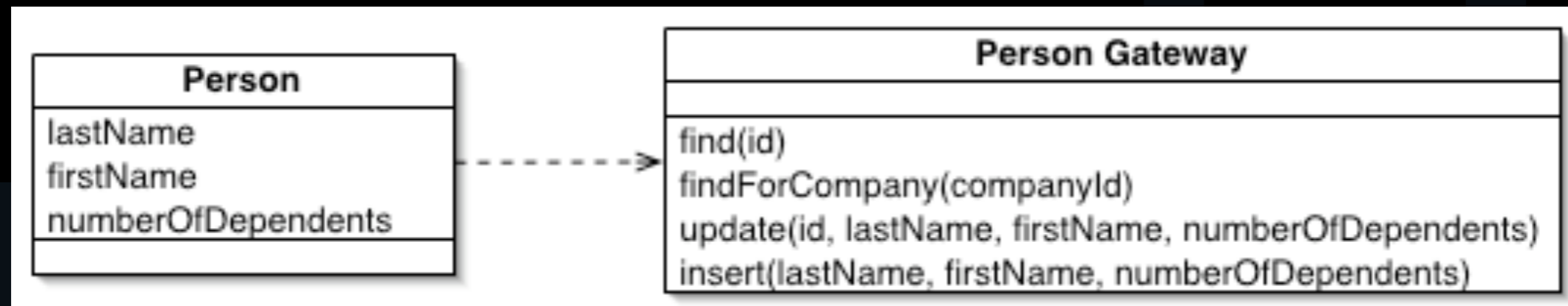


ORM Patterns



Table Data Gateway

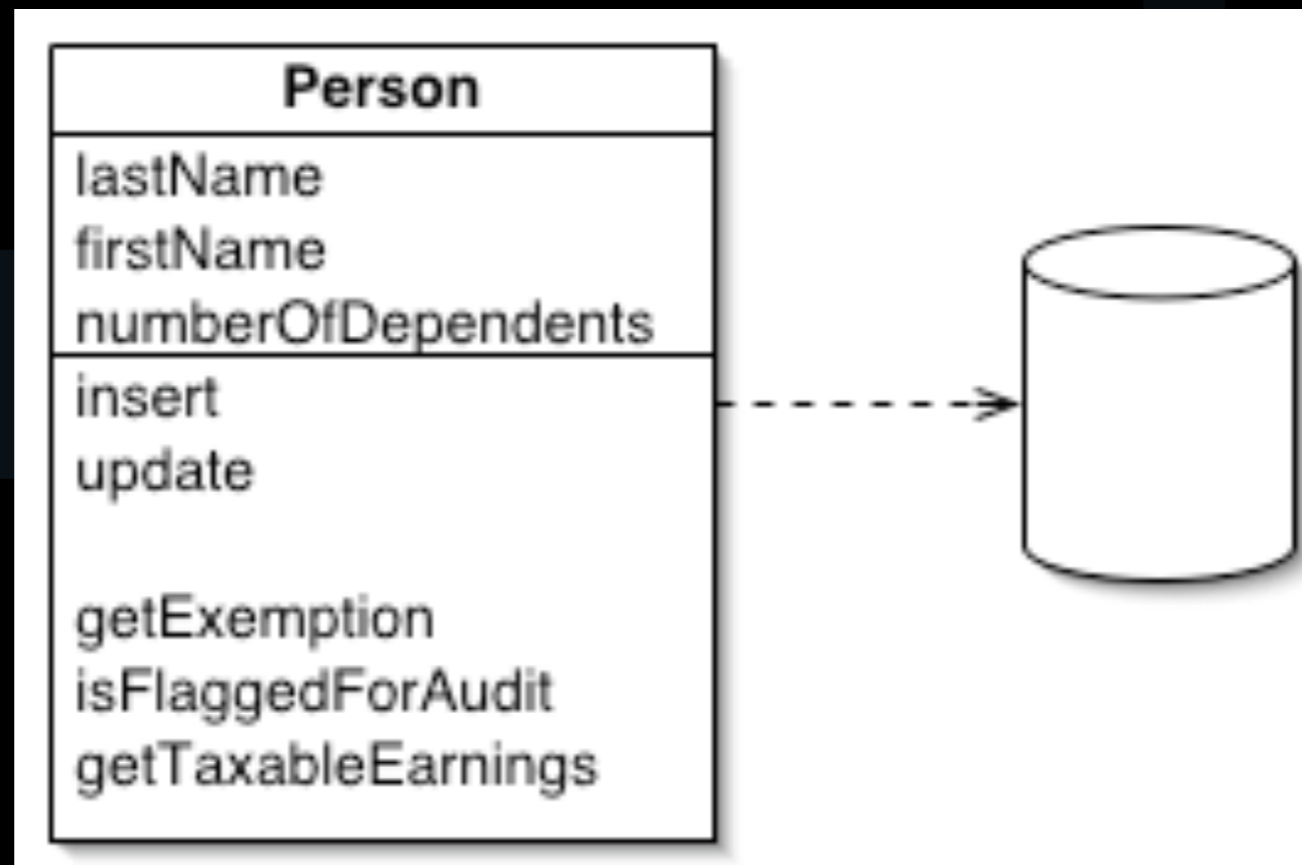
usually paired with
Row Data Gateway



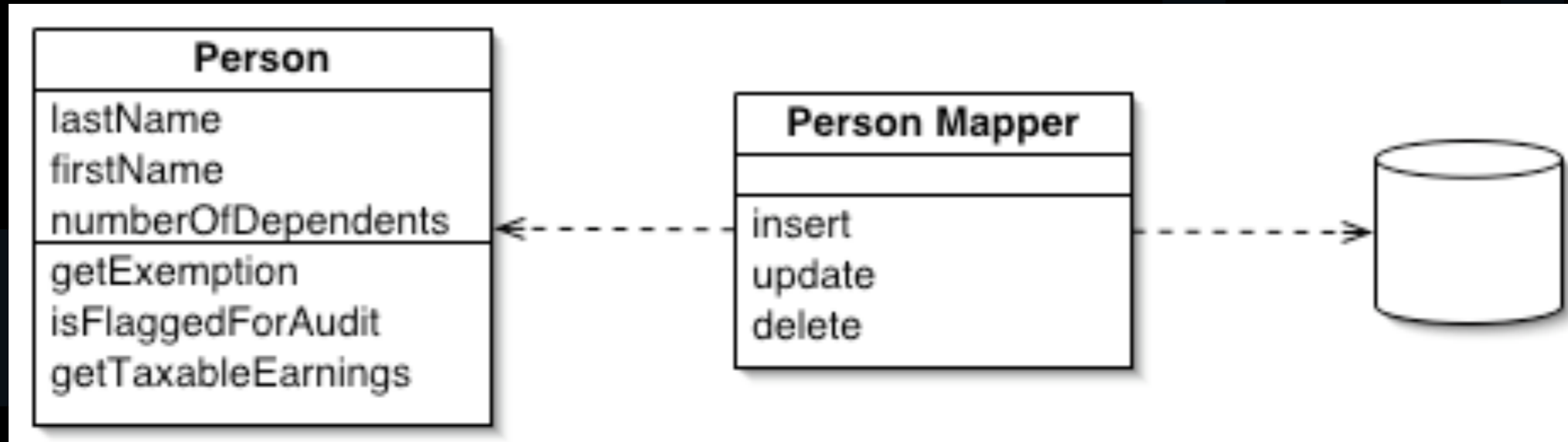
An object that acts as a gateway to a single row / table.

There is one instance per row / table.

Active Record (think Ruby on Rails!)



Data Mapper (my fave!)



(Thanks to Martin Fowler
for the delicious pattern
diagrams!)

Generating SQL



what does this mean?

```
$select = $db->select()  
    ->from(array('p' => 'products'),  
         array('product_id'))  
    ->join(array('l' => 'line_items'),  
         'p.product_id = l.product_id',  
         array('line_items_per_product' => 'COUNT(*)'))  
    ->group('p.product_id');
```

this may be clearer...

```
SELECT p."product_id", COUNT(*) AS line_items_per_product
FROM "products" AS p JOIN "line_items" AS l
  ON p.product_id = l.product_id
GROUP BY p.product_id;
```

but this is nicer (skoop agrees)

```
select p.product_id,  
       count(*) line_items_per_product  
from products p,  
     line_ites l  
where p.product_id = l.product_id  
group by p.product_id;
```

compare...

```
$select = $db->select()  
->from(array('p' => 'products'),  
       array('product_id'))  
->join(array('l' => 'line_items'),  
       'p.product_id = l.product_id',  
       array('line_items_per_product' => 'COUNT(*)'))  
->group('p.product_id');
```

```
select p.product_id,  
       count(*) line_items_per_product  
from products p,  
     line_ites l  
where p.product_id = l.product_id  
group by p.product_id;
```

Benefits of ORM

Gets rid of repetitive code.

Avoids having to set up database objects “by hand”.

Easy to understand for new developers.

Portability*.

Downsides of ORM

Gets really convoluted when retrieving complicated relationships.

Takes away the power of optimizing database access.

May be inefficient.

Strong MySQL preference in the PHP world.

Ok, so what's out there?

Propel

“It allows you to access your database using a set of objects, providing a simple API for storing and retrieving data.”

Provides then M in MVC.

Propel uses the Creole database abstraction layer.

“ORM tool but also a code generator.”

Row Data Gateway + Table Data Gateway
(NOT Active Record!)

Propel

Object classes for rows

Peer classes for table operations

Build database in XML

Criterion object for more complex queries.



Criterion object

```
$c = new Criteria();  
$c->add(SomePeer::ID = 10);  
  
$results = SomePeer::doSelect($c);
```

Doctrine

proprietary object oriented SQL dialect called Doctrine Query Language (DQL), inspired by Hibernate's HQL.

DQL simple example

```
$things =  
    $this->getThing()  
    ->orderBy('created_at DESC')  
    ->limit(50)  
    ->execute();
```



DQL complicated example

```
static function sort2dBySubLength($arr, $key, $order){  
    foreach($arr as $sub) $sc[] = count($sub[$key]);  
    $order = $order == 'desc' ? SORT_DESC : SORT_ASC;  
    array_multisort($sc, SORT_NUMERIC, $order, $arr);  
    return $arr;  
}
```

dORM

by Olivier Lalonde

“It implements multiple software design patterns such as Unit of Work, Identity Map, Lazy Load, Foreign Key Mapping, Association Table Mapping, Metadata Mapping and Data Mapper.”

<http://getdorm.com/>

Easy to use and understand

```
$dorm->getClassName("id");  
$dorm->save($object);  
$dorm->delete($object);
```



DORM Example: XML Mapping

```
<dorm>
  <map database="mysql://user:pass@hostname/dbname">
    <!-- sample class -->
    <className table="tableName">

      <!-- sample scalar property -->
      <propertyName1 field="fieldName" setter="setterMethod" getter="getterMethod" />

      <!-- sample object property -->
      <propertyName2 fkey="fkeyName" class="objectClassName" setter="setterMethod"
getter="getterMethod" />

      <!-- sample object array property -->
      <propertyName3 pivot="pivotTableName" class="objectClassName"
key="arrayKeyFieldName" />

    </className>
  </map>
</dorm>
```

Zend Framework

Zend_Db component:

Non-ORM related parts:

- Database Adapters
- Query profiler

ORM parts:

- Query builder
- CRUD functions
- Table and Row OO patterns
- Zend_Db_Mapper (proposed)



ZF Example 1: Running custom queries

```
$stmt = $db->query(  
    'SELECT * FROM bugs WHERE reported_by = ? AND  
bug_status = ?',  
    array('goofy', 'FIXED')  
);
```

ZF Example 2: ORM as an SQL “Translator”

```
$data = array(  
    'updated_on' => '2007-03-23',  
    'bug_status' => 'FIXED'  
);
```

```
$n = $db->update('bugs', $data, 'bug_id = 2');
```

ZF Example 3: Zend_Db_Mapper

```
public function increaseNumberOfBeds($byCount)
{
    for($i = 0; $i < $byCount; $i++) {
        $bed = new Clinic_Bed();
        $bed->setStation($this);
        $this->beds->add($bed);
    }
    return $this->getBeds();
}
```

Xyster Framework

Based on ZF, but takes some and improves on a lot.

ORM based on a Data Mapper design pattern.

<http://xyster.devweblog.org/wiki/xyster/Orm>

CodeIgniter

ActiveRecord

Database class that manages data retrieval and parsing.

Nice and light – doesn't really impose anything on the developer.

CodeIgniter Example 1: Running custom queries

```
$this->db->query('YOUR QUERY HERE');
```

CodeIgniter Example 2: result_object()

```
$query = $this->db->query("YOUR QUERY");

if ($query->num_rows() > 0)
{
    foreach ($query->result() as $row)
    {
        echo $row->title;
        echo $row->name;
        echo $row->body;
    }
}
```

eZ Component

Provides Persistent Object which provides Persistent Mapping – you can customize how an object is represented in the database.

eZ Component example

```
1. <?php
2. $def = new ezcPersistentObjectDefinition();
3. $def->table = "persons";
4. $def->class = "Person";
5.
6. $def->idProperty = new ezcPersistentObjectIdProperty;
7. $def->idProperty->columnName = 'id';
8. $def->idProperty->propertyName = 'id';
9. $def->idProperty->generator = new
ezcPersistentGeneratorDefinition( 'ezcPersistentNativeGenerator' );
10.
11. $def->properties['name'] = new ezcPersistentObjectProperty;
12. $def->properties['name']->columnName = 'full_name';
13. $def->properties['name']->propertyName = 'name';
14. $def->properties['name']->propertyType = ezcPersistentObjectProperty::PHP_TYPE_STRING;
15.
16. $def->properties['age'] = new ezcPersistentObjectProperty;
17. $def->properties['age']->columnName = 'age';
18. $def->properties['age']->propertyName = 'age';
19. $def->properties['age']->propertyType = ezcPersistentObjectProperty::PHP_TYPE_INT;
20.
21. return $def;
22. ?>
```

symfony

Offers support for both Propel and Doctrine.

PHPLinq

SQL translator

<http://phplinq.codeplex.com/>



PHPLinq example

```
$result = from('$employee')->in($employees)
->where('$employee => strlen($employee->Name) == 4')
->orderBy('$employee => $employee->Name')
->thenByDescending('$employee => $employee->Age')
->select('new {
    "EmailAddress" => $employee->Email,
    "Domain" => substr($employee->Email,
strpos($employee->Email, "@") + 1)
}');
```

Solar Framework

Solar_Sql_Model (for dealing with models) –
uses Table_Data_Gateway
– returns Collection of Object

Solar_Sql (for SQL generation and handling
results)

http://solarphp.com/package/Solar_Sql_Model

Outlet PHP ORM

SQL translator

<http://www.outlet-orm.org>



Outlet PHP Orm Example: SQL + property mix

```
$outlet = Outlet::getInstance();

// select using prepared statement
$projects = $outlet->select(
    'Project',
    'WHERE StatusID = ? OR StatusID = ?',
    array(0, 1)
);

// select one entity by primary key
$project = $outlet->load('Project', 1);

// select using a relationship method
$bugs = $project->getBugs('WHERE {Bug.StatusID} = 1');

// select using fluent-interface
$bugs = $outlet->from('Bug b')
    ->with('Project')
    ->where('{b.Status} = ?', array(Bug::OPEN))
    ->find();
```



CoughPHP

Collection Handling Framework

Very light weight, very simple.

<http://www.coughphp.com>



CoughPHP Example: CRUD

Create (INSERT)

```
$customer = new Customer();  
$customer->setName('First Customer');  
$customer->save();  
$customerId = $customer->getKeyId();
```

Read/Retrieve (SELECT)

```
$customer = Customer::constructByKey($customerId);
```

Update (UPDATE)

```
$customer->setName('New Name');  
$customer->save();
```

Delete/Destroy (DELETE)

```
$customer->delete();
```



ORMer

ActiveRecord

<http://www.greaterscope.net/projects/ORMer>

ORMer Example

```
// Pull user objects "where email='me@host.com'"  
  
$users =  
  user::find()->where('email=?')->parameters('me@host.com');
```

And many more?



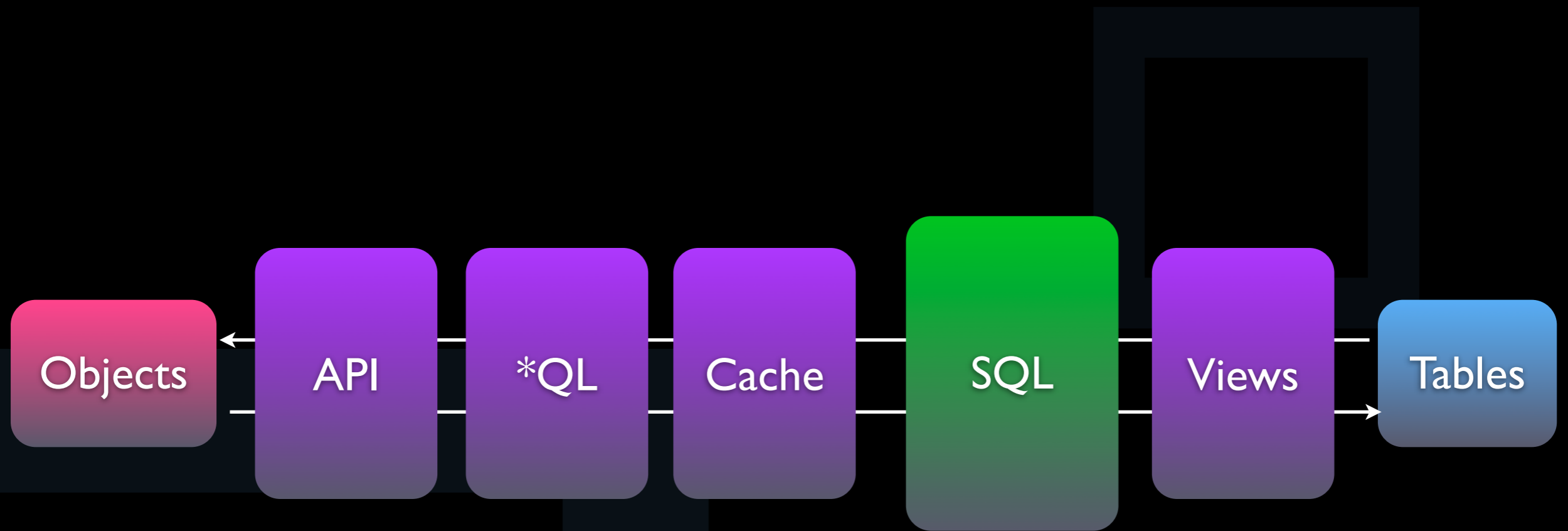
Alternatives?



Non-relational data storage

- key-value databases
- resource-oriented interfaces (think a RESTful API)
- *QL (Facebook's FQL, Google's GQL, etc.)

More layers!



Thanks!



maggienselson.com