Object Relational Mapping in PHP5
Acknowledgements
Many thanks to Matthew Weier O'Phinney and Ryan Mauger for taking the time to participate in interviews and sharing information very valuable to this thesis.

I would also like to thank my classmates Martin Lindberg, Tobias Åström, Fredrik Johansson and Dennis Sangmo at Linnaeus University who have given their constant support.
Abstract
Using an object relational mapper, ORM, is a good idea if you have an object relational code base and are following standards. By using an ORM you get a separation between actual objects and data persistence. This also makes it possible for the programmer to fully focus on the application without knowing much about how the database itself works.

However, this doesn’t come without drawbacks. Depending what ORM you use and how you use it, it may sacrifice performance.

The research in this thesis shows that using an ORM in a very small application does not sacrifice performance.
Sammanfattning

Men, detta kommer inte utan nackdelar. Beroende på vilket ORM system du använder och hur du använder det kan det påverka prestandan.

Forskningen i den här rapporten pekar på att när man använder en ORM i en väldigt liten applikation berörs inte prestandan.
# Table of contents

Chapter 1: Introduction ................................................................................................................................. 1  
1.1 Background ........................................................................................................................................... 1  
1.1.1 The relational database model ....................................................................................................... 1  
1.1.2 The object-oriented model .............................................................................................................. 2  
1.1.2.1 Objects and classes .................................................................................................................. 2  
1.2 Problem statement ............................................................................................................................. 3  
1.2.1 Research questions ....................................................................................................................... 3  

Chapter 2: Theory ....................................................................................................................................... 4  
2.1 Mapping objects to relational databases: Object Relational Mapping ........................................... 4  
2.1.1 The simple case ............................................................................................................................. 4  
2.1.2 Objects containing other objects .................................................................................................. 4  
2.1.3 The problem with inheritance .................................................................................................... 5  
2.1.4 Another approach to object oriented mapping ........................................................................... 6  
2.3 Using already existing solutions ...................................................................................................... 6  
2.3.1 Using idiorm ............................................................................................................................... 7  
2.3.2 Using Doctrine2 .......................................................................................................................... 7  

Chapter 3: Methodology ............................................................................................................................ 10  
3.1 Making example applications .......................................................................................................... 10  
3.1.1 The performance testing ............................................................................................................. 10  
3.2 Interviews ........................................................................................................................................... 10  
3.3 Criticism of chosen methods ............................................................................................................. 10  
3.3.1 Example applications ............................................................................................................... 10  
3.3.2 Interviews ................................................................................................................................... 10  

Chapter 4: Result ....................................................................................................................................... 11  
4.1 Performance of the example applications ..................................................................................... 11  
4.2 Changing database in Doctrine2 .................................................................................................... 11
4.2.1 Changing to MongoDB..........................................................................................11

4.2.2 Changing to SQLite.............................................................................................11

4.2 Advantages of using an ORM ..................................................................................12

4.2.1 Data portability ....................................................................................................12

4.2.2 Developer performance.......................................................................................12

4.2.3 Refactoring..........................................................................................................12

4.2.4 Time saving..........................................................................................................12

4.3 Disadvantages of using an ORM ............................................................................12

4.3.1 Site performance.................................................................................................12

4.3.2 Difficult to cover all use cases............................................................................12

4.3.3 A lot of objects....................................................................................................12

Chapter 5: Discussion ..................................................................................................13

5.1 Discussion of the results.........................................................................................13

5.2 Why you should use an ORM ...............................................................................13

5.3 Why you may want to think twice before doing so...............................................14

Chapter 6: Conclusion ................................................................................................15

Chapter 7: References ..................................................................................................16

7.1 Literature................................................................................................................16

7.2 Interviews...............................................................................................................16

7.3 Websites................................................................................................................16
Chapter 1: Introduction
Relational databases is one of the most common databases of today, especially in combination with PHP.

With PHP5 the object orientation of the language was greatly improved and there has been more focus on object orientation.

Then the problems start, you have an object oriented codebase and a relational database. How do you as a programmer combine these in the best way? One of the ways is using a technique called object relational mapping, ORM.

1.1 Background
In studying object relational mapping we need a full understanding for what it is and how it can be implemented in PHP. We need to know what the relational database model is and how it works. It’s also important that we know how object orientation work.

In this section both models will be briefly explained.

1.1.1 The relational database model
The year 1969 Edgar F. Codd issued his paper "A Relational Model of Data for Large Shared Data Banks".

The relational database model is often compared with tables.

![Figure 1.1.1.1; The relational model](image)

As we can see in figure 1.1.1.1 we have a table with columns and rows, every column is an attribute and the values in a column is of the same type. Together these columns create rows, also called tuples. A row in a relational database is often called “a set of values”.

The entire table is called a relation. Relations can be connected to each other by pointing to unique values in another relation, called keys.

Relational databases are queried with a language called “SQL”, structured query language.
1.1.2 The object-oriented model
To fully understand how object relational mapping works you have to be a programmer with extensive knowledge of object-oriented programming, when you read this thesis some knowledge is assumed. But here is a very quick recap to get you up to speed.

1.1.2.1 Objects and classes
Objects are made from classes. In fact, you often call an object “An instance of a class.” So an object is a type of the specific class. To understand this further let’s look at an example.

We have a car. What is a car? To really simplify it we can say a car has a colour, 4 wheels and a shape. A car can also do several things like drive and brake. A car is an object.

To make this car they need a blueprint at the factory, this is where the class comes in. The class is simply the drawing of the car. Every car has the same attributes and the same functionality but it may differ in shape or colour which makes it a unique car, a unique object.

Objects may also inherit the attributes and methods from parent objects making object-orientation very flexible. Let’s say we have a generic car class defining all the important parts of a car. But then we also have different kind of cars like a sports car or a family car with specific attributes of their own, they may be children of the generic class “car”.

This is a way that object oriented programming is often explained. It’s a way to make the parts of the program remind us more of every day life.

1 The car reference was inspired by the book “Objects First with Java” By David J. Barnes and Michael Kölling
1.2 Problem statement
Data management in PHP5 is typically implemented using non-scalar values or “entities”. Such as a person or a house. But many of the most popular databases store scalar values.
This forces the programmer to convert the objects into simple scalar values for storage in a database or to only use scalar values through the entire application.

In a typical object oriented approach there’s many advanced objects so the second approach is often no good solution. This is where the object relational mapper comes in: To function as a middle hand to connect the two.

1.2.1 Research questions
• How do you go about implementing an object relational mapper in PHP5 and MySQL?
• Why should you use an object relational mapper?
• What already existing object relational mappers are there for PHP5?
• Can you easily switch between different databases with an ORM?
• What disadvantages are there in using an ORM?
• Are there any significant changes in the execution speed of database queries?
• Does it sacrifice the design of the database and will that in turn sacrifice performance?
Chapter 2: Theory
This chapter will go through the theory in mapping relational databases with object oriented codebases. General knowledge of object oriented programming and relational databases will be assumed.

2.1 Mapping objects to relational databases: Object Relational Mapping
We have our objects and we have our tables. The goal is to make a mapper so that the programmer themselves will only have to code in an object oriented way and not think in the terms of tables/relations. This will also separate the database from the code and make it easier to change database in the future if wanted.

2.1.1 The simple case
A simple example is when an object consists of single values such as name, e-mail and phone-number.

These objects can beneficially be directly translated to a relation with scalar values in the database. Every object gets a row of their own. Then when you collect the information again the rows get converted into objects. Figure 2.1.1.1 reflects this.

2.1.2 Objects containing other objects
An object can also contain a collection of other objects. A typical example is a person who has a collection of numbers. To reflect this in a relational database one approach is to have two relations: Person and number. The number will have the ID of the person thus connecting the two. Figure 2.1.2.1 reflects this.

Figure 2.1.1.1; A simple mapping

Figure 2.1.2.1; A person with more than one number
There’s also a case of addresses where more than one person has the same address and one person may have more than one address. To reflect this in a relational database there would typically be three relations. Person, Address, AddressToPerson. The first relation reflects the person and has a unique id. The second relation reflects the address and has a unique id. The third relation connects the two by having two attributes: PersonID and AddressID, this way an address can be used by several persons and a person can have several addresses. Figure 2.1.2.2 reflects this.

![Figure 2.1.2.2](image)

**Figure 2.1.2.2; An address shared by many persons where a person can have more than one address.**

### 2.1.3 The problem with inheritance

One of the biggest problems in object relational mapping starts when we’re discussing inheritance. Let’s take the example of a person again. We may have an application for a school. There we have the generic person. Every person has a name and a social security number. Then we have the teachers who also has an employee id and the students who has a student id, these inherit from Person. ²

How do you reflect this in the relational model?

A common approach is to make a relation for every class. Thus a person relation, a student relation and a teacher relation. The person has a unique id which connects a specific person to a specific student or teacher. Figure 2.1.3.1 reflects this.

---

² Naturally in a real school application every person would have more information than that.
2.1.4 Another approach to object oriented mapping

When you try to map objects to model another approach is to simply map the objects into relations instead of mapping the content of the objects into relations. So you may have a class relation, this class relation has attributes, which has values. This approach quickly becomes very big and is therefore not seen in any of the PHP solutions that has been looked at and will therefor not be covered more than this in the thesis.

2.3 Using already existing solutions

When it comes to object relational mapping it’s very complex to cover every single case. Therefore it is often advised to use an already existing solution instead of building your own as those solutions have a lot of the work already figured out.

There’s many ORM’s out there for PHP. Some of the common ones are: Propel, Doctrine, Outlet ORM, Flourishlib, RedBeanPHP and a bunch of smaller ones such as idiorm created by PHP developers to have a simple ORM solution.

---

3 There are more solutions than the researcher can cover, there may actually be one that uses a solution like this that the researcher doesn’t know about.

4 As confirmed in the interview with Ryan Mauger
In this thesis two ORM’s for php5 was tested. The small and simple idiorm\(^5\) and the large Doctrine2\(^6\).

\subsection*{2.3.1 Using idiorm}
Idiorm is meant for smaller applications with simple CRUD\(^7\) operations only and is therefore a perfect candidate for a lot of simple websites out there.

To use idiorm you download it from github, include the single class in your code and add some configuration parameters. When this is done you can start querying the database with simple queries. It is very small and simple to use\(^8\).

\subsection*{2.3.2 Using Doctrine2}
Doctrine2 is a full fledged ORM framework and is one of the biggest ORM solutions for PHP5. It covers most situations and implements cache and lazy loading to keep the performance high.

There’s several ways to install Doctrine2. The documentation explains this very well.

When you then have installed Doctrine2 there’s some things that you need to configure. This ORM uses so called entities, then you can choose to have them in PHP, XML or YAML. In this thesis PHP is used for consistency.

Doctrine2 requires PHP 5.3 and to use it you also need good knowledge of namespaces as it’s being used within.

---

\(^5\) [https://github.com/j4mie/idiorm](https://github.com/j4mie/idiorm) - 2011-05-22

\(^6\) Version 2.0.5 was used.

\(^7\) CRUD - Create, Read, Update, Delete (In SQL: Insert, Select, Update, Delete)

\(^8\) Based on the experience of the researcher - This can vary depending on programmers experience.
When you have setup the classloading you may begin using Doctrine 2. First of all you create your entities as previously mentioned. Figure 2.3.2.2 displays an example of an entity in Doctrine2 using PHP. To define what type the attributes should be there’s PHP comments in the entity. To enforce good object orientation all attributes are private and has getters and setters set to them.
When you have created your entities you can have Doctrine2 generating the database schedule for you by using either the console tool that comes with Doctrine2, or writing a few lines of code.

Then you can communicate with the entity manager to save and load entities from and to the database.

Doctrine2 has more than one way to query the database. One of the most flexible ways is the DQL, Doctrine Query Language which is structured in an object oriented way.

Due to the way that Doctrine2 is programmed it’s very easy to change from one supported database to another. To change to SQLite all that needs to be done is change the connectionOptions and you have an SQLite database instead of a MySQL database.

To change to for instance MongoDB you need another version of Doctrine2, you need an ODM, object document mapper and not an ORM.

Doctrine2 is a big and powerful ORM framework and is useful if you have a bigger object oriented PHP application.

```php
namespace Entities;
/**
 * @Entity @Table(name="users") */
 class User
 {
  /**
   * @Id @Column(type="integer")
   * @GeneratedValue(strategy="AUTO")
   */
  private $id;
  /**
   * @Column(type="string", length=128)
   */
  private $forename;
  /**
   * @Column(type="string", length=128)
   */
  private $lastname;
  /**
   * @Column(type="string", length=128)
   */
  private $email;
  /**
   * @Column(type="string", length=128)
   */
  private $phone_number;
  public function getId()
  {
    return $this->id;
  }
  public function getForename()
  {
    return $this->forename;
  }
  public function setForename($name)
  {
    $this->forename = $name;
  }
  public function getLastName()
  {
    return $this->lastname;
  }
  public function setLastName($name)
  {
    $this->lastname = $name;
  }
  public function getEmail()
  {
    return $this->email;
  }
  public function setEmail($email)
  {
    $this->email = $email;
  }
  public function getPhoneNumber()
  {
    return $this->phone_number;
  }
  public function setPhoneNumber($number)
  {
    $this->phone_number = $number;
  }
 };
```

*Figure 2.3.2.1; An entity in Doctrine2*
Chapter 3: Methodology
In this chapter the methods used to gather the information will be discussed.

3.1 Making example applications
To gain further knowledge about ORM an example application was made.

The application in question is a very simple address book to demonstrate two different ways of implementing an ORM and how you would typically implement such an application without using an ORM. Also in the Doctrine applications there was two versions, one using MySQL5 and one quickly changing to SQLite to further show and understand how easy it is to change the database.

The application was then tested to see which ones generated the quickest queries.

3.1.1 The performance testing
To test the performance of the MySQL applications slow query logging is used together with long query time to simply log all queries. Some queries will be taken out and a medium time will be reported.

3.2 Interviews
When it comes to using an ORM opinions of the programmers using it is a big part. To gain a deeper understanding of ORM in general and the opinions of people that use it daily two programmers were interviewed. Ryan Mauger who uses ORM daily in his work and Matthew Weier O'Phinney who’s the project leader of the Zend Framework, a big PHP framework.

3.3 Criticism of chosen methods

3.3.1 Example applications
When making example applications every case could not be covered as this would take way too long time, therefore the more advanced cases with an ORM has not been covered and will not be taken into consideration in the performance testing.

This is a part that may or may not have affected the results, especially when using a bigger ORM framework such as Doctrine2.

The time it takes to execute a query is also dependent on how strong the computer is. Since all of the applications was tested on the same computer, this should not affect the difference between the queries.

3.3.2 Interviews
Only interviewing two people doesn’t give a fair amount of opinions that differ but it does give valuable information from professionals who uses ORM daily and gives another point of view and some strength to the discussion.
Chapter 4: Result

4.1 Performance of the example applications
The performance testing generated the same results for all different solutions and the conclusion is therefore that for simple applications the performance of the database is not affected by using an ORM.

An example query from the log looks like this:

```
# Query_time: 0.000472  Lock_time: 0.000210  Rows_sent: 0  Rows_examined: 0
SET timestamp=1306236984;
INSERT INTO users (forname, lastname, email, phone_number) VALUES ('fsdds', 'dsdffds', 'dsdsffsd', 'dsfdfs');
```

Medium insert query time was: 0.000468

Delete query time was: 0.000186

Update query time was: 0.000216

Select all query time was: 0.000219

4.2 Changing database in Doctrine2
One of the many benefits with an ORM is the possibility to change the database if you wish. This has been tested in Doctrine2.

4.2.1 Changing to MongoDB
Changing to MongoDB in Doctrine2 requires that you use an ODM instead of an ORM. While Doctrine2 provides this it requires changing your structure and the libraries used with the application.

4.2.2 Changing to SQLite
Changing to SQLite in Doctrine2 required one single change. From this:

```php
$connectionOptions = array(
    'driver' => 'pdo_mysql',
    'user'  => 'user',
    'password' => 'password',
    'host'  => 'localhost',
    'dbname' => 'addressbook',
    'port'  => '3306'
);
```

To this:

```php
$connectionOptions = array(
    'driver' => 'pdo_sqlite',
    'path' => __DIR__ . '/db.sqlite',
);
```

Then you use the tool to generate the schema again in your SQLite database, and you’re ready to go. Thus the conclusion is that changing from one relational database to another in Doctrine2 is easy but changing from a relational database to an object oriented database requires some changes.
4.2 Advantages of using an ORM

4.2.1 Data portability
When you use an ORM you’re separating your programmatic object design from where you store them. This means that a developer can actually change the persistence layer at any time with only minor changes of the code. We saw this specifically when changing from MySQL5 to SQLite in Doctrine2, only a few configuration changes and the change was done.

This means that you can also easily change your ORM as it’s all in one place that’s all you have to change and your objects and general logic still stays the same. So data portability is a strong and powerful usage of an ORM.

4.2.2 Developer performance
A programmer who uses an ORM doesn’t have to know much about relational databases or SQL. Therefore an expert developer can focus on the task at hand without necessarily knowing the details behind the persistence layer.

4.2.3 Refactoring
When you’re using an ORM the database is typically in one place, the ORM. So it becomes easy to refactor the application.

4.2.4 Time saving
When you have learnt how to use the ORM it saves a lot of time since you don’t need to structure the database every single time you create a new application. You just create the objects and entities in the ORM and it does the structure for you.

4.3 Disadvantages of using an ORM

4.3.1 Site performance
Sometimes the code the ORMs use is not as optimal as writing the SQL code by hand. In some cases it may result in dragging the performance of the general site down. Most ORM’s however also have the possibility to write your own SQL when desired. Knowing when this is needed can be a big issue.

4.3.2 Difficult to cover all use cases
There’s many ways to design a database and there’s a lot to object oriented programming. To cover all use cases in an ORM is a difficult task and most ORM’s does not succeed. There may be cases where you may actually have to change your code to reflect the ORM.\(^9\)

4.3.3 A lot of objects
When using an ORM every row in the relational database is often reflected as an object. These objects then lurk in your application. If you’re not careful it can become too many objects and it will drag site performance down. Some ORM frameworks try to minimize this, Doctrine2 has made it a specific goal.

---

\(^9\) As pointed out in the interview with Ryan Mauger
Chapter 5: Discussion
My experience now shows that using an ORM or not fully depends on your intentions. There are some good alternatives to them one being creating domain-specific mappers. Then you can tailor those mappers directly for the type of data you want to persist.

Often changing from one database to another is a rare task and it’s more common to change the structure of your application or database than changing the type of database used.

5.1 Discussion of the results
The performance testing surprised me a lot. I was expecting to get different queries and performance between idiorm and Doctrine2 specifically. But the fact is that it was such a simple application with a very simple object this generated the most simple query in all cases which was the same type of query, a simple select, update, insert or delete.

Since the query was the same it didn’t effect the performance of the database. Also the building up of the database by Doctrine schedule built exactly the same schedule as the one I built by hand. This of course makes the application exactly the same as far as the database is concerned. Which is actually a good thing. You typically want the ORM to “think in your way”.

The problem here is that I didn’t have time to test more advanced use cases. It would be very interesting to see what would happen if I have a very advanced object oriented application and try using Doctrine2, how much would it effect performance? For this, I’m listening a lot to the opinions of Matthew Weier O’Phinney who says following:

“Because ORMs abstract away how relations work, sometimes the code they utilize can be sub-optimal. As an example, you will often end up with N+1 queries, or cases where the SQL generated cannot hit indices easily (if at all).”

He also says:
“While a good ORM will allow you to drop down to native SQL when desired, it's often difficult to find the exact point where and when to do that within your code. As a result, ORMs have a reputation of dragging site performance down.”

5.2 Why you should use an ORM
“If you are following good object oriented programming paradigms and writing good objects that compose other objects, an ORM is often a good fit.” - Matthew Weier O’Phinney

You get all of the advantages as discussed in the previous chapter:

- Data portability
- Developer performance
- Easy refactoring
- Time saving
5.3 Why you may want to think twice before doing so

Using an ORM of your own takes a lot of time and as mentioned earlier it’s very hard to cover every use case.

When you use an already existing ORM that does cover the use cases the footprint is huge. If you’re making a small application it may be beneficial to not use an ORM because of that. But where is the border?

That’s up to every programmer to decide. My personal opinion is that the border is where the application will demand a lot of refactoring if you want to change the persistence later without using an ORM or similar solution.

If you have a small application you may want to think about using idiorm, as briefly discussed in chapter 2. This will create a small separation in the layers.

There’s also several alternatives to using an ORM, as a programmer you may have something that you prefer using instead. Look at the alternatives before you use an ORM.

Using an ORM has a lot of benefits but the learning curve is always high. Figure out if the learning curve is worth it before you dive in.
Chapter 6: Conclusion
Now we have answered all the research questions. In short, using an ORM has both advantages and disadvantages, think of them before you use one.

How do you go about implementing an object relational mapper in PHP5 and MySQL?
There’s many ways of doing this. One of them is using an already existing mapper such as Doctrine2 or the smaller idiorm.

Why should you use an object relational mapper?
Because there’s many advantages. It’s easier to refactor your code and the programmer doesn’t have to know anything about the persistence layer which means he can focus at the task at hand. When you follow good object oriented principles, using an ORM is a good idea.

What already existing object relational mappers are there for PHP5?
Doctrine2, idiorm, propel, redbeans and many others.

Can you easily switch between different databases with an ORM?
Yes, if the ORM supports the other protocols. In Doctrine2 it was very easy changing between MySQL and SQLite but changing from MySQL to MongoDB was not as easy.

What disadvantages are there in using an ORM?
There’s many objects circling around which can affect site performance. Also the queries created by the ORM may not be optimal even if most ORM’s allow you to write your own SQL it may be hard to know when to do so.

Are there any significant changes in the execution speed of database queries?
The tests in this applications showed that for a very simple application there are no significant changes but for bigger applications there may be due to the queries not being optimized. The fact is that sometimes the ORM may even optimize the query better than the programmer did.

Does it sacrifice the design of the database and will that in turn sacrifice performance?
You can have the ORM making the database design for you, then it may sacrifice the performance depending which ORM you use and how well it is at the task. You can also choose to make your own database structure, then it doesn’t sacrifice performance in any way.
Chapter 7: References

7.1 Literature

E.F. Codd. (1969) *Derivability, redundancy and consistency of relations stored in large data banks*, IBM Research Report

7.2 Interviews
Interview with Matthew Weier O'Phinney
Interview with Ryan Mauger

7.3 Websites
http://www.doctrine-project.org/ [2011-05-10]