**СОФИЙСКИ УНИВЕРСИТЕТ**
**"СВ. КЛИМЕНТ ОХРИДСКИ"**

---

**Факултет по Математика и Информатика**
**Катедра „Информационни Технологии”**

# ДИПЛОМНА РАБОТА

**тема:**

Обектно-Релационна Персистентност на Данните
(Object Relational Mapping)

**Дипломант: Йордан Йорданов**

**Специалност: Информатика**

**Специализация: Софтуерни технологии**

**Научен ръководител: доц. Силвия Илиева**

**София, 2007 г.**

# Съдържание

# Увод

Автоматизирането на работата с база данни е проблем, по който отдавна се търси решение. Тенденцията е в кода на програмиста да липсват истински SQL заявки, вместо тях да се използва някакъв друг похват, а грешките да се откриват на ниво създаване (компилиране) на програмния продукт или поне да не са заплаха за съдържанието на базата.

Видовете Frameworks за обектно-релационна персистентност се базират на логическата връзка между класовете и таблиците в базата, като последвалите заявки - (Selects, Inserts, Updates, Deletes), вкл. създаването и евентуално променянето на самата база се извършват от самите тях. Няма ограничение към конкретната база, която се използва, стига да се даде конкретна реализация на основните операции с нея, изведени обикновено в интерфейс. От друга страна работата с база данни е просто един частен случай на сериализиране, също както и работата с XML, така че обхвата на работа с такъв тип програмен продукт може да бъде дори по-широк.

# Цел и задачи на дипломната работа

Тази дипломна работа може да бъде ползвана от широк кръг хора като източник на детайлна информация в областа на ORM технологията и имплементацията и със средствата на обектно ориентираното програмиране. Разглеждат се различни ОРМ системи, като за всяка от тях се дават предимства и недостатъци. Проследява се еволюцията на ОРМ технологията. Разглеждат се различни похвати за ОРМ персистентност – също с положителни и отрицателни черти. Приложението към дипломната работа има повече демонтративен характер и не претендира за пълнота във функционалност и архитектура. Въпреки това то представлява добър пример за изследване, разработване и имплементиране на система, базирана на ORM технологията.

Целта на дипломната работа е да се изследват и покажат с помощта на примери похватите на Обектно Релационна Персистентност (Object Relational Mapping - ORM).
След като бъде разгледана, читателят трябва да може да намери отговорите на следните въпроси:

Как може да се представят данните от обектите като релационни дании?

Как това може да става автоматично?

Какво усилие ще е необходимо за имплементирането на ORM система, и какви ще са ползите и недостатъците за приложението?

Структура на дипломната работа:

- Глава 1 „ORM системи” разглежда различни ОРМ системи, като подредбата е съгласно тяхната функционалност и пълнота. Също така се обхващат системи за различни езици с възможности за ООП (Обектно Ориентирани Програмиране).

- Стратегии за Обектно-релационно съответствие – изследват се различните похвати за ORM. Изброяват се положителни и отрицателни страни за всеки похват.

- Мета Език – допълнителна информация за ORM системите – Показва се нуждата от допълнителна информация, нужна за работата на ORM системите, както и конкретните възможности.

- Положителни и отрицателни страни на ORM системите – обективна оценка на базата на вече изложената информация

- Реализиране на ORM framework – практическо прилагане и демонстриране на идеите на ORM.

# Глава 1 ORM системи

Работата на все повече компютърни системи, независимо от тяхната големина и сложност, използват база данни. За това и интегрирането на базата, нейното използване и търсенето на по-голяма ефективност при работата с нея, води до все по-усъвършенствани драйвери, ком обекти или пък до свеждането на базата до приложение, което е написано на същият програмен език, и ползва файловата система например. Разбира се в общия случай не можем да очакваме някаква ефективност от заместването на legacy СУБД системата с нещо друго, което я копира, даже напротив. Обикновено тези „опростени“ бази данни се ползват в процеса на имплементиране, тестване и демонстриране на продукта. А да очакваме видимо подобрение на работата на приложението, само защото използваме по-нов драйвер за връзка с базата също е немислимо. Все пак работата на този клас или модул се свежда до комуникация с базата данни, изпращане на заявки и евентуално преобразуването на резултата във вид, подходящ за работа. На това ниво не се извършват никакви оптимизации относно самите заявки и съответно времето за отговор.

През последните години се обръща все повече внимание на Обектно ориентираното програмиране. Тъй като обектите реално могат да описват всичко, се обръща внимание на това дали таблиците в базата данни не могат да бъдат използвани също през обекти. Нещо повече – обектите могат да капсулират в себе си голяма част от функционалноста по създаване на връзка към базата, извършването на нужната заявка(например select), инициализиране на полетата в обекта, затваряне на връзката към базата, дори прихващане на грешки при протичането на този процес. Така стигаме всъщност и до идеята за Обектно - Релационно трансформиране на информацията или просто ORM система.

Английската дума ORM (Object-Relational Mapping) означава обектно-релационно съпоставяне. Понеже релационната база данни е начин за съхранение на данните (персистентност), можем да използваме и термина ОРП (Обектно-Релационна Персистентност) .

Понастоящем съществуват множество обектно ориентирани езици, като всеки език се откроява с някакви специфики. Ние ще разгледаме конкретни реализации на ORM системи за някои от обекто ориентираните езици, преди да обобщим идеята за обектно – релационната персистентност на данните. Разглеждането на различните ОРМ системи ще бъде в последователност, определена от програмния език и конкретните характеристики, без оглед на хронологическият ред, в които са били разработени.

## 1.1. Propel (PHP).

Езикът PHP не е основоположник на ORM идеята, нито съществуват някаква особена функционалност в библиотеките за автоматизирана работа с Бази Данни, но той все пак е обектно ориентиран след своята 5-та версия, а поради това, че е скриптов език и лесно се администрира, е може би най-масовият език за интернет приложения.

PHP Data Objects (PDO) не е нищо друго, освен стандартизиран интерфейс за достъп до базата данни (виж PHP Manual, Introduction,[1]). Това означава просто абстрактно ниво за достъп до данните (DB layer). За автоматизиран достъп до самата база не може да се говори, а предимствата в използването на PDO се изчерпват в скриването на конкретния драйвер и стандартизиране на неговата функционалност. За да покажем идеите на автоматизирана работа с базата днни, реализирана на PHP, ще разгледаме PROPEL (ref.1). Този продукт взаимства идеите си от свой предшественик, написан на Java. Това, което прави той, е просто „обличане“ на таблиците от базата данни в класове. Освен това може да създава и поддържа таблиците в базата. Понеже самият той създава класовете, то програмиста трябва да окаже какви са самите те. А това се осъщесвява чрез XML. В последствие ще видим, че това е често използван похват, като начин за добавяне на мета данни към помощна програма, която да създава класове с ORM функционалност. Един елементарен варант на База Данни с 2 таблици – книга и автор, може да се опище по следния начин.

*Примерен код 1 а), XML конфигуриране на Propel*

```
<table name="book">
 <column name="book_id" type="INTEGER" required="true" primaryKey="true"/>
 <column name="title" type="VARCHAR" size="100" required="true"/>
 <column name="author_id" type="INTEGER" required="true"/>
 <foreign-key foreignTable="author">
   <reference
     local="author_id"
     foreign="author id"/>
 </foreign-key>
</table>
<table name="author">
 <column name="author_id" type="INTEGER" required="true" primaryKey="true"/>
 <column name="fullname" type="VARCHAR" size="40" required="true"/>
</table>
```

Поддържат се също onDelete="CASCADE" и onDelete="CASCADE" атрибути за foreign-key елемента, като програмно се реализира дори когато базата данни не поддържа тази функционалност. Връзката между таблиците в базата и класовете е една таблица - един клас. За това и връзки от тип много към много могат да се осъществят чрез отделен клас. Нека към горният пример добавим таблица Читател:

*Примерен код 1 б)*

```
<table name="reader">
 <column name="reader_id" type="INTEGER" required="true" primaryKey="true"/>
 <column name="Name" type="VARCHAR" size="100" required="true"/>
</table>
```

Сега връзката между книга и читател можем да опишем така:

*Примерен код 1 в)*

```
<table name="book_reader_ref">
 <column name="book_id" type="INTEGER" required="true" primaryKey="true"/>
 <column name="reader_id" type="VARCHAR" size="100" required="true"/>
 <foreign-key foreignTable="book">
   <reference
     local="book id"
     foreign="book_id"/>
 </foreign-key>
   <foreign-key foreignTable="reader">
   <reference
     local="reader_id"
     foreign="reader id"/>
 </foreign-key>
</table>
```

А както вече казахме, това води и до клас, който съответства на таблицата за връзка Book_reader_ref.

Приложението успява да преодолее до някаква степен това неудобство, като генерираните от него класове освен стандартните методи, които можем да очакваме, съдържат и такива, които правят JOIN с таблици, към които има Foreign Key.

*Примерен код 2, Propel и PHP*

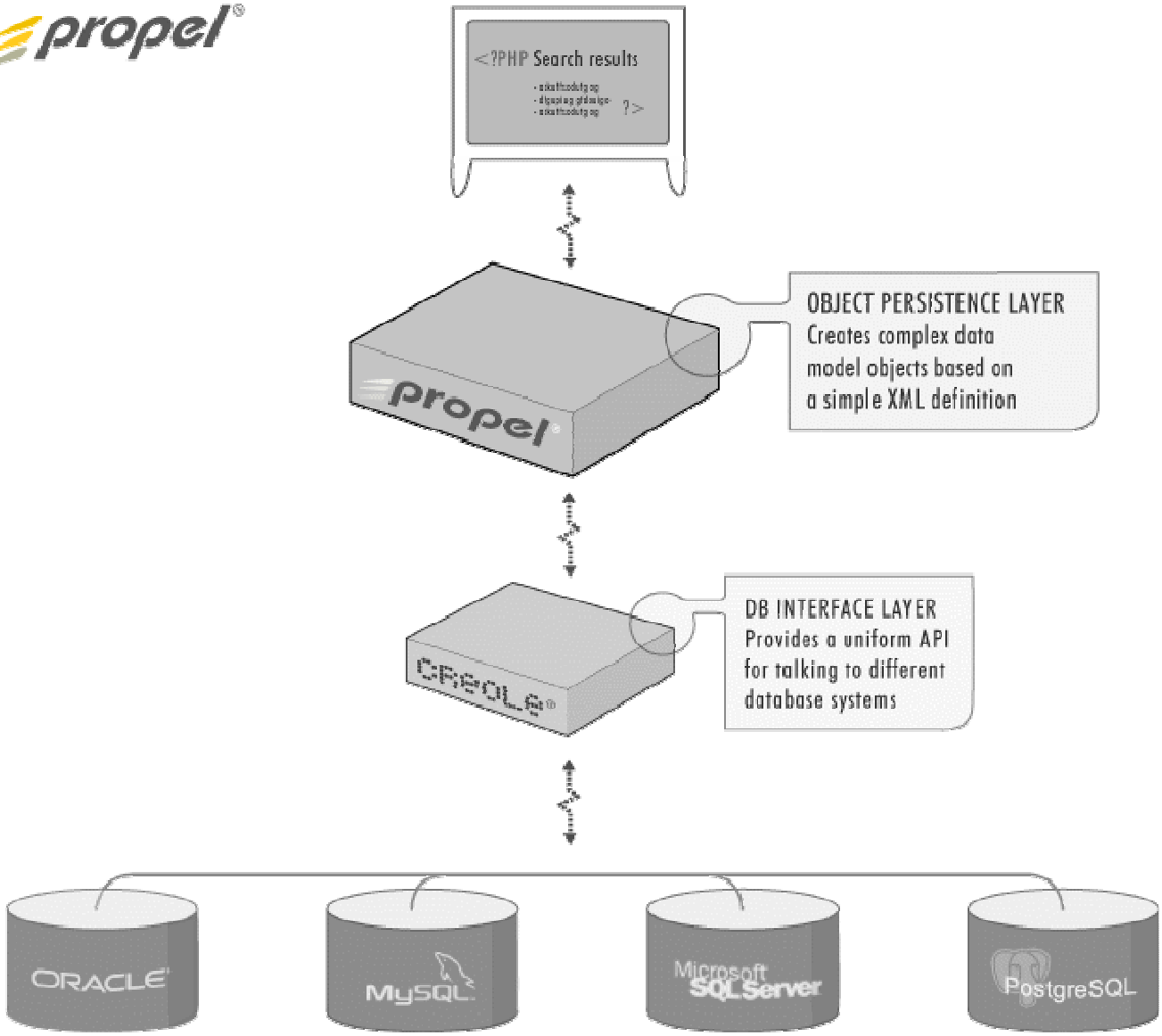
```
<?php
$books = BookPeer::doSelect(new Criteria());
// за всяка книга вземаме всичките и читатели
foreach($books as $book) {
  $readerRefs = $book->getBookReaderRefsJoinReader();
  foreach($readerRefs as $ref) {
    $reader = $ref->getReader(); // Тук няма допълнителна заявка,
                                 // тъй като ползваме JOIN метод на класа
  }
}
```

Този код изпълнява n+1 заявки, където n е броят на книгите, върнати от първата заявка.

Това, което можем да кажем в обобщение за PROPEL е:

- Генерира и класовете и скриптовете за базата танни
- Използва се XML за описанието на мета танните
- Представлява ниво за съхраняване на обектите(object persistence layer)
- Реализира ORM като на всяка таблица съответства клас.
- Симулация на Foreign Keys за бази, които не ги поддържат (вкл. OnDeleteCascade)

Можем да видим и мястото на този продукт в многослойна среда на фиг. 1




**Фиг.1, Propel в многослойна архитектура**

## 1.2. Torque (Java)

Според спецификацията на PROPEL, този продукт е копие на Torque, вариант на ORM, писан на Java. Java е платформено независим език за програмиране и обектно ориентиран. Масово се използва за широк кръг от приложения, за това е и нормално да очакваме най-разнообразните имплементации на ORM система да са именно имплементирани на и за Java.

Тъй като по същество Torque и PROPEL целят една и съща фунционалност, разликите между двете идват главно във възможностите на средата. Създателите на Torque разделят този продукт условно на 4 сравнително обособени част(модула):