Term Paper

Object Oriented & Object Relational Mapping Databases

Alexander R. Ramsey

DBST 651

University of Maryland, University College

Abstract

Increasing complexity of applications created by the incongruency of relational database and application languages has created a phenomenon called the Impedance Mismatch. Object oriented relational mapping (ORM) systems aim to alleviate these problems by integrating the best capabilities of both. This term paper explores what an Object-oriented relational mapping system provides in advantage and disadvantage through the lens of a high level analysis of a popular ORM system called Hibernate.

Table of Contents

[Introduction 4](#_Toc500072936)

[Object Relational Mapping 5](#_Toc500072937)

[Advantages and Disadvantages of ORM 6](#_Toc500072938)

[ORM Variants Available 7](#_Toc500072939)

[How ORM Solutions Work 7](#_Toc500072940)

[Hibernate 8](#_Toc500072941)

[Hibernate Architecture 9](#_Toc500072942)

[Conclusion 10](#_Toc500072943)

[References 11](#_Toc500072944)

# Introduction

Object Databases are Database Management Systems which utilize Object Oriented Design as the guiding template for Schema Development. They differ from Relational Database systems primarily due to their capability of integration with application systems. Object Oriented Databases were developed in the mid-1980s as a response to the need for more complex data storage, for applications like Computer Aided Design (CAD), for example (Stajano, 1998).

Object Oriented Database seek to address these complexity problems by integrating database and application systems together. The disconnect created by Relational Databases and their associated Applications is referred to as an “Impedance Mismatch”. An Impedance Mismatch occurs when a combination of SQL exists with the host language of an application, when the interface with SQL databases are not clear, and particularly when the host language uses object oriented design principles to define data models. Object Databases combine the best features of an object-oriented language and a database management system by handling data as objects, providing the features of object definitions, and by providing access to SQL for more complex data management logic (Signer, 2010).

Object Databases that utilize an Object Relational Mapper (ORM) provide benefits to the software developer because they address the Impedance Mismatch. ORM’s address the routine storage of Object data within an application environment, providing convenience for developers. The end goal is to integrate the programming language of the application with that of the applications’ Database Manipulation Language (DML).

# Object Relational Mapping

Persistence is the concept that data will remain beyond the realm of an applications temporary memory, which would be used to fulfill business logic needs in real time. Persistence creates a stored state for later use. It is particularly helpful in modern web applications, where data is stored for users of an authenticated website among a host of many other applications. Persistence can also be used for other software applications where data is to be stored long term.

As the complexity of these applications grows, so does the data persistence layer of the application over time. To address these concerns, object oriented databases can be used alongside Object Relational Mappers (ORM). An ORM addresses the Impedance Mismatch problem through the following 5 concerns (Hibernate, n.d.):

* Granularity, the concept that more detail exists in the object model than that of the relational table model, increases complexity and discrepancy between these differing models. This rift must be managed and doubles the work load, understanding, and implementation of requirements for developers and database administrators.
* Subtypes, the concept that an object can have variants of itself which inherit some form of the parent’s data, is unique to object oriented programming but not unique to the available relational database models. Relational databases utilize keys to define relationships, which limit the behavior and increase complexity of defining such relationships among data models.
* Identity, the concept that unique data rows are paired via key relationships, providing limited capability to establish unique relationships. Relational database systems uniquely identify individual rows, whereas object oriented databases can identify identity and equality, providing more capability to the developer.
* Associations, when defined in a relational database model must be defined in both directions using primary and foreign key relationships. Such associations are sometimes classified as aggregate or as composition relationships.
* Data Navigation, the concept that extracting data relationships requires some movement across a data model. In relational database models, join statements are used to navigate the model. These join statements decrease performance and increase complexity. Alternately, object models can be easily navigated via the object network.

## Advantages and Disadvantages of ORM

Advantages of using an ORM may include the reduction of development time, the reduction of development costs, and resource costs caused by the redundancy of SQL code, vendor specific database language translation, and upkeep of multiple data models. Disadvantages of using an ORM may include an initial learning curve to establish a workflow with the ORM tool, an induced lack of understanding in how data objects are manipulated at the database level, and difficulty in producing proper results where complex data queries are needed (Signer, 2010). However, these advantages and disadvantages vary from one ORM tool to another. Much discussion exists addressing the disadvantages stated above, with argument that modern ORM tools provide just as much flexibility in handling complex logic as traditional RDBMS systems do.

## ORM Variants Available

A wide variety of ORM solutions exist, with a well-documented list available online via Wikipedia, which outlines available solutions by base programming language. Although not a comprehensive list, it reinforces the reality of how ORM remains a highly relevant solution to the Impedance Mismatch. Popular languages like C++, Java, .NET, PHP, and Python provide multiple offerings of ORM. Some solutions like Entity framework for .NET are baked into the core language, with support offered by Microsoft and a large open source community. Hibernate is a popular ORM for the Java Programming Language.

## How ORM Solutions Work

Object Databases allow for special data types like arrays, sets, and lists. In addition, Object Databases utilize has-a and is-a relationships. Like stored procedures, methods containing actions on an object are scanned by the ORM tool depending on what functionality the tool provides. And in communicating with each other, strong typing is preserved amongst both the core language and the language of the ORM tool. Differing flavors of implantation of an ORM exist per ORM tool and should be chosen by the developer to best suit the requirements of the application. A few variations of ORM tools for Hibernate exist, for example and are as follows (Bauer, 2007):

* Pure Relational, utilizes an ORM interface with a database while maintaining relational database design. Best used for less complex applications where code reuse comes at little to no consequence. Business logic would exist in the form of stored procedures on the database side.
* Light Object Mapping, utilizes an ORM interface with a database where the database is populated manually. Like Pure Relational, best used for simple applications where code reuse comes at little to no consequence.
* Medium Object Mapping, utilizes an ORM interface with a database where SQL code generation occurs automatically at build/run time with the application code. Stored procedures are not generated in SQL and a level of code reuse is limited to ensure that the application can scale to medium and large sizes while minimizing complexity.
* Full Object Mapping, supports all features of a complex ORM tool. Understands the relationships establishes by object oriented principles. These include polymorphism, inheritance, lazy loading & fetching are supported, caching strategies for data management. This level of implementation of ORM tool is best for large scale complex applications.

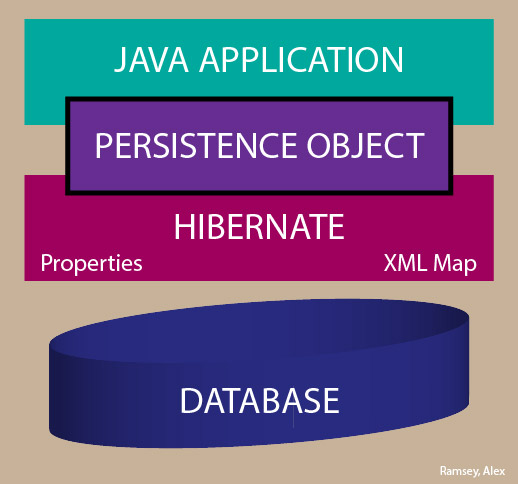
# Hibernate

Hibernate is an ORM tool created for use with Java, provided as an open source solution. Hibernate was created by Gavin King in 2001. Hibernate acts as a Persistence Layer manager by allowing developers to maintain existing database technology. Hibernate saves developers time by interpreting the data model of a Java Class into a SQL relational model (Bauer, 2007).

Hibernate manages the mapping of Java classes with a simple set of APIs that handle common CRUD operations which developers would encounter with a relational database model system. Hibernate supports many databases include MySQL, Oracle, PostgreSQL, and even Microsoft SQL Server. Hibernate works well with the Eclipse Integrated Development Environment (IDE) via plug ins and via the Maven dependency manager (Bauer, 2007).

## Hibernate Architecture

Hibernate utilizes a layered architecture, reducing the need for understanding by the developer of the database and persistence layer to implement simple persistence tasks like CRUD (Create, Read, Update, Delete tasks). Below is a diagram which provides a high-level overview of the Hibernate architecture in a Java application (Bauer, 2007):



To connect to the database, Hibernate utilizes Java Database Connectivity drivers, sometimes referred to as JDBC. The configuration object is represented in the diagram above as the Properties and XML Map, which identify how Hibernate is to connect to the database. These configuration files are generated during initial setup of the application and are rarely modified once the application is in place, because Hibernate will adjust the database based on changes to the objects in Java automatically. The Persistence Object represents an Object in the Java environment which is supported by the Hibernate transaction tool and eventually, the database (Bauer, 2007).

# Conclusion

Although Object Oriented Database concepts have now existed for more than 20 years, they continue to evolve over time. Object Relational Mappers provide powerful interfaces for developers to integrate relational database systems with applications quickly and easily to avoid having to develop their own database manipulation language among common use cases of CRUD operations. Determining whether to use these technologies is an important part of the development process. Hibernate and other popular ORM tools provide powerful transaction processing capability while minimizing code reuse and the need for complex SQL relationships. As these solutions continue to evolve, so does the world of database. The latest database change comes in the form of NoSQL databases, where data manipulation language is now dependent upon the application interface with the solution itself. Data is increasingly being approached from an Object-oriented perspective, no matter if the end database is relational or not. ORM tools provide a wealth of capability out of the box and remain a highly relevant option for developers seeking to quickly develop data persistent applications, utilizing complex combinations of database solutions.

# References

Bauer, C., & King, G. (2007). Hibernate in action. Greenwich, CT: Manning.

Hibernate Organization. (n.d.). Hibernate ORM. What is Object/Relational Mapping?. Retrieved from http://hibernate.org/orm/what-is-an-orm/

Signer, B. (2010). *Introduction to Databases: Object and Object-Relational Databases*. [Presentation]. Retrieved from http://vub.academia.edu/BeatSigner

Stajano, F. (1998). *A Gentle Introduction to Relational and Object Oriented Databases.* Retrieved from https://www.cl.cam.ac.uk/~fms27/db/tr-98-2.pdf