Hibernate is one of the most popular Object/Relational Mapping (ORM) framework in the Java world. It allows developers to map the object structures of normal Java classes to the relational structure of a database. With the help of an ORM framework the work to store data from object instances in memory to a persistent data store and load them back into the same object structure becomes significantly easier.

At the same time ORM solutions like Hibernate aim to abstract from the specific product used to store the data. This allows using the same Java code with different database products without the need to write code that handles the subtle differences between the supported products.

Hibernate is also a JPA provider, that means it implements the Java Persistence API (JPA). JPA is a vendor independent specification for mapping Java objects to the tables of relational databases. As another article of the Ultimate series already addresses the JPA, this article focuses on Hibernate and therefore does not use the JPA annotations but rather the Hibernate specific configuration

files.

Hibernate consists of three different components:

• Entities: The classes that are mapped by Hibernate to the tables of a relational database system are simple Java classes (Plain

Old Java Objects).

• Object-relational metadata: The information how to map the entities to the relational database is either provided by annotations (since Java 1.5) or by legacy XML-based configuration files. The information in these files is used at runtime to perform the mapping to the data store and back to the Java objects.

• Hibernate Query Language (HQL): When using Hibernate, queries send to the database do not have to be formulated in native SQL but can be specified using Hibernate’s query language. As these queries are translated at runtime into the currently used dialect of the chose product, queries formulated in HQL are independent from the SQL dialect of a specific vendor.

In this tutorial we are going through different aspects of the framework and will develop a simple Java SE application that stores and retrieves data in/from a relational database. We will use the following libraries/environments:

• maven >= 3.0 as build environment

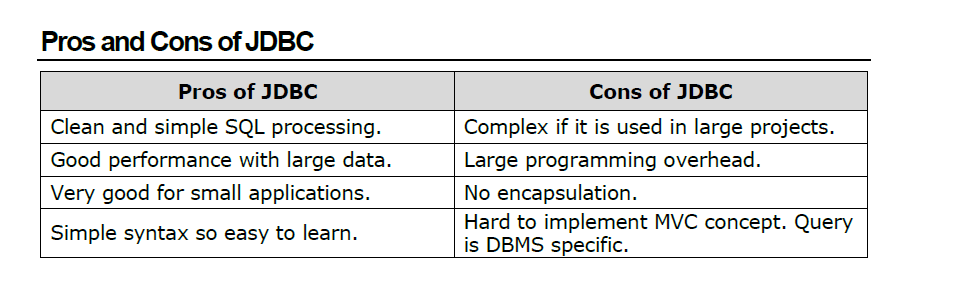
• Hibernate(4.3.8.Final)

• H2 as relational database (1.3.176)

**What is JDBC?**

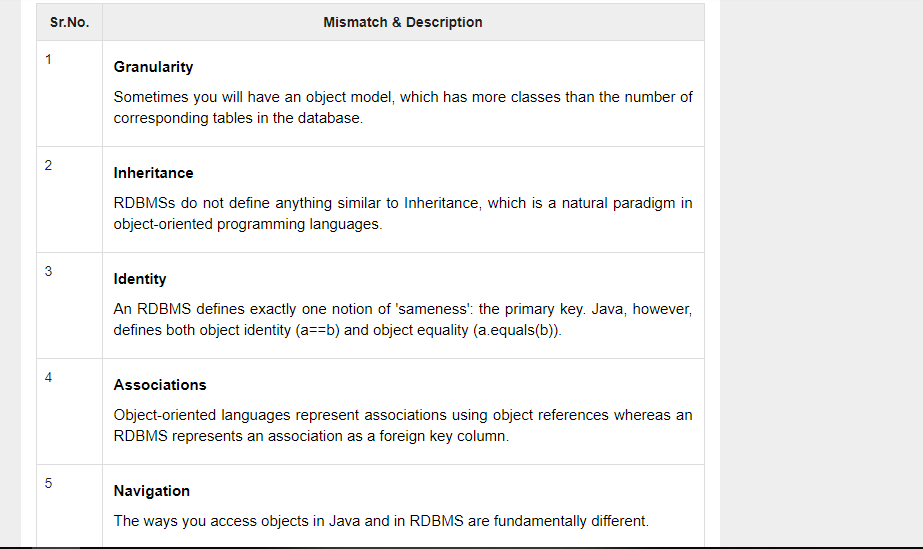
JDBC stands for **Java Database Connectivity**. It provides a set of Java API for accessing the relational databases from Java program. These Java APIs enables Java programs to execute SQL statements and interact with any SQL compliant database.

JDBC provides a flexible architecture to write a database independent application that can run on different platforms and interact with different DBMS without any modification.



**Why Object Relational Mapping (ORM)?**

When we work with an object-oriented system, there is a mismatch between the object model and the relational database table. RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java or C# represent it as an interconnected graph of objects.



Java ORM Frameworks

There are several persistent frameworks and ORM options in Java. A persistent framework is an ORM service that stores and retrieves objects into a relational database.

* Enterprise JavaBeans Entity Beans
* Java Data Objects
* Castor
* TopLink
* Spring DAO
* Hibernate
* And many more

