HIBERNATE - ORM OVERVIEW

What is JDBC?

JDBC stands for **Java Database Connectivity** and 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.

Pros and Cons of JDBC

Pros of JDBC	Cons of JDBC
• Clean and simple SQL processing • Good performance with large data • Very good for small applications • Simple syntax so easy to learn	• Complex if it is used in large projects • Large programming overhead • No encapsulation • Hard to implement MVC concept • Query is DBMS specific

Why Object Relational Mapping *ORM*?

When we work with an object-oriented systems, there's a mismatch between the object model and the relational database. RDBMSs represent data in a tabular format whereas object-oriented languages, such as Java or C# represent it as an interconnected graph of objects. Consider the following Java Class with proper constructors and associated public function:

```
public class Employee {
   private int id;
   private String first_name;
   private String last_name;
   private int salary;

   public Employee() {}
   public Employee(String fname, String lname, int salary) {
      this.first_name = fname;
      this.last_name = lname;
      this.salary = salary;
   }
   public int getId() {
      return id;
   }
   public String getFirstName() {
      return first_name;
   }
   public String getLastName() {
      return last_name;
   }
   public int getSalary() {
      return salary;
   }
}
```

Consider above objects need to be stored and retrieved into the following RDBMS table:

```
create table EMPLOYEE (
   id INT NOT NULL auto_increment,
   first_name VARCHAR(20) default NULL,
   last_name  VARCHAR(20) default NULL,
   salary     INT  default NULL,
   PRIMARY KEY (id)
);
```

First problem, what if we need to modify the design of our database after having developed few pages or our application? Second, Loading and storing objects in a relational database exposes us to the following five mismatch problems.

Mismatch	Description
Granularity	Sometimes you will have an object model which has more classes than the number of corresponding tables in the database.
Inheritance	RDBMSs do not define anything similar to Inheritance which is a natural paradigm in object-oriented programming languages.
Identity	A RDBMS defines exactly one notion of 'sameness': the primary key. Java, however, defines both object identity $a == b$ and object equality $a.equals(b)$.
Associations	Object-oriented languages represent associations using object references where as am RDBMS represents an association as a foreign key column.
Navigation	The ways you access objects in Java and in a RDBMS are fundamentally different.

The **O**bject-**R**elational **M**apping *ORM* is the solution to handle all the above impedance mismatches.

What is ORM?

ORM stands for **O**bject-**R**elational **M**apping *ORM* is a programming technique for converting data between relational databases and object oriented programming languages such as Java, C# etc. An ORM system has following advantages over plain JDBC

S.N.	Advantages
1	Lets business code access objects rather than DB tables.
2	Hides details of SQL queries from OO logic.
3	Based on JDBC 'under the hood'
4	No need to deal with the database implementation.
5	Entities based on business concepts rather than database structure.
6	Transaction management and automatic key generation.
7	Fast development of application.

An ORM solution consists of the following four entities:

S.N.	Solutions
1	An API to perform basic CRUD operations on objects of persistent classes.

2 A language or API to specify queries that refer to classes and properties of classes.

3 A configurable facility for specifying mapping metadata.

4 A technique to interact with transactional objects to perform dirty checking, lazy association fetching, and other optimization functions.

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