## Database connection

1. Hibernate

Hibernate is an **Object-Relational Mapping (ORM) solution** for JAVA. It is a powerful, high performance object/relational persistence and query service allowing to develop persistent classes following object-oriented idiom – including association, inheritance and polymorphism.

When a Java application needs to access database:

1. open connection to database,
2. convert HQL (Hibernate Query Language) statements to database specific statement,
3. receive result set
4. perform mapping of these database specific data to Java objects which are directly used by Java application.
5. close the connection.

Hibernate Architecture:

Application

Hibernate

Persistence Object

Hibernate properties

XML Mapping

Database

1. JDBC

JDBC stands for **Java Database Connectivity.** It allows developers to connect, query and update a database using the Structured Query Language. JDBC API standard provides Java developers to interact with different RDBMS and access table data through Java application without learning RDBMS details and using Database Specific JDBC Drivers.

JDBC makes the interaction with RDBMS simple and intuitive. When a Java application needs to access database:

1. open connection to database,
2. use JDBC driver to send SQL queries to database,
3. process the results that are returned,
4. close the connection.

JDBC Demonstration Architecture

Browser

Servlet

JDBC

ODBC

Data source

JDBC-ODBC bridge

**7. JDBC Vs Hibernate**

**7.1 Why is Hibernate better than JDBC**

1) Relational Persistence for JAVA

Working with both Object-Oriented software and Relational Database is complicated task with JDBC because there is mismatch between how data is represented in objects and relational database. So with JDBC, developer has to write code to map an object model's data representation to a relational data model and its corresponding database schema. Hibernate is flexible and powerful ORM solution to map Java classes to database tables. **Hibernate itself takes care of this mapping using XML files so developer does not need to write code for this.**

2) Support for Query Language

JDBC supports only native **Structured Query Language** (SQL). Developer has to find out the efficient way to access database, i.e. to select effective query from a number of queries to perform same task. Hibernate provides a powerful query language **Hibernate Query Language** (independent from type of database) that is expressed in a familiar SQL like syntax and includes full support for **polymorphic** queries. Hibernate also **supports native SQL** statements. It also selects an **effective** way to perform a database manipulation task for an application.

3) Database Dependent Code

Application using JDBC to handle persistent data (database tables) having database specific code in large amount. The code written to map table data to application objects and vice versa is actually to map table fields to object properties. As table changed or database changed then it’s essential to change object structure as well as to change code written to map table-to-object/object-to-table. Hibernate provides this mapping itself. The actual mapping between tables and application objects is done in XML files. **If there is change in Database or in any table, then the only need to change XML file properties**.

4) Maintenance Cost

With JDBC, it is developer’s responsibility to handle JDBC result set and convert it to Java objects through code to use this persistent data in application. So with JDBC, mapping between Java objects and database tables is done manually. Hibernate reduces lines of code by maintaining object-table mapping itself and returns result to application in form of Java objects. **It relieves programmer from manual handling of persistent data**, hence reducing the development time and maintenance cost.

5) Optimize Performance

Caching is retention of data, usually in application to reduce disk access. Hibernate, with Transparent Persistence, cache is set to application work space. Relational tuples are moved to this cache as a result of query. **It improves performance if client application reads same data many times for same write**. Automatic Transparent Persistence allows the developer to concentrate more on business logic rather than this application code. With JDBC, caching is maintained by hand-coding.

6) Automatic Versioning and Time Stamping

By database versioning one can be assured that the changes done by one person is not being roll backed by another one unintentionally. **Hibernate enables developer to define version type field to application**, due to this defined field Hibernate updates version field of database table every time relational tuple is updated in form of Java class object to that table. So if two users retrieve same tuple and then modify it and one user save this modified tuple to database, version is automatically updated for this tuple by Hibernate. When other user tries to save updated tuple to database then it does not allow to save it because this user does not have updated data. In JDBC there is no check that always every user has updated data. This check has to be added by the developer.

7) Open-Source, Zero-Cost Product License

Hibernate is an **open source and free to use** for both development and production deployments.

8) Enterprise-Class Reliability and Scalability

**Hibernate scales well in any environment**, no matter if use it in-house Intranet that serves hundreds of users or for mission-critical applications that serve hundreds of thousands. JDBC cannot be scaled easily.

**7.2 Disadvantages of Hibernate**

1. **Steep learning curve**.
2. Use of Hibernate is an **overhead** for the applications which are:
   1. simple and use one database that never change
   2. need to put data to database tables, no further SQL queries
   3. there are no objects which are mapped to two different tables
      1. Hibernate increases extra layers and complexity. So for these types of applications JDBC is the best choice.
3. **Support** for Hibernate on Internet is **not sufficient**.
4. For complex data, mapping from Object-to-tables and vice versa reduces performance and increases time of conversion.
5. Hibernate does not allow some type of queries which are supported by JDBC. For example, it **does not allow to insert multiple objects** (persistent data) to same table using single query. Developer has to write separate query to insert each object.