* Prior to the use of ORM, JDBC (Java DataBase Connectivity) API was introduced by Sun Micro Systems.

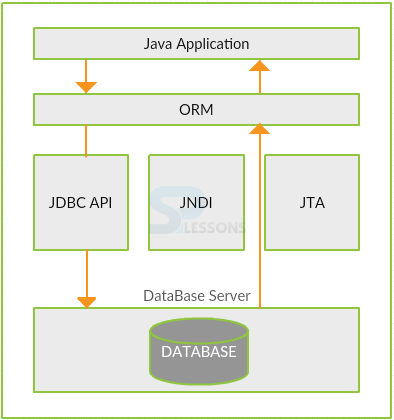
**Drawback of JDBC:**

* While opening the connection developer needs write that in **try block** and if any exception happened then that ought to be in catch block and finally block will be helpful to end the connection.
* After creating the table once if developer **change the table of values**, then needs to **redeploy tha**t.
* **SQL commands** will be **different** for **different databases.**
* **Mysql**: SELECT column\_name FROM table\_name ORDER BY column\_name ASC LIMIT 10;
* **Sql Server**: SELECT TOP 10 column\_name FROM table\_name ORDER BY column\_name ASC;

Main feature of hibernate is an ORM Which remove the flaws of JDBC

ORM(Object Relational Model)

* ORM is a mechanism which contains all the  data in the Object format.



# **Hibernate Architecture**

* ORM Software uses the JDBC Technology to perform the database operations on the DataBase Server.
* ORM Software uses JNDI when **Connection pool** technique is applied for database connections.(Java Naming and Directory interface)
* ORM Software uses JTA for managing the Transactions.

### **Hibernate:**

Hibernate is an **Open Source Framework**. Hibernate is called as **ORM tool**. Hibernate is the **persistence layer** between the Java Application and DataBase. Java Application gives **Queries** to Hibernate and then Hibernate converts queries into **Objects**. These objects are transferred to the **DataBase**.

### **Advantages of Hibernate**

* Performance will be increased.
* Fetching the data between the multiple tables.
* Automatic table creation
* Light weight
* Open source

### **Features**

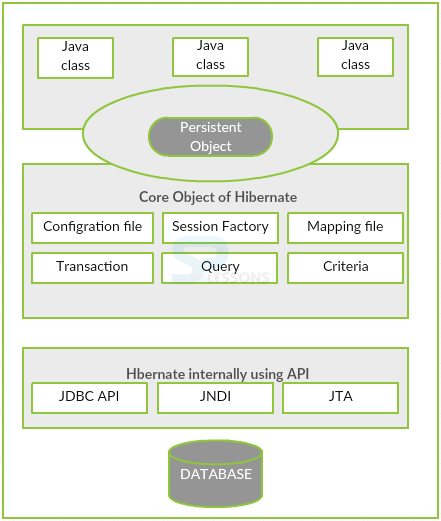
The chapter demonstrates about Hibernate Features. Hibernate has its own Query language called**Hibernate Query Language(HQL).**Hibernate Queries are Database independent. HQL contains **Database Independent Commands** so if made any changes in the database, then also it will work as Hibernate Queries. Following are some of the Hibernate Features.

* Hibernate contains**Translator**. Hibernate changes the **Checked Exception** into **Unchecked Exception**. Hibernate has the unchecked exceptions. So, no need to write the code in the try, catch and throws which is a good Hibernate Features. The programmer can’t handle the exceptions explicitly.
* Hibernate provides **Dialect classes.** So, no need to write **SQL Queries** in Hibernate. Only use the **instance methods,** then the Hibernate automatically creates the Queries in Command prompt.
* In the Hibernate, all the data will be stored in the table format. So, data is easily understandable by the user which is a good Hibernate Features.
* While inserting a record into the table if not found the particular table in the database, then JDBC throws Exception like**View not exist**, but Hibernate will create the table for us.
* Hibernate supports**Inheritance** which means if the derived class object is saved, then its base class object will also be stored in a database.
* Hibernate has **Caching mechanism**. It reduces the number of round trips between Java application to DataBase. Then Hibernate performance will increase.
* Hibernate supports **Annotations**apart from XML.
* Getting a **pagination** in Hibernate is very simple.
* Hibernate is a **light-weight Framework** because Hibernate transfers POJO class from Java Application to the Database.
* Hibernate supports Collections like List, Set, and Map. JDBC doesn’t support Collections.
* In Hibernate, there is no need to create any Connection pool object. Only Hibernate-c3po.jar is used.

Following are the disadvantages of hibernate.

* Hibernate doesn’t support **Batch Processing** i.e inserting multiple records in the database table in a single time is not possible in Hibernate.
* JDBC is supported Batch Processing.

Hibernate Architecture.



To create any Hibernate application or when any Hibernate Application is initialized, first create the Configuration Object. Hibernate can provide connection between **Java Classes** and **Database tables**. This is taken care of through different Configuration records bolstered by Hibernate. These Files are hibernated.cfg.XML, hibernate properties.

Configuration Object is utilized to create a**Session Factory Object**. It can read the all Configure Properties and provides the operations. It initializes the **Session Object**.

Session Factory is a **thread-safe object** and it is used in all thread-based applications.  
Session Factory Object is the **Heavy-Weight object.**This object is created only once and later it will be using multiple times.

The session is utilized to get a physical association with the database. Session is a Light-Weight Object. The session will be made for each database operation and Session is introduced by utilizing SessionFactory Object. By Using the Session Object, persistence items are spared and recovered. Session Object is not a string safe item. Along these lines, Session Object can close or decimate unfailingly.

A Transaction speaks to a Unit of work with a Database table operations. Transactions of Hibernate is handled by **Transaction Manager** or **JNDI** and **JTA**. Transaction object is optional. The transaction can be used in the Hibernate to perform some operations. Once a Transaction object is created, one have to commit that transaction.

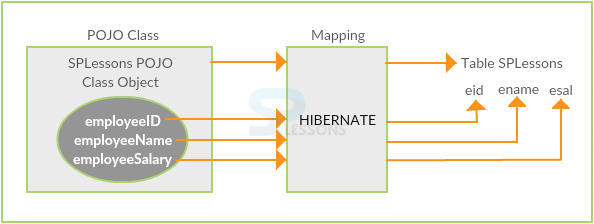
To perform any operations in SQL or Hibernate object, create **Query Object**.

Criteria Object is used only to perform **SELECT** Query.

**Hibernate Mapping**

***Hibernate Mapping*** is a system utilized to educate ORM device abo

ut recognizable proof of POJO classes put away in tables. Mapping is a procedure to outline a POJO class name and table name furthermore to delineate class variables and the table segment names.



**Classname.hbm.xml**

|  |
| --- |
| <div class="crayon-line" id="crayon-575c2fd7819be894077258-1">  <hibernate-mapping>  <class name="POJO class name" table="table name in database">  <id name="variable name" column="column name in database" type="java/hibernate type" />  <property name="variable1 name" column="column name in database" type="java/hibernate type" />  <property name="variable2 name" column="column name in database" type="java/hibernate type" />  </class>  </hibernate-mapping> |

* On constructing a mapping file in Hibernate, multiple Pojo classes are obtained.All classes are mapped in one Mapping file.
* Mapping file is like back bone to hibernate ORM tool.
* Property tag represents a Non-Primary Column.
* Mapping for hibernate may be in the form annotation or XML.

**Hibernate Configuration File**

XML file which is used to define

* Database connection details: Driver class, URL, username and Password.
* Hibernate properties: Dialect class property and show\_sql property.
* Mapping file names.
* One configuration files for each database connection.

|  |  |
| --- | --- |
| **Properties** | **Description** |
| hibernate.connection.driver\_class | The JDBC driver class. |
| hibernate.dialect | This property makes Hibernate generate the suitable SQL for the picked database. |
| hibernate.connection.url | The JDBC URL to the database instance. |
| hibernate.connection.username | The database username. |
| hibernate.connection.password | The database password. |
| hibernate.connection.pool\_size | Limits the number of connections waiting in the Hibernate database connection pool. |
| hibernate.connection.autocommit | Allows autocommit mode to be used for the JDBC connection. |

* Before XML file developers used to write hibernate configuration file in property file.