**ORM**

ORM stands for Object-Relational Mapping. It's a programming technique used to convert data between incompatible systems by creating a "virtual object database" that can be used from within the programming language. In simpler terms, ORM is a tool or library that helps developers to interact with a relational database using an object-oriented paradigm.

ORM tools offer several benefits:

1. Abstraction: Developers can work with objects in their code without needing to worry about the underlying database structure.
2. Productivity: ORM tools can generate much of the boilerplate code needed to interact with a database, reducing development time and effort.
3. Portability: Since the ORM layer abstracts the database operations, switching to a different database system becomes easier as developers only need to change the ORM configurations, rather than rewriting all the database-related code.
4. Maintenance: ORM tools often provide features like automatic schema migrations, making it easier to manage changes to the database structure over time.

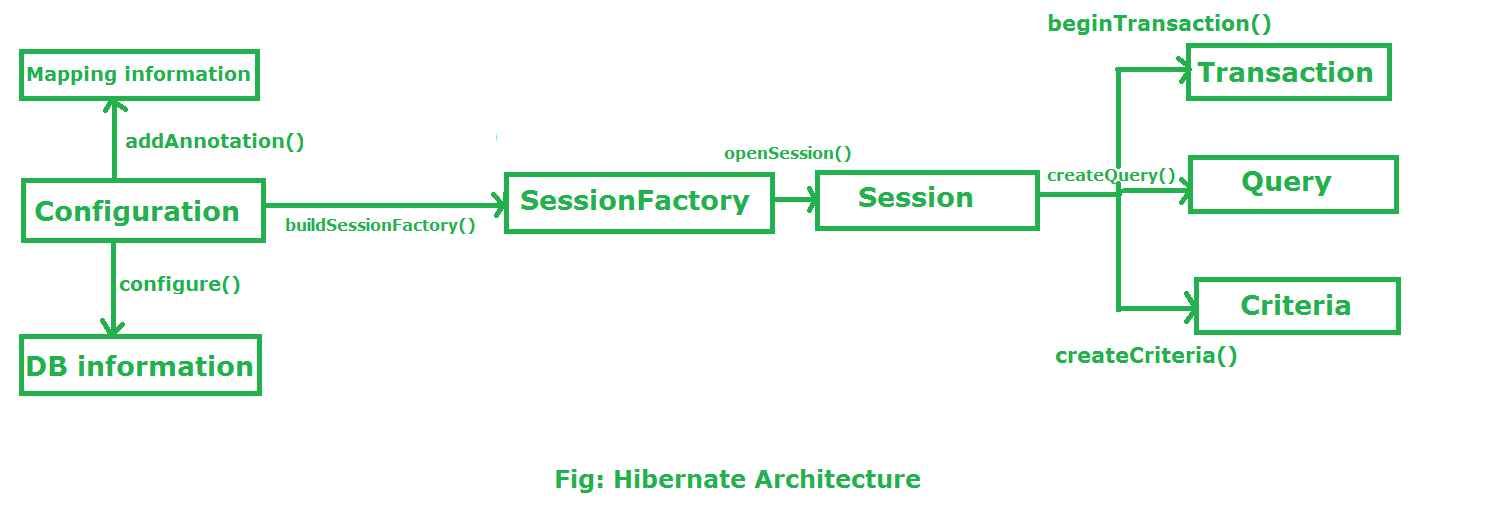
Some tools: Hibernate, EclipseLink, iBatis or MyBatis, TopLink

**Hibernate**

Most Popular ORM tool for java

**Advantages of Hibernate**

1. Open Source and Lightweight
2. Fast Performance
3. Database independent Query
4. Automatic Table creation
5. Simplified Complex Joins



**Main Components of Hibernate**

#### SessionFactory

SessionFactory is a crucial component responsible for creating and managing sessions.

1. Session Creation
2. Thread Safety
3. Database Connection Pooling
4. Configuration Management
5. First-Level Cache Management:
6. Query Compilation:

#### Session

Session represents a single-threaded unit of work between a Java application and a relational database. It serves as the main interface for developers to perform CRUD (Create, Read, Update, Delete) operations on persistent objects, as well as to execute queries against the database.

1. Transaction Management
2. Database Operations
3. Loading
4. Life cycle Management

#### Transaction

Transaction object represents a database transaction, which is a sequence of database operations that are treated as a single unit of work. Transactions ensure data consistency, integrity, and isolation in a multi-user environment by following the ACID properties:

1. Transaction Management
2. Beginning a Transaction
3. Committing a Transaction
4. Rolling Back a Transaction:

#### Configuration

Configuration class is a central component used to bootstrap and configure Hibernate within a Java application. It is responsible for reading configuration settings, mapping metadata, and building a SessionFactory

1. Configuration Settings
2. Entity Class Registration:

**CRUD OPERATIONS IN HIBERNATE**

**CREATE -** session.save(Object o);

**READ -** session.get(Object o) or session.load(Object o);

**UPDATE -** same as save but after fetching

**DELETE -** session.delete(Object o)

**Java Persistence API**

It's a Java specification for accessing, persisting, and managing data between Java objects and relational databases. JPA provides a standardized way for Java developers to interact with databases, abstracting many of the complexities involved in database access and management.

**Main Components of JPA**

**Entity Manager**

Data Persistence, Queries Execution, Transaction Management

**EntityManagerFactory**

Creation of EntityManager and Configuration

**JPA vs Hibernate vs Spring Data JPA**

**Spring Data JPA**

Property file configuration:

spring.datasource.url=jdbc:mysql://localhost:3306/emp

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.jdbc.Driver

spring.jpa.database-platform = org.hibernate.dialect.MySQL5Dialect

spring.jpa.generate-ddl=true

spring.jpa.hibernate.ddl-auto = update