**2.Hibernate XML Config implementation walk through**

**1.Object to Relational Mapping in Hibernate XML Configuration**

Hibernate uses **Object Relational Mapping (ORM)** to connect Java classes with relational database tables. When using **XML configuration**, the mapping between a Java class and a database table is defined in a separate XML file (\*.hbm.xml). This helps in separating mapping logic from business logic, which is especially useful in large or legacy systems.

The XML file defines:

* The class to map (class element)
* The corresponding table name
* The fields and their database column mappings (id, property, etc.)

**employee.hbm.xml:**

**<hibernate-mapping>**

**<class name="com.example.Employee" table="employee">**

**<id name="id" column="id">**

**<generator class="increment"/>**

**</id>**

**<property name="name" column="name"/>**

**<property name="salary" column="salary"/>**

**</class>**

**</hibernate-mapping>**

This mapping tells Hibernate how to persist the Employee object into the employee table, including how to generate IDs and which fields go into which columns.

2. **SessionFactory**

 SessionFactory is the core of Hibernate.

 It is a thread-safe, heavyweight object, created once during application startup.

 It is responsible for creating Session objects.

 Typically configured using hibernate.cfg.xml, which contains database connection info and mapping file references.

3. **Session**

 Session is a lightweight, non-thread-safe object used to interact with the database.

 It provides methods for performing CRUD operations (save, update, delete, get, etc.).

 Opened using the SessionFactory.

**4. Transaction**

 Represents a unit of work.

 Required to ensure atomicity and consistency.

 All write operations (save, delete, update) should be done inside a transaction.

5. **beginTransaction()**

 Starts a new transaction.

 Returns a Transaction object that can later be committed or rolled back.

6. **commit()**

 Commits the transaction to the database.

 All changes made during the transaction become permanent

7. **rollback()**

 If an exception occurs, the transaction should be rolled back to avoid partial updates or errors.

 rollback() reverts all operations performed during the transaction

8. **session.save()**

 Persists a **new object** to the database.

 Generates an INSERT SQL statement.

9. **session.createQuery().list()**

 Used to **execute HQL (Hibernate Query Language)** to retrieve data.

 Returns a List of objects from the database.

10. **session.get()**

 Fetches an object from the database using its primary key.

 Returns null if no record is found.

 Immediately hits the database (eager load).

11. **session.delete()**

 Deletes an object from the database.

 Must pass an entity object (usually fetched with get()).