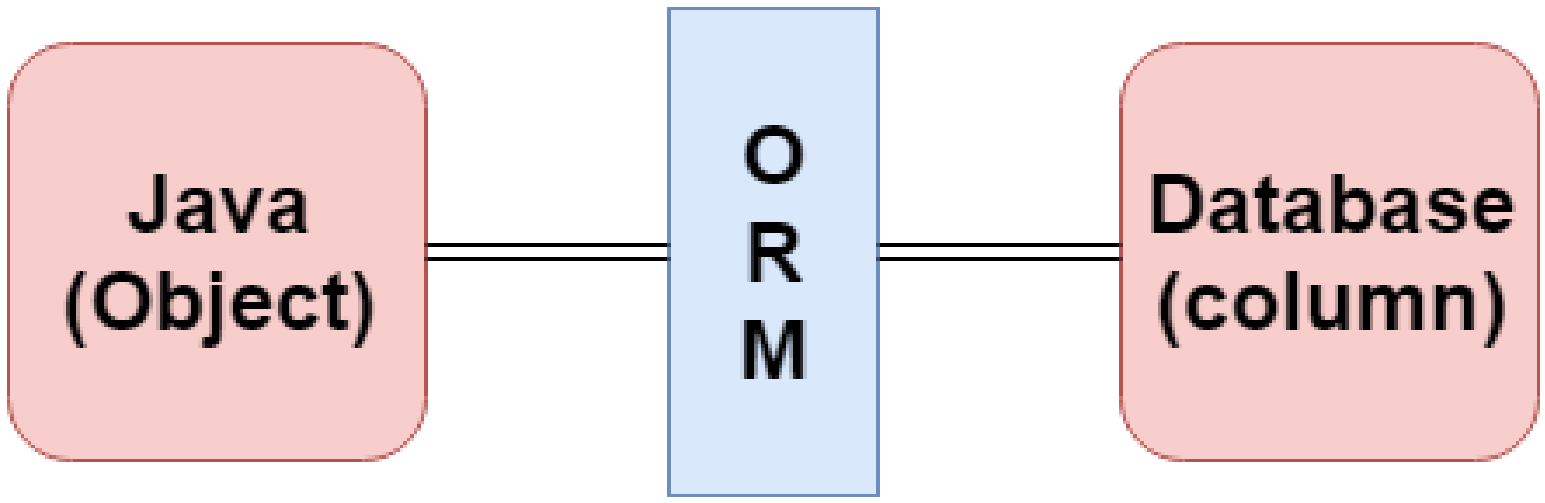
HIBERNATE

-AN ORM TOOL

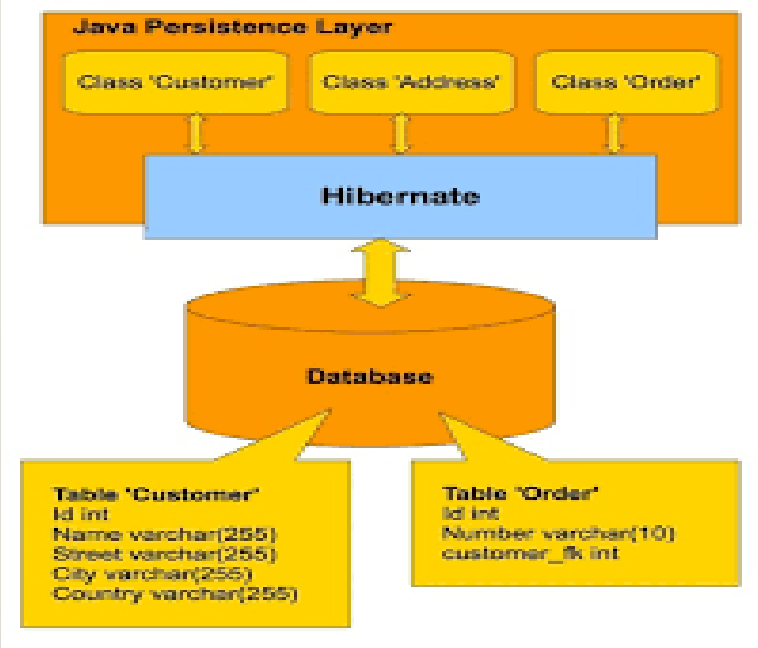
WHAT IS ORM FRAMEWORK

- ORM Framework ORM stands for **Object Relation Mapping**. It is a middleware tool that lies between the application and database. It wraps the implementation ...



HIBERNATE:-

- Hibernate is a ORM-Tool and java framework used to interact (communicate) with the database.. It is an open-source, lightweight, ORM(Object Relational Mapping) tool. Hibernate implements the specifications of JPA (Java Persistence Api) ….



ADVANTAGES OF HIBERNATE:-

1. Open source and Lightweight…
2. Fast performance….
3. Database independent Query
4. Automatic table creation
5. Simplifies complexities in developing java application.

WHAT IS ENTITY CLASS….?

- Typically, an entity class represents a table in a relational database, and each entity class object corresponds to a row in the table.

ENTITY MANAGER FACTORY

- EntityManagerFactory is an **Interface used to interact with the persistence unit to establish connection with the application and database**.
- The **EntityManagerFactory** interface present in **java.persistence** package is used to provide an object of entity manager.
- **Persistence -** The Persistence is a class which is used to obtain object of EntityManagerFactory interface.

Syntax :

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory(“String”);//string = persistence unit name.
```

ENTITYMANAGER

EntityManager interface is used **to allow applications to manage and search for entities in the relational database**. We can save, update and delete the data in database

IMPORTANT METHODS :

- persist – Make an instance managed and persistent.
- merge – Merge the state of the given entity into the current persistence context.
- remove – Remove the entity instance.
- find – Find by primary key.

Syntax :

EntityManagerFactory emf = Persistence.createEntityManagerFactory(“String”);//string = persistence unit name

EntityManager em1 = EntityManagerFactory.createEntityManager();

EntityManager em2 = EntityManagerFactory.createEntityManager();

EntityManager em3 = EntityManagerFactory.createEntityManager();

NOTE : we can create multiple entity managers for one entity manager factory.

ENTITY TRANSACTION

- IMPORTANT METHODS :
- **begin() method -** This method is used to start the transaction.
- **commit() method -** This method is used to commit the transaction.

Syntax :

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("String");//string = persistence unit name

EntityManager em1 = EntityManagerFactory.createEntityManager();

EntityTransaction et = EntityManager.getTransaction();
```

DIFFERENCE BETWEEN PERSIST AND MERGE

Persist	Merge
Persist() will only save data in the database	Merge() will save the data and update the data
If we pass the object with duplicate primary key, it will throw an exception	If we pass the object with duplicate primary key, if primary key matches in the table it update the data , if it does not matches it inserts the data

MAPPING IN HIBERNATE

- The ***mapping*** between entity classes and the relationships between tables is the soul of ORM.
- hibernate mappings are one of the key features of hibernate. they establish the relationship between two database tables.
- you can establish either unidirectional or bidirectional..

DIFFERENT MAPPING ANNOTATIONS ARE :

- @OneToOne(uni-direction)
- @OneToMany(uni-direction)
- @ManyToOne(uni-direction)
- @ManyToMany(uni-direction)
- @OneToOne(bi-direction)
- @OneToMany(bi-direction)
- @ManyToMany(bi-direction)

ONE-TO-ONE MAPPING

- One to one represents that a single entity is associated with a single instance of the other entity. An instance of a source entity can be at most mapped to one instance of the target entity.
- In simple words, we can say one row of one table is associated with one row of another table
- In this type of mapping one entity has a property or a column that references to a property or a column in the target entity.
- Ex : One person has one pan, a pan is associated with a single person.

ONE TO MANY MAPPING

- In one-to-many mapping one entity is associated with many instances of other entity. for example one person has many bank accounts.
- In simple words, one row of one table is associated with multiple of another table
- Here you can retrive the data in uni-direction .If you have person id you can get all the account details and you cannot get person details with account id.

Ex : One person has many accounts, many accounts are associated with a single person.

MANY-TO-ONE MAPPING

- In many-to-one mapping many instances of entity is associated with one instance of other entity. for example many branches have one hospital.
- In simple words, we can say that many rows of one table are associated with one row of another table
- Here you can retrive the data in uni-direction .If you have branch id you can get the hospital details and you cannot get branch details with hospital id.

Ex : Many branches are associated with one hospital.

MANY-TO-MANY MAPPING

- In many-to-many mapping many instances of entity is associated with many instances of other entity. for example many students have many courses.
- In simple words, many rows of one table is associated with another table..

Ex : Many students are associated with many courses.

CASCADING IN HIBERNATE

- Cascading is a feature in Hibernate, which is **used to manage the state of the mapped entity whenever the state of its relationship owner (superclass) affected**. When the relationship owner (superclass) is saved/ deleted, then the mapped entity associated with it should also be saved/ deleted automatically.
- **When we perform some action on the target entity, the same action will be applied to the associated entity.**

TYPES OF CASCADING

- All JPA-specific cascade operations are represented by the javax.persistence.CascadeType enum containing entries:
- ALL
- PERSIST
- MERGE
- REMOVE

CACHING IN HIBERNATE

- Hibernate caching improves the performance of the application by saving the object in the cache. It is useful when we have to fetch the same data multiple times.
- There are mainly two types of caching:
  - First Level Cache, and
  - Second Level Cache

SECOND LEVEL CACHING

- SessionFactory object holds the second level cache data. The data stored in the second level cache will be available to entire application. But we need to enable it explicitely.
- Three steps to enable second level cache :
- 1.Add hibernate-echache dependency . (version should be same as hibernate version)
- 2.Add @cacheble annotation to Entity class.
- 3.Enable second-level cache by configuring in persistence.xml file.