Hibernate

(28) What is ORM Framework ?

ORM stands for Object-Relational Mapping (ORM) is a programming technique for converting data between relational database and object Oriented programming languages such as java, c# etc.

what does an ORM Framework do ?

Basically, the ORM Framework / software that generates objects (as in oops) virtually map (like the map of a city) the table in a database. Then you as a programmer, could use these objects to interact with the database. So the main idea, is to try and shield the programmer from having to write optimized SQL code - the ORM generated objects take care of that for you. ~~So let's take a look at a simple example.~~

Advantages

① A relational database use data types for each of the fields,

Ex:— int, small int, blob, char etc.
The thing is that Sometimes you have to Convert the data types of the fly to properly add a second to the database, A good ORM will take Care of these details.

② This makes it easier to write and debug your application, esspecially if you have more programmers on a job.

③ ORM Framework will shield your application from SQL injection attacks since the Framework will be filtering the data for you.

④ (29) List few of the ORM Framework?

① Apache Cayenne - open Source for java
② Java Data Objects (JDO)
③ Hibernate - open Source ORM Framework
④ Enterprise Objects Framework

30 What is Hibernate ?

Hibernate is a Java Framework that simplifies the development of Java application to interact with the database. It is an open source, & ORM (object Relational Mapping) tool. Hibernate implements the Specifications of JPA (Java persistence API) for data persistance.

What is JPA

Java persistence API (JPA) is a Java Specification that provides certain functionality and standard to ORM.

The javax.persistence package contains the JPA classes and interfaces. (It give our mapping & cascading all these thing given by JPA)

Advantages of Hibernate

1. Fast performance. The performance of the hibernate framework is fast because Cache is internally used in hibernate framework.
2. Database Independent Query.
3. Automatic table Creation — Hibernate framework provides the facility to create the tables of the database automatically, so there

is no need to Create tables in the database manually.

(31) What is entity Manager factory?

Interface
EntityManagerFactory → Creating object

① Creating an entity manager factory object.

The EntityManagerFactory interface present in Javax.persistence package is used to provide entity managers.

Interface
EntityManagerFactory is responsible to create a object for the interface

emf = Persistence.createEntityManagerFactory("student_details");

→ persistence → The persistence is a bootstrap class which is used to obtain an EntityManager factory interface.

→ CreateEntityManagerFactory() method —

The role of this method is to Create and return an EntityManagerFactory for the named persistence.xml. This method contains the name of the persistence unit passed in the persistence.xml file

↓

url, username, password.

② Obtaining an entity manager from factory.

```
EntityManager em = emf.createEntityManager();
```

(33) → EntityManager – An EntityManager is an interface.

→ createEntityManager() method –
It creates new application-managed EntityManager. → (Creating statement - inserting (deleting, save, update)

③ Initializing an entity manager.

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

→ getTranscation() – perform Transcation.
(34) This method returns the resource-level EntityTransaction object.

```
et.begin();
```

→ begin() – This method is used to start the Transaction.

④ persisting a data into relational database.

→ persist(); – This method is used to make an instance managed and persistent. A entity instance is passed with this method. (Save the object).

```
em.persist(object reference);
```

(5) closing the transaction

et.Commit();

Commit() - It will close the transaction.

(35) what is the difference between persist & merge.

persist(Object entity) :- Make an instance managed and persistent. It takes a parameter of type Object and returns void.

→ After the persist operation executed successfully

a) a new row would be inserted in to the corresponding table in database (save the data)

b) merge(T entity) —

Merge the state of the given entity in to Current persistence Context.

→ After the ~~persist~~ merge operation executed successfully

a) a new row would be inserted or an existing one would be updated.

b) the return object is different than the passed object.

c) If the passed object is unmanaged, it says unmanaged.

→ it will update the data through the existing id.

→ If the id is already present then whatever you have modified, after you can see that modification in your data base.

→ If it is not there it return not match. a new entry (in the service id will create a new data inside your database).

→ whenever you are going to update an existing data you should use merge().

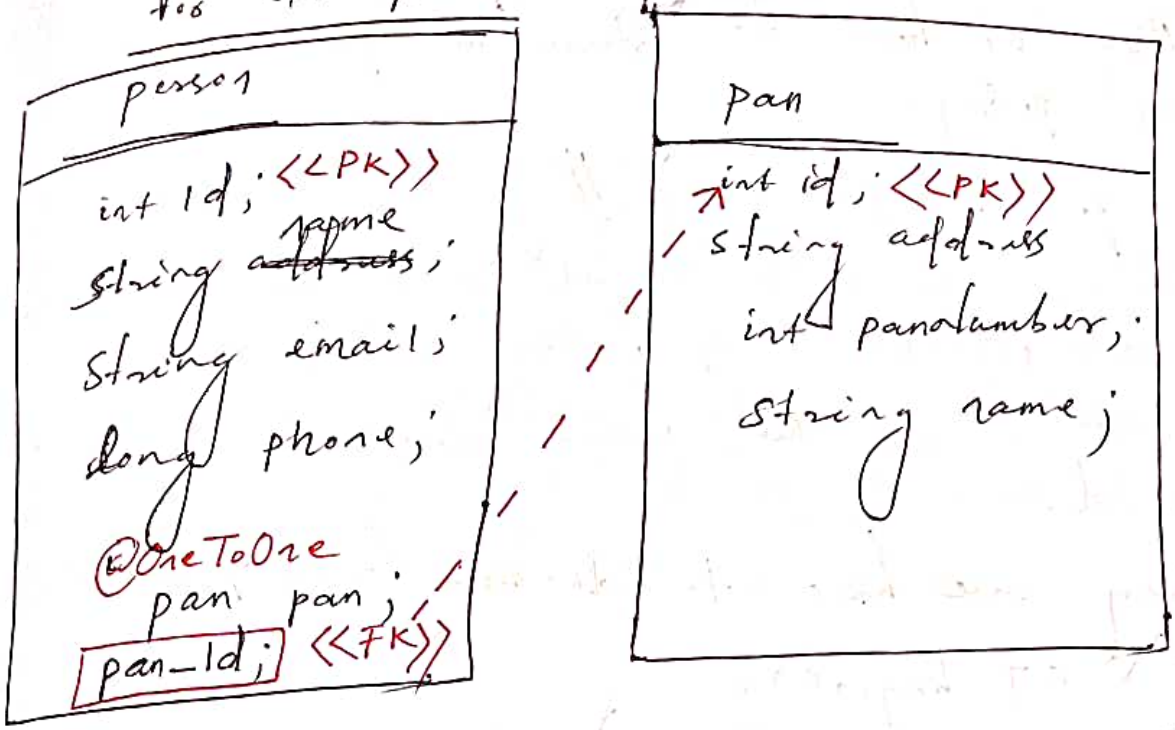
(36) whatis Mapping in hibernate ?

It is used to build the relationship between two database tables.

(37) Explain one-to-one-uni direction with example.

one-to-one unidirectional, (one table has a foreign key column that (2 references) the primary key of associated (1 table. By unidirectional relationship means only one side navigation is possible.

For example



Here one person can have one pan id.

Here first create person object and set the person attributes.

Such that → [To save data in database]

```
Person person = new Person();
person.setName("radhu");
person.setEmail("radhu@gmail.com");
person.setPhone(7328032890);
```

then create pan object and set the pan attributes.

```
Pan pan = new Pan();
pan.setPanNumber(438);
pan.setAddress("bangalore layout");
```

they we have to store the pan object in person.

person.setPan(pan);//

Note - its unidirectional so we can get the pan details using person id but we cannot get the person details by using pan id.

They we have to do our Crud operation like

```
eT.begin();
eM.persist(person);
eM.persist(pan);
eT.commit();
```

person

<<PK>> id	email	name	phone	pan_id <<FK>>
1	madhu@gmail.com	madhu	7325032390	1

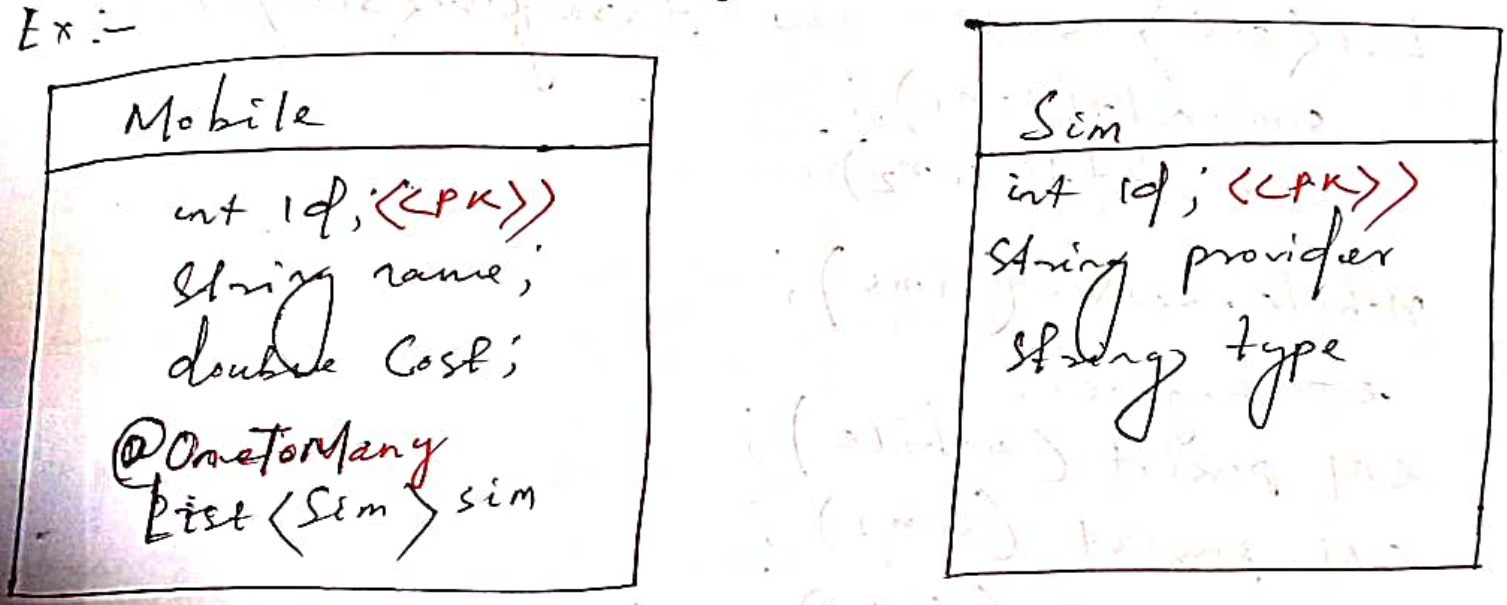
pan

<<PK>> id	address	name	panNumber
1	bangalore layout	ABY426	7328

(38) Explain one-to-many unidirectional with example.

In one-to-many relationship using a foreign key Column or the table that represents ~~the~~ to-many side. When we map this association as a unidirectional one-to-many association with Hibernate, it uses a ~~join~~ table instead of the foreign key column.

Ex:-



Here One Mobile Can have many Sim.

```
Mobile mobile = new Mobile();
mobile.setName("oppo");
mobile.setCost(30000);
```

we Can Create (n) number of Sim, So one mobile Can have multiple Sim.

```
Sim sim1 = new Sim();
sim1.setProvider("JIO");
sim1.setType("4G");
Sim sim2 = new Sim();
sim2.setProvider("Airtel");
sim2.setType("3G");
```

We can create many sims so mehna lance list so,

```
List<sim> sims = new ArrayList<sim>();
sims.add(sim1);
sims.add(sim2);

mobile.setSim(sims);
et.begin();
em.persist(mobile);
em.persist(sim1);
em.persist(sim2);
et.commit();
```

Mobile

id	cost	name
1	20000	oppo

sim

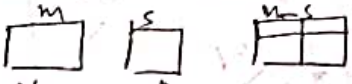
id	provider	type
1	BSNL	3G
2	Airtel	4G

Mobile_id	sim_id
1	2
1	3

(39) what are the different mapping in hibernate.?

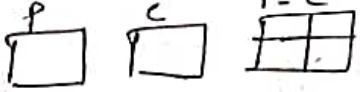
1- One-to-One - (2 tables)

2- One-to-many



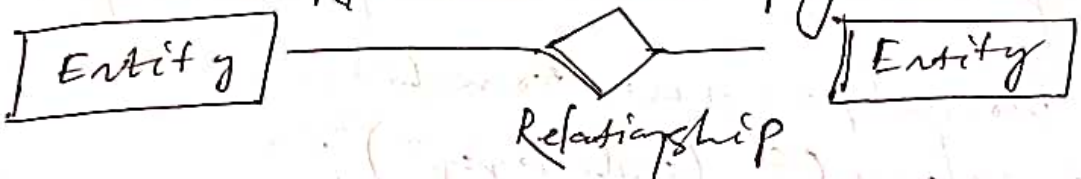
3- Many-to-one

4- Many-to-many

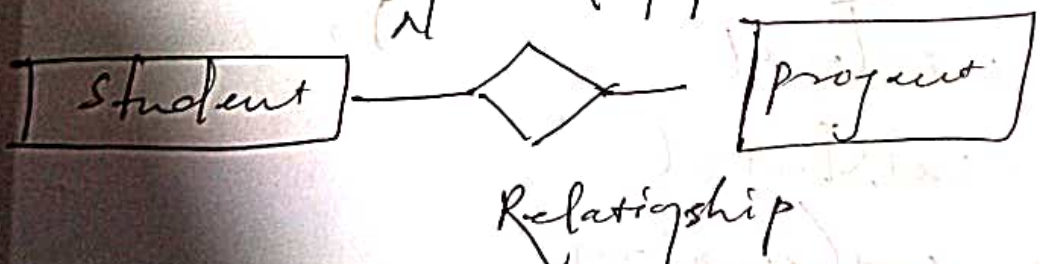


(40) Explain Many-to-one unidirectional with example.

Many-to-one relationship between more than one instance of an entity with one instance of another entity.



A project can have more than one Student working on it. A team of five students in a College is assigned a project that they need to Complete in Let us say two month.



~~Hospital~~

Hospital

int id, <PK>

String name;

String web;

Branch

Branch

int id; <PK>

String name;

String state,

String Country.

@ManyToOne

Hospital hospital

hospital_id <FK>

Hence Many branch have Certain One hospital.

```
Branch branch1 = new Branch();
branch1.setName("Cardiology");
branch1.setState("Odisha");
branch1.setCountry("India");
branch1.setPhone("7328032390");

Branch branch2 = new Branch();
branch2.setName(
branch2.setState(
branch2.setCountry(
branch2.setPhone(
```

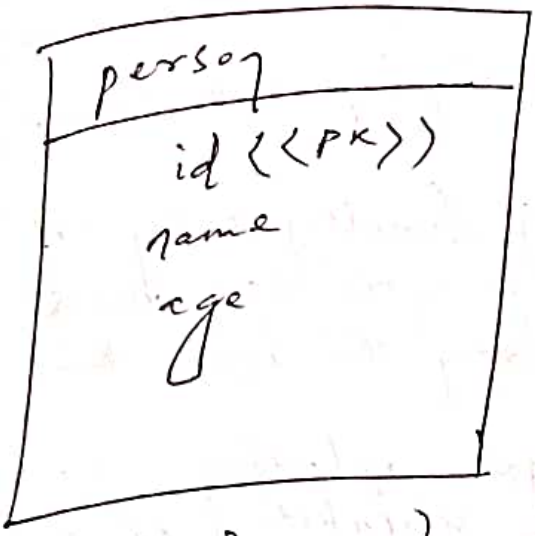
```
Branch branch2 = new Branch();
branch3.setName(
branch3.setCountry(
branch3.setState(
branch3.setPhone(

Hospital hospital = new Hospital();
hospital.setName("Ramaya hospital");
hospital.setWeb("ramaya.his.www.com");

branch1.setHospital(hospital);
branch2.setHospital(hospital);
branch2.setHospital(hospital);

et.begin();
em.persist(hospital);
em.persist(branch1);
em.persist(branch2);
em.persist(branch3);
et.commit();
```

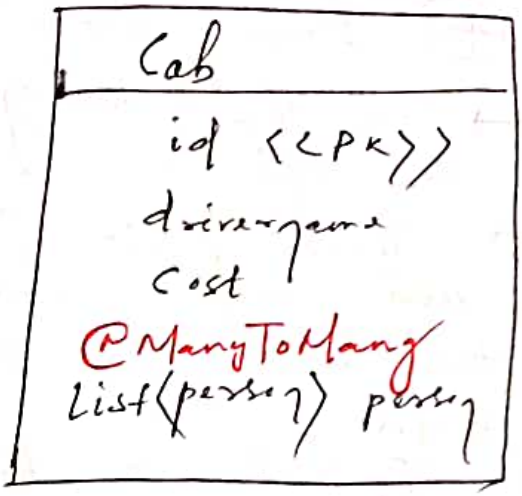
(41) Explain query parameters in JPQL with example.

(42) Explain Many-to-Many unidirection. with example.

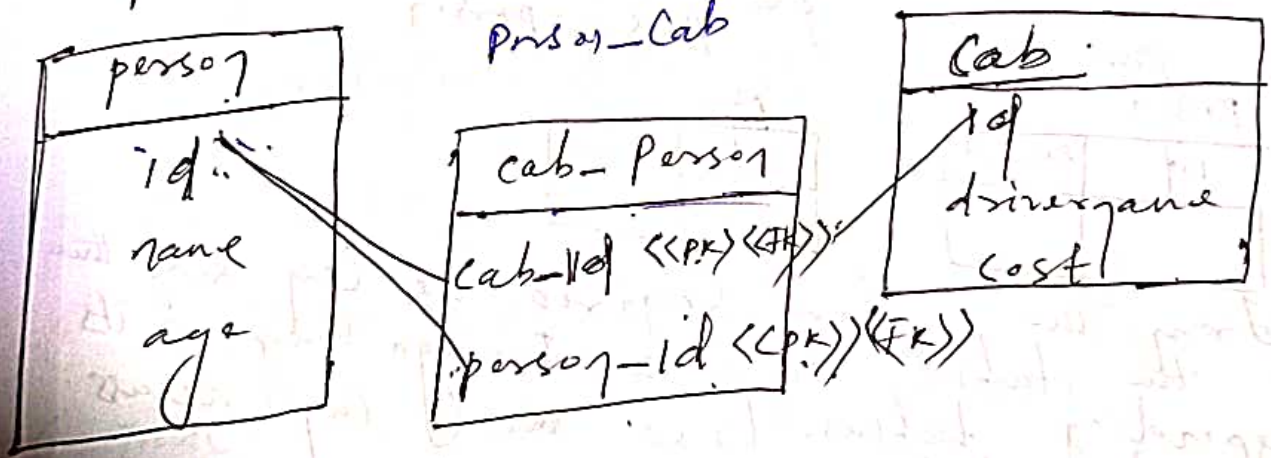
In Many-to-Many mapping, an extra table is used (known as joined table) whose primary key is the combination of primary key of both the associated tables

In other word there is a foreign key association ~~tables~~. between the joined table and ~~the~~ associated tables.





we are discussing an example of person and Cab relationship. A person can book multiple Cab. And a Cab booked by multiple person. we are considering unidirectional mapping, means only person to Cab entity navigation is possible.



(43) Explain OneToOne bidirectional with example.

→ one object having the relationship with another one object.

→ There is no change in functionality of uni and bi but in bi we can access the both the objects from the both the sides.

→ To achieve bidirectional relation we have to create the referenced variable of one object type in to another class for both the class.

→ For OneToOne we have to use the same annotation in the both class object of variable.

example

```
class person                    class pan
{ @Id                           @Id
  int id;                       int id;
@OneToOne                     @OneToOne
  pan pan;                    } person person;
}
```

person

id	pan_id

pan

id	person_id

→ From the above Scenario we can say that both the table owns the foreign key of its corresponding tables, so we can access both side tables.

→ But in above Scenario we have data duplication which can be avoid by using a) @JoinColumn – (owning side which table owns the foreign key)

b) @mappedBy – (which is not own the foreign key (variable))

→ avoid data duplication we have to decide which entity should owns and which entity should referral.

(44) what is the different between @mappedBy & @joinColumn

~~MappedBy~~ :- (owning side)

→ Remove the duplication in bidirectional, for that reason we are using these two things.

@JoinColumn Annotation

→ In a One-to-Many / Many-to-One relationship. the owning side is usually defined on the 'many' side of the relationship; its usually the which owns the foreign key.

Ex :-
```
@Entity
public class Email
    @Id
    @GeneratedValue(strategy = GenerationType)
    private Long id;
    @ManyToOne(Feth = FethType.LAZY)
    @JoinColumn(name = "employee_id")
    private Employee employee;
```

NOTE -

It simply means that our Email entity will have a foreign key/column named employee_id referring to the primary attribute id of our Employee entity.

mappedBy Attributes

Once we have defined the owning side of the relationship. Hibernate already has all the information it needs to map that relationship in our database. To make this association bidirectional, all we'll have to do is to define the referencing side. The inverse (or the referencing side simply maps to the owning side.

```
@Entity
public class Employee
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private Long id;
@OneToMany(fetch = FetchType.LAZY, mappedBy
                                    = "employee")
private List<Email> emails;
```

(45) OneToMany bidirectional with example.

* one object having the relationship with many objects.
* In OneToMany bidirectional we can fetch the object from both the sides.
* Hence we can achieve bidirectional relation two ways.
* In first case we have to mention OneToMany annotation where the class which object having multiple relationship with other class objects.
* In other class where the multiple objects also containing relationship with same object we have to mention @ManyToOne.

mobile
@Id
id
[@OneToMany( )
@JoinColumn(mappedBy="mobile")
List<sim> sim;

sim
@Id
id
@ManyToOne
@JoinColumn
mobile mobile;

mobile

id

sim

id	mobileId

46

ManyToMany bidirectional with example?

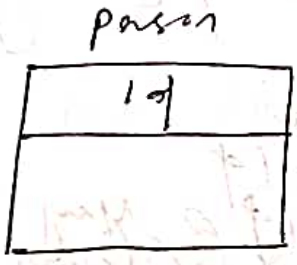
→ Many object having the relationship with many object.

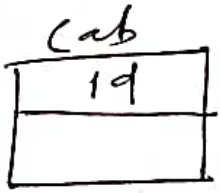
→ In this mapping we can access both the sides.

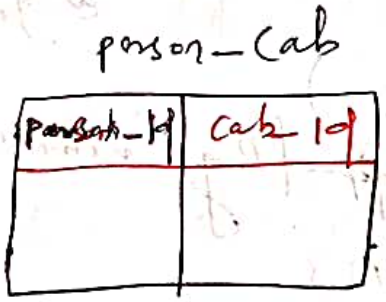
→ There is a foreign key associating both the join table & the associated tables.

person
@Id
@ManyToMany(@JoinTable)
@JoinColumn
List<Cab> cab

cab
@Id
id
@ManyToMany(mappedBy
person person = a cab







(47) what is @JoinTable explain it with an example.

→ In hibernate, @JoinTable is an annotation in javax.persistence package.

→ where we are using mapping in relationship between the table in data, sometime the duplicate tables get created.

→ In that case if you want to

(48) what is Cascading and list each cascading type.

→ It is going to build the exact relationship between the parent and child entity.

→ We are using cascading inside the mapping annotation.

→ It is present in javax.persistence.

we have many types of Cascading and few Cascading is those —

(1) ALL
(2) PERSIST
(3) MERGE
(4) REMOVE
(5) REFRESH
(6) DETACH

(1) @OneToOne(cascadeTy = CadecadeType.ALL) :-

→ It will accept all type of cmd operation.

(2) @OneToOne(cascade = CascadeType.PERSIST) :-

→ Any modification can done in your parent class with respect to save, your child also get affected.

(3) @OneToOne(cascade = CascadeType.REMOVE)

→ If you are removing data in your parent table then child table data also deleted and no need to delete child table it will do automatically.

(4) @OneToOne(cascade = CascadeType.MERGE)

→ If you are update any data in your data base you can use MERGE & it will affect your chicd also.

(99) Explain Cascading with an example?

* @Entity
public class Person {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int id;
private String name;
@OneToMany(mappedBy = "person", cascade = CascadeType.ALL)
private List<Address> address;

* @Entity
public class Address {
@Id
@GeneratedValue(strategy = GenerationType.AUTO)
private int id;
private String street;
private int houseNumber;
private String city;
private int zipCode;
@ManyToOne(fetch = FetchType.LAZY)
private Person person;

(50) Explain FetchType in hibernate with JPA with example ?

→ FetchType defines when Hibernate gets the related entities from the database.

→ getting the data.

→ FetchType decides whether or not to load all the data belongs to associations, as soon as you fetch data from parent table

(51) List different FetchType ?

(52) → FetchType is of 2 types

1 — Eager — FetchType.LAZY

2 — Lazy — FetchType.EAGER

① FetchType.LAZY → (~~Automatic~~)

If fetches the child entities lazily, that is at the time of fetching parent entity it just fetches proxy of the child entities and when you access any property of child entity then it is actually fetched by hibernate. // only on demand it will fetch the data.

② FetchType.EAGER (Automatic)

→ It fetches the child entities along with parent.

→ Lazy improves performance by avoiding unnecessary ~~implementation~~ consumption & reduce memory requirements.

→ Eager takes more memory consumption and process speed is slow.

Bidirectional

→ OneToOne - by default - ~~Lazy~~ Eager

→ OneToMany - by default - Lazy

→ ManyToOne - Eager (by default)

→ ManyToMany - Lazy (by default)

depend on table.

(53) What is Composite key?

It is the Combination of primary key.

→ We can create a Composite key by using @Embeddable annotation

→ We use @Embeddable annotation in entity containing the information of primary key.

Rules

→ The Composite key class should be public

→ The Composite key class must implement the 'Serializable' interface.

→ Constructor is mandatory

→ it should override equals() & hashcod() -

@Embeddable

UserId implements Serializable

```
email;
phone;
```

User

```
@EmbeddedId
UserId userId;
```

Some Annotation

@Entity → To make our class as table. To Create a table in our class we have to use @Entity.

@Id → we are telling to hibernate that make our variable as primary key.

@GeneratedValue = strategy = GenerationType.AUTO → I want to make auto increment, so to need to Create Id, hibernate will Create Id automatically.

@Table — In table we can modify the table like, name = " "

@unique = "true".

& default = false (duplicate value can allow in your database)

(43) ---->

* In bidirectional we can make a Object as parent and other as referred.
* parent or Owning object contains the foreign key which other object refers the variable.

ex

```
class person
{
  @Id
  int id;
  [@OneToOne
  @JoinColumn]
} Pan pan;
```

```
class pan
{
  @Id
  int id;
  @OneToOne(mappedBy=
  person person;
}
```

person

id	pan_id

pan

id

* From the above scenario we can avoid the duplicating as well as we can achieve bidirectional relationship ($A \rightleftharpoons B$).

(54) How to build a Composite key in hibernate explain with an example.

```
@Embeddable
Class UserId
    email
    phone

Class User
{
@EmbeddedId
UserId userId;
```

(55) What is ~~@EmbeddedId~~ @EmbeddedId

→ It is used for Composite primary key. ie, more than one column behaves jointly as primary key. we need to create an entity as Embeddable and Embeddable entity can be embedded in other entity to achieve Composite primary key.

(56) what is @Embeddable.

Ex:

The @Embedded annotation is used for specify the Address entity should be store in the student table as a Component. @Embeddable annotation is used for specify the Address class will be used as a Component. The Address class can not have a primary key of its own, it uses the enclosing class primary key

(57) what is persistence.xml file.

persistence.xml file must define a persistence-unit with a unique name

→ It is a standard configuration file.

→ it has be added in META-INF folder in src/main/resources folder.

(58) what is persistence unit'.?

The persistence unit defines all the meta data required to an EntityManager factory, like entity mapping, data source and JPA & transaction settings, as well as JPA provides configuring properties.

(47) what is @JoinTable, Explain it in a example?

→ In hibernate, @JoinTable is an annotation in javax.persistence package.

→ where we are using mapping in relationship between the table in database, Some time the duplicate tables gets Created.

→ In Case, if you want to avoid the duplication of table, you Can use @JoinTable.

For example :—

Let us Consider Student & Course as an entities and build a manyToMany bidirectional relationship between them.

Student

```
Id
@ManyToMany
List<Course> courses;
```

Course

```
Id
@ManyToMany
List<Student> students.
```

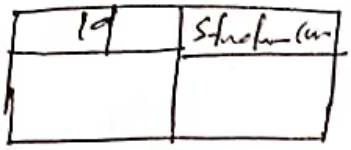
Table in database Created.

Here 4 table will created.

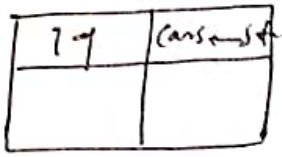
Student

Course

student_course



Course_Student



Here duplicate gets created so we can use @JoinTable on the owningside & mappedby on the defining side.

Student

Id

@ManyToMany

(mappedBy = "student")

List<Course> course;

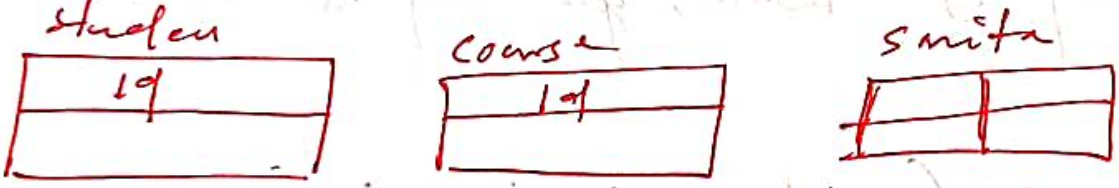
Course

Id

@ManyToMany

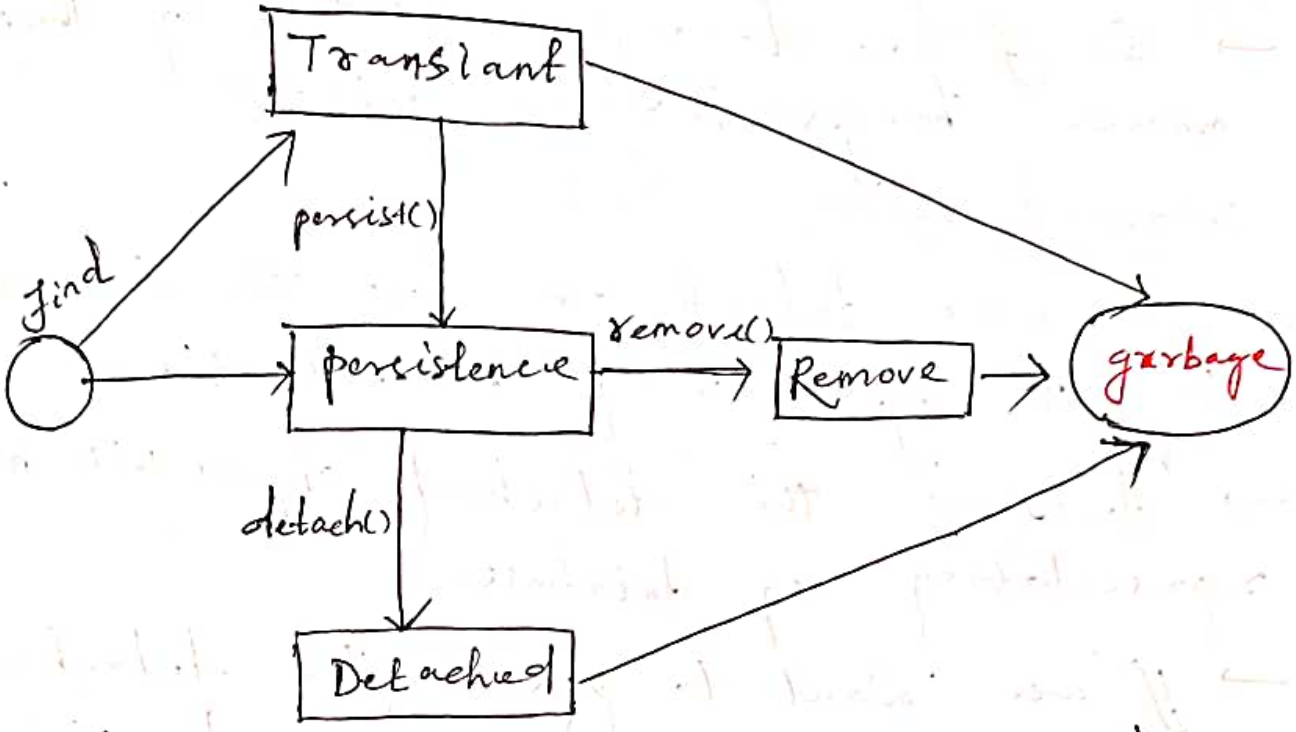
@JoinTable(name = "smi

joinColumns;



→ only 3 table will Created.

(59) Explain Entity lifecycle in Hibernate ?



→ In hibernate every entity is associated with the life cycle.

Transiant state

→ when we Create any object of entity it is in transiant state.

→ It is not related to database.

→ Hence, modification in the data ~~don't~~ it does not affect any changes in database.

→ The transiant object is independent.

persistent

→ After Creating any object, when we want to do any Crud operation, like save, remove, find all are done in persistence state.

→ The object will be in Connecting database
→ So if we do any modification in that
make changes to our database.

Detached state

→ When we detach or close the Connection
then the object is in detached state.
→ However the detached object will have
representation in database.
→ If we want to persist the detached
object again, then we have to use
load(), merge(), update() etc.
→ once we close the Connection, it will collected
in garbage collection.
→ We can use following method to detach
the object.
close(), detach(), clear(), evict().

(60) what is Caching ?

→ Caching is a mechanism to enhance
the performance of our System.
→ In hibernate we have 2 types of
Caching. 1) First level (by default)
2) Second level

(c1) Explain hibernate caching?

→ Caching is a mechanism provided by ORM framework which helps users to get fast running application.

→ Reduce no of queries made to database in a single transaction.

→ Hibernate utilize multilevel caching systems as
  * First level
  * Second level

First-level

→ In hibernate first level Caching is enabled by default

→ We can not disable it

→ For every entityManager its own first level cache will be enabled

→ EntityManagerFactory have entityManager

→ EntityManagerFactory can have n, number of entityManagers.

→ When we find first object, the entitymanager will request the database & database will give the data to entitfyManger, now entitymanger first it will save the data to the first level cache

→ after saving to the first level cache, & this object will retrive to the column.

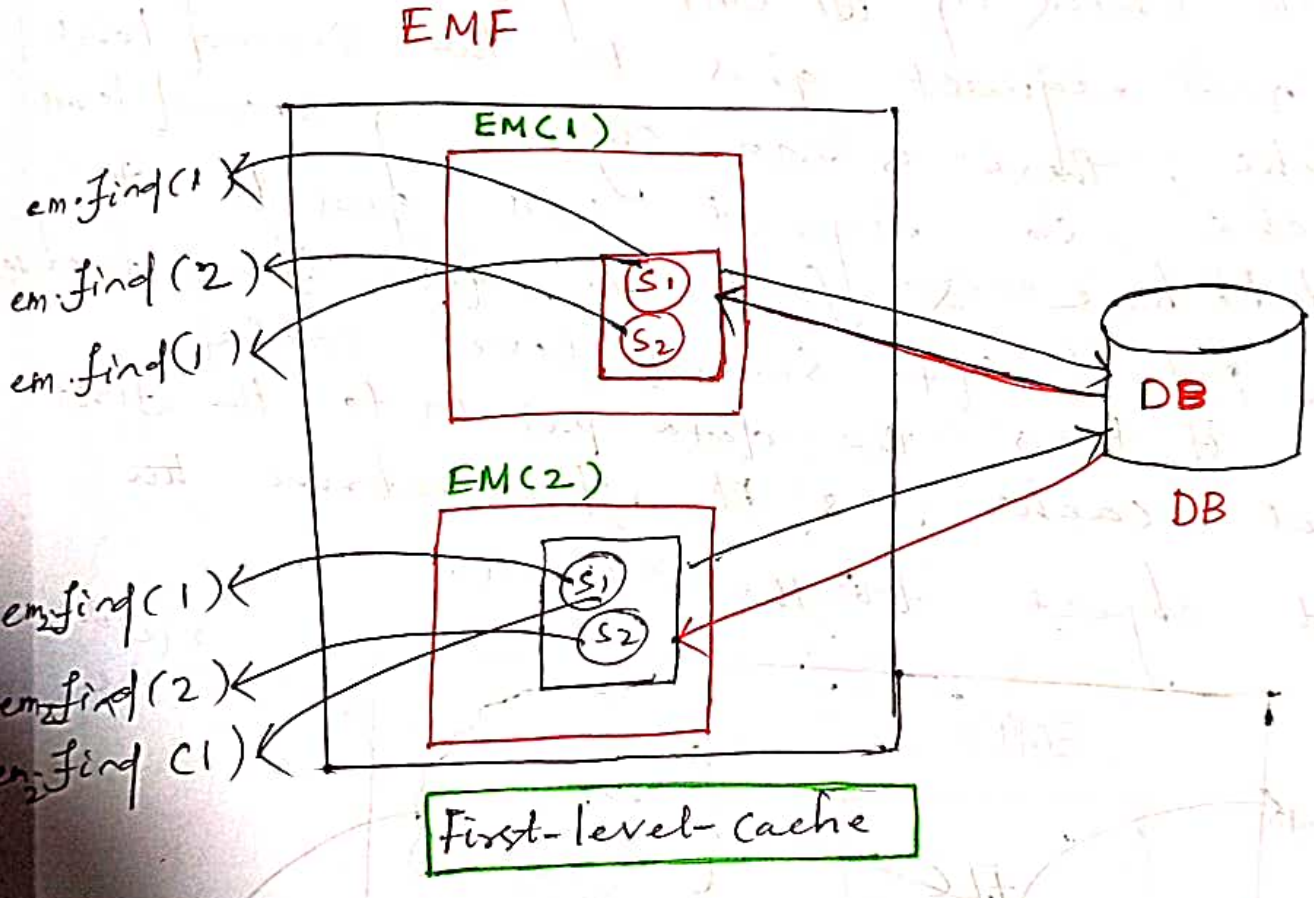
→ & after when we find 2nd object, entitymanager will check, is there any object that particular same in the cache ~~data~~ If data is not there in the cache, request will go to the database, now database will send the data to the entitymanager & entitymanager will store in the cache & 2nd object will retrive to the column.

→ If one object is already store in to the entitymanager & you are going find the same object again then, it already present so it will send the data to the respective Column.

→ So for the 2nd time you want to fetch the same object they no need to request database, you will get from the first level cache, Hence it reduces no. of hits to database.

Hibernate Caching

- First level caching (default)
  - no need to do any modification, hibernate automatically created.
- Second level caching



Second level cache

→ Second level cache is not enabled by default, we have to enable it explicitly.

→ So we need a third party API to do second level cache. (eh cache).

→ When we find first object, first it will check in EM1 in first level cache & now request goes to the second level cache, there is no object in second level cache, so request will goes to the database & fetch the data from database & put it in to second level cache & they it takes the data put in to the first level cache, & after retrive the first object to the column.