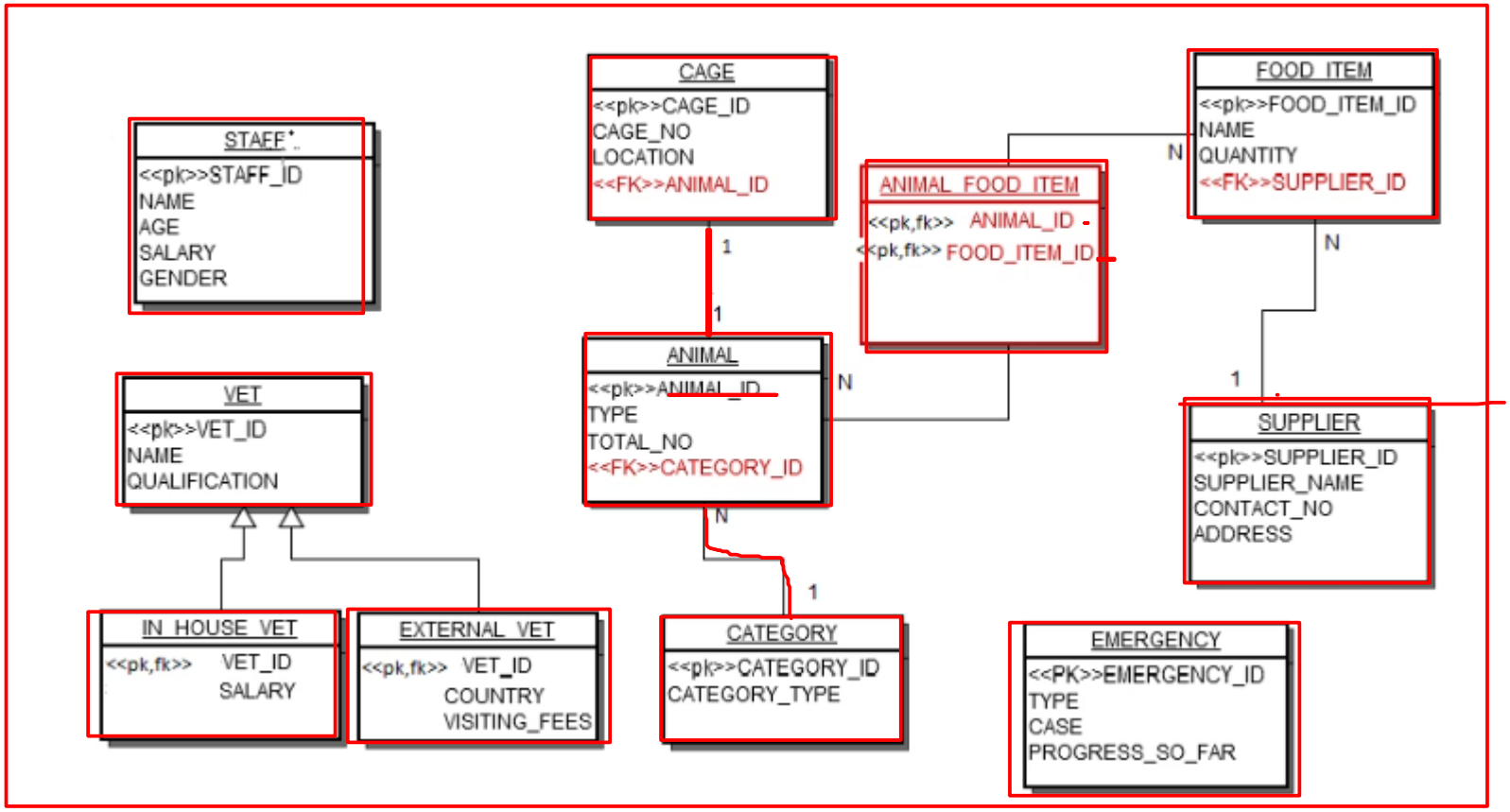
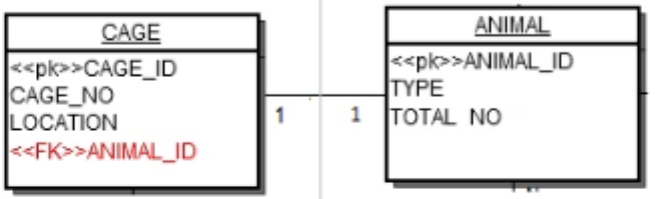

STAFF
<<pk>>STAFF_ID
NAME
AGE
SALARY
GENDER
CAGE
<<pk>>CAGE_ID
CAGE_NO
LOCATION
<<FK>>ANIMAL_ID
FOOD_ITEM
<<pk>>FOOD_ITEM_ID
NAME
QUANTITY
<<FK>>SUPPLIER_ID
ANIMAL_FOOD_ITEM
<<pk,fk>> ANIMAL_ID
<<pk,fk>> FOOD_ITEM_ID
VET
<<pk>>VET_ID
NAME
QUALIFICATION
ANIMAL
<<pk>>ANIMAL_ID
TYPE
TOTAL_NO
<<FK>>CATEGORY_ID
SUPPLIER
<<pk>>SUPPLIER_ID
SUPPLIER_NAME
CONTACT_NO
ADDRESS
IN_HOUSE_VET
<<pk,fk>> VET_ID
SALARY
EXTERNAL_VET
<<pk,fk>> VET_ID
COUNTRY
VISITING_FEES
CATEGORY
<<pk>>CATEGORY_ID
CATEGORY_TYPE
EMERGENCY
<<PK>>EMERGENCY_ID
TYPE
CASE
PROGRESS_SO_FAR
1
N
1
N
N
1
1
N
1

One to One mapping

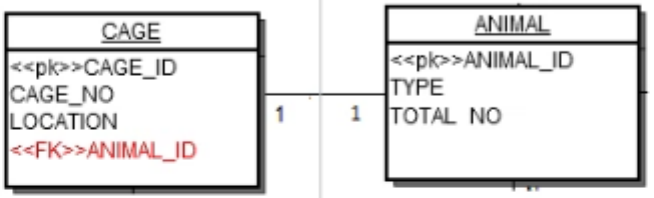


Step 1: Map non relationship attributes

```
@Entity
public class Cage implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer cage_id;
    private String cage_no;
    private String location;

    //getters and setters omitted
}
```

```
@Entity
public class Animal implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer animal_id;
    private String type;
    private Integer total_No;

    //getters and setters omitted
}
```

One to One mapping



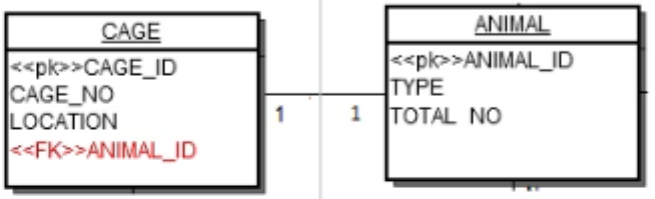
This is a bidirectional relationship since both entities have a reference to each other.

Step 2 : Add relationship attributes

```
@Entity
public class Cage implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer cage_id;
    private String cage_no;
    private String location;
    @OneToOne
    private Animal animal;
}
```

```
@Entity
public class Animal implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer animal_id;
    private String type;
    private Integer total_No;
    @OneToOne
    private Cage cage;
}
```

One to One mapping

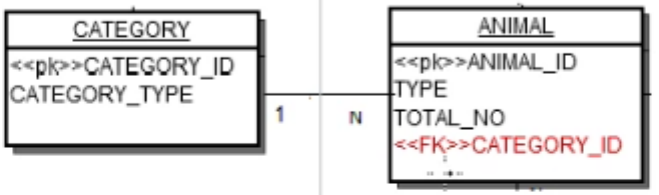


Step 3 : Introduce join column in table containing the foreign key ,mappedBy attribute on other side

```java
@Entity
public class Cage implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer cage_id;
    private String cage_no;
    private String location;
    @OneToOne
    @JoinColumn(name="animal_id")
    private Animal animal;
}
```

```java
@Entity
public class Animal implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer animal_id;
    private String type;
    private Integer total_No;
    @OneToOne(mappedBy="animal")
    private Cage cage;
}
```

One to Many mapping

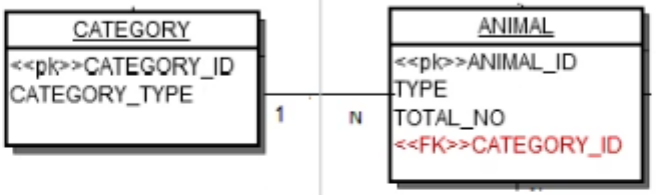


Step 2 : Add relationship attributes

```
@Entity
public class Category implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer category_id;
    private String category_type;
    @OneToMany
    private List<Animal> animalList;
}
```

```
@Entity
public class Animal implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer animal_id;
    private String type;
    private Integer total_no;
    @ManyToOne
    private Category categoryId;
}
```

One to Many mapping

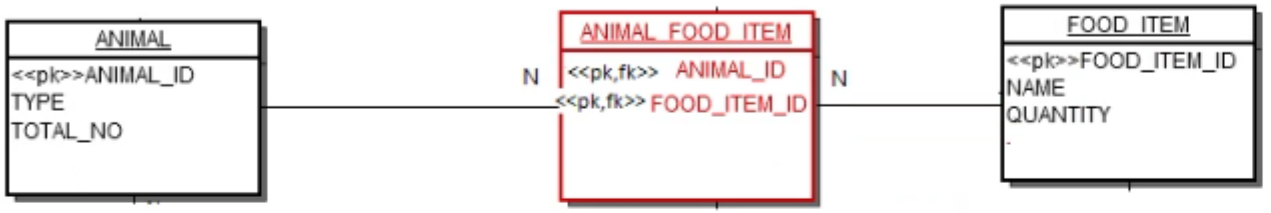


Step 2 : Introduce join column in table containing the foreign key ,mappedBy attribute on other side

```java
@Entity
public class Category implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer category_id;
    private String category_type;
    @OneToMany(mappedBy = "category")
    private List<Animal> animalList;
}
```

```java
@Entity
public class Animal implements Serializable {
    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer animal_id;
    private String type;
    private Integer total_no;
    @ManyToOne
    @JoinColumn(name = "category_id")
    private Category category;
 }
```

Many to Many Mapping

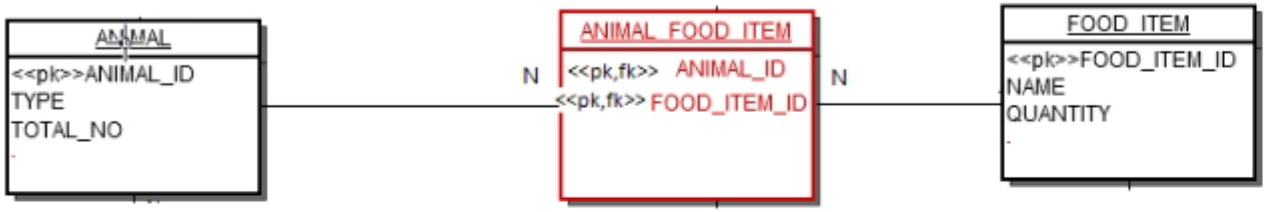


Step 1: Map non relationship attributes

```java
@Entity
public class Animal implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer animal_id;
    private String type;
    private Integer total_no;

}
```

```java
@Entity
public class FoodItem implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer food_item_id;
    private String name;
    private String quantity;


}
```

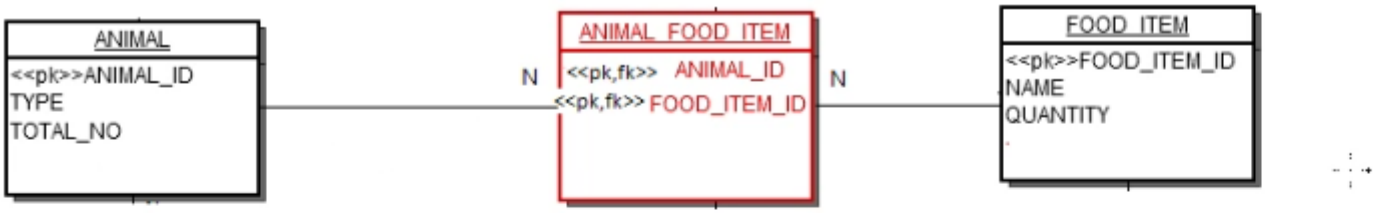
Many to Many Mapping



Step 2 : Add relationship attributes

```
@Entity
public class Animal implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer animal_id;
    private String type;
    private Integer total_no;
    @ManyToMany
    private List<FoodItem> foodItemList ;
}
```

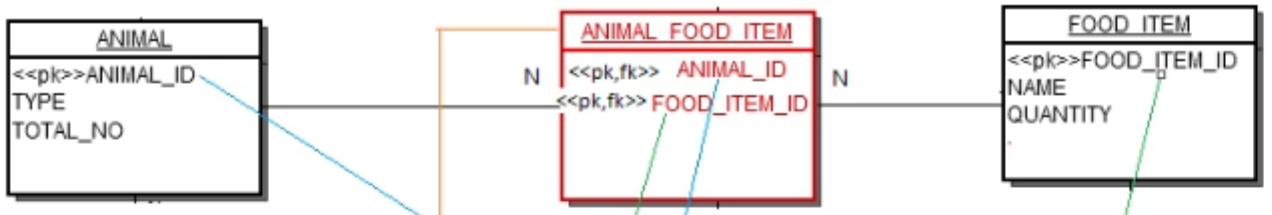
```
@Entity
public class FoodItem implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer food_item_id;
    private String name;
    private String quantity;
    @ManyToMany
    private List<Animal> animalList;

}
```



Step 3: Introduce Join Table on any one side ,mappedBy attribute on other side

```
@Entity
public class Animal implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer animal_id;
    private String type;
    private Integer total_no;
    @ManyToMany
    @JoinTable(name = "animal_food_item",
    joinColumns = {@JoinColumn(name = "animal_id", referencedColumnName = "animal_id")},
    inverseJoinColumns = {
        @JoinColumn(name = "food_item_id", referencedColumnName = "food_item_id")})
    private List<FoodItem> foodItemList ;
}
```

```
@Entity
public class FoodItem implements Serializable {

    @Id
```



Step 3: Introduce Join Table on any one side ,mappedBy attribute on other side

```
@Entity
public class Animal implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer animal_id;
    private String type;
    private Integer total_no;
    @ManyToMany
    @JoinTable(name = "animal_food_item",
    joinColumns = {@JoinColumn(name = "animal_id", referencedColumnName = "animal_id")},
    inverseJoinColumns = {
        @JoinColumn(name = "food_item_id", referencedColumnName = "food_item_id")})
    private List<FoodItem> foodItemList;
}
```

```
@Entity
public class FoodItem implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

JPA Inheritance Mapping Strategies

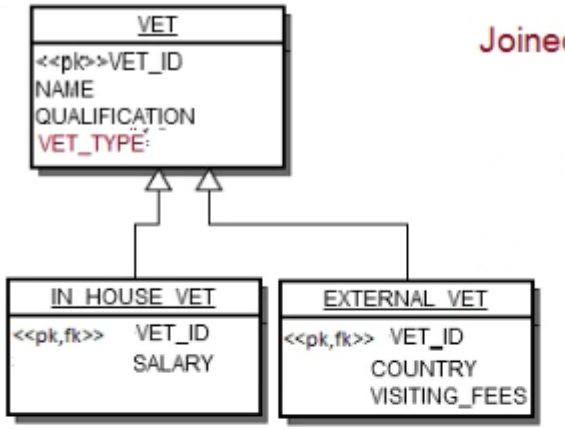
1) Single-Table Strategy

2) Joined Strategy

3) Table-per-Concrete-Class Strategy

Strategy to use is not picked up randomly. It will depend upon how the tables are modeled in the database.

Joined Strategy



VET

vet_id	name	qualification	vet_type
1	Ashitraj more	mvsc	IN_VET
2	Raj	bvsc	IN_VET
3	Steven	mvsc	EXT_EVT
4	Rakesh	mvsc	IN_VET
5	John	mvsc	EXT_VET

IN_HOUSE_VET

vet_id	salary
1	35000
2	30000
4	29000

EXTERNAL_VET

vet_id	country	visiting_fees
3	UK	500
5	US	450

```java
@Entity
@Inheritance(strategy= InheritanceType.JOINED)
@DiscriminatorColumn(name="VET_TYPE")
public abstract class Vet implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer vet_id;
    private String name;
    private String qualification;

    //getters and setters
 }
```

```java
@Entity
@DiscriminatorValue("IN_VET")
public class In_House_Vet extends Vet {

    private Integer salary;

    //getters and setters

}
```

```java
@Entity
@DiscriminatorValue("EXT_VET")
public class External_Vet extends Vet {

    private String country;
    private Integer visiting_fees;
```

Single-Table Strategy

vet_id	name	qualification	salary	country	visiting_fees	vet_type
1	Ashitraj More	mvsc	35000	NULL	NULL	IN_VET
2	Raj	bvsc	30000	NULL	NULL	IN_VET
3	Steven	mvsc	NULL	UK	500	EXT_VET
4	Rakesh	mvsc	29000	NULL	NULL	IN_VET
5	John	mvsc	NULL	US	450	EXT_VET

```java
@Entity
@Inheritance(strategy= InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name="VET_TYPE")
public abstract class Vet implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY
    private Integer vet_id;
    private String name;
    private String qualification;

    //getters and setters
}
```

```java
@Entity
@DiscriminatorValue("IN_VET")
public class In_House_Vet extends Vet {

    private Integer salary;

    //getters and setters

}
```

```java
@Entity
@DiscriminatorValue("EXT_VET")
public class External_Vet extends Vet {
```

Table-per-Concrete-Class Strategy

IN_HOUSE_VET

vet_id	name	qualification	salary
1	Ashitraj More	mvsc	35000
2	Raj	bvsc	30000
3	Rakesh	mvsc	29000

EXTERNAL_VET

vet_id	name	qualification	country	visiting_fees
1	Steven	mvsc	UK	500
2	John	mvsc	US	450

```java
@Entity
@Inheritance(strategy= InheritanceType.TABLE_PER_CLASS)
public abstract class Vet implements Serializable {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer vet_id;
    private String name;
    private String qualification;

    //getters and setters
 }
```

```java
@Entity
public class In_House_Vet extends Vet {

    private Integer salary;

    //getters and setters

}
```

```java
@Entity
public class External_Vet extends Vet {

    private String country;
    private Integer visiting_fees;
```

SQL join

```
SELECT * FROM animal join category On animal.category_id = category.category_id
```

JPQL Join

```
SELECT a.type,c.categoryType FROM  Animal a JOIN a.category c
```