**BIDIRECCIONAL – UNO A MUCHOS - 01**

|  |  |
| --- | --- |
| **SOURCE (non-owner)** | **TARGET (owner)** |
|  | |
| * When an entity (Department) is associated with a collection of other entities (Employee), it is more often in the form of a one-to-many mapping. * A department would normally have a number of employees. |  |
| * When a relationship is bidirectional there actually two mappings, one for each direction * In this example, there is a **one-to-many** mapping from Department to Employee | * And there is a **many-to-one** mapping from Employee back to Department. |
|  |  |
| * When a **source** entity (**Department**) has an arbitrary number of **target** entities (**Employee**) stored in its collections, * There is no scalable way to store those references in the database table that it maps to. |  |
|  |  |
| * How it would store an arbitrary number of keys in a single row? |  |
|  |  |
|  | * Instead, it must let the table of the entities in the collection have foreign keys back to the source entity table. |
|  |  |
|  |  |
| * This is way the one-to-may relationship associations is usually bidirectional. * And the **inverse (non-owning)** is the **“one”** side. | * The **owner** is the **many** side. |
|  | |
| * The side that does not have the join column (foreign key) is called the **non-owning** or **inverse side**. * While the presence of the **mappedBy** element means the entity is on the **inverse side** of the relationship. * The **mappedyBy** is used when the relationship is **bidirectional**. | * Independent of source and target sides, one of the two sides will have the join column (foreign key) in its table. * That side is called the **owning side** or the **owner** of the relationship. * The absence of the **mappedBy** element in the mapping annotation implies **ownership** of the relationship. |
|  | |
|  | * If the target entity table have foreign keys thay points back to the source table * The target entities should have many-to-one association back to the source entity. * Employee is the owner of the relationship. * Employee has the join column (foreign key) |
|  |  |
| * On the Department side of the relationship, we need to map the **employees** collection of **Employe** entities as **one-to-many** associations using the **@OneToMany** annotation. * Because this is the inverse side of the relationship, we need to include the **mappedBy** element. |  |
|  |  |
| * The one-to-many mapping should be the inverse side, so the **mappedBy** element should be used. | * The many-to-one side should be the owning side, so the join column should be defined on that side. |
| * Failing to specify the **mappedBy** element in the **@OneToMany** annotation will cause the provider to treat it as a unidirectional one-to-many relationship that is defined to use a **join table**. |  |
|  |  |