Hibernate Tek Yönlü(Unidirectional) @ManyToMany Kullanm

```
@Entity
public class Author {
     @Id
     @GeneratedValue(strategy = GenerationType.IDENTITY)
     @Column(name = "ID")
     private int id;


     @Column(name="NAME")
     private String name;

     @Column(name="SURNAME")
     private String surname;

     @ManyToMany(cascade = CascadeType.ALL)
     @JoinTable(
         name="Author_Book",
         joinColumns = @JoinColumn( name="AuthorID"),
         inverseJoinColumns = @JoinColumn( name="BookID")
     )
     private List<Book> books;
}
@Entity
public class Book {
     @Id
     @GeneratedValue(strategy = GenerationType.IDENTITY)
     private int id;

     @Column(name="NAME")
     private String name;
}
```

`@JoinTable` her iki tablonun primary key'lerini tutar. Bu annotasyonda **joinColumns** ksm **Book** entity'nin **DEL,** bulunduu entity'in yani **Author** entity'nin **PRIMARY KEY** deerini, yani **id** deerini temsil eder. id deerleri veritabannda **Author** isimli tablonun **ID** isimli sütunda tutulur. Fakat `@JoinColumn(name="AuthorID")` diyerek, bu id deerinin, **Author_Book** tablosunun **AuthorID** isimli sütunda tutulmasn salarz. `@JoinColumn(name="AuthorID")` yerine istediimiz sütun adn verebiliriz: `@JoinColumn(name="adfsfmsdfmsaf").` Bu ekilde kullanld zaman **Author** tablosunun primary key deerleri yani id deerleri **Author_Book** tablosundaki **adfsfmsdfmsaf** isimli sütunda tutulmu olur.

Ayn mantkla `inverseJoinColumns = @JoinColumn( name="BookID")` **Author_Book** tablosunda **Book** entity'nin **primary key** deerlerinin yani id deerlerin **BookID** isimli sütunda tutulacan ifade eder. Dikkat edersek **Book** entity içerisinde id deikenine `@Column` annatasyonu kullanarak herhangi bir sütun ismi verilmedi. Bu durumda **Book** tablosu id isimli bir sütun ile primary key'leri tutmu olur. `@Column` annotasyonundaki name property, istediimiz sütun adn vermeyi salar. Bu annotasyon kullanlmad zaman sütun ad deiken adyla ayn olur. Bundan dolay **Book** tablosu aadaki gibi olur:

```
id | NAME
```

`cascade=Cascade.ALL` ifadesi hangi entity içerisinde kullanlmsa, örneimizde **Author** entity, o entity'i **parent** yapar! Dier entity ise, örneimizdeki **Book child entity** olur! Parent entity üzerinde hangi ilem yaplrsa, insert, update, delete, tüm bu ilemler `cascade=Cascade.ALL` dediimiz için child entity'de de yaplr!

Not: cascade hiç kullanlmam olsayd **Author** eklendiinde **Book** eklenmezdi; fakat **Author_Book** tablosu güncellenirdi. Çünkü cascade sadece child entity, örneimizdeki **Book** entity, üzerinde etki yapar.

Örnek olarak Insert ileminde aadaki gibi sorgu çalr:

```
       insert
       into
           Author
           (ID,NAME,SURNAME)
       values
           (?, ?, ?)

       insert
       into
```

```
        Book
        (id, NAME)
    values
        (?, ?)

    insert
    into
        Author_Book
        (AuthorID, BookID)
    values
        (?, ?)
```