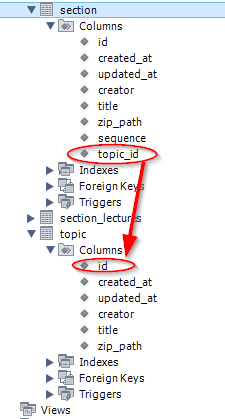
## bidirectional @OneToMany association

|  |
| --- |
| @Entity  **public** **class** Topic **extends** AbstractLesson{  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **long** id;  @OneToMany(cascade = CascadeType.***ALL***,  fetch = FetchType.***LAZY***,  mappedBy = "topic") // bidirectional One-to-many association  **private** Set<Section> sections = **new** HashSet<>();  … … |
| @Entity  **public** **class** Section **extends** AbstractLesson{  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **long** id;  **private** **int** sequence;    @ManyToOne(fetch=FetchType.***LAZY***)  @JoinColumn(name="topic\_id", nullable=**false**) // bidirectional @OneToMany association  **private** Topic topic; |

topic和sectiion双向绑定。也就是说topic可以有很多的section，而每一个section都有固定的topic。

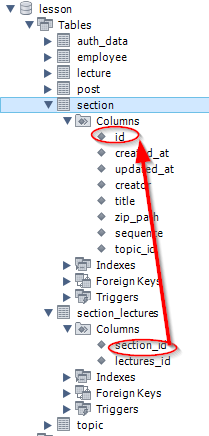


每个孩子都知道自己的父母是谁。每个section都属于确定的topic，换句话说，任何一个section，都只能属于一个确定的topic。

## unidirectional @OneToMany association

|  |
| --- |
| @Entity  **public** **class** Section **extends** AbstractLesson{  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** **long** id;  @OneToMany(cascade = CascadeType.***ALL***, // unidirectional @OneToMany association, parent-side only OneToMany  orphanRemoval = **true**)  @JoinColumn(name="section\_id")  **private** List<Lecture> lectures = **new** ArrayList<>(); |
| @Entity  @Table(name="lecture")  **public** **class** Lecture **extends** AbstractLesson{  @Id  @GeneratedValue(strategy = GenerationType.***IDENTITY***)  **private** Long id;  **private** **int** sequence=0;  @Column(name="VIDEO\_PATH")  **private** String videoFilePath;  @Column(name = "SOURCE\_PATH")  **private** String sourceFilePath;  // @ManyToOne(fetch=FetchType.LAZY)  // @JoinColumn(name="section\_id", nullable=false)  // private Section section; |

section可以有很多的lecture，但一个lecture可以属于多个不同的section。



这和many-to-many的做法很是接近。只是在lecture子类中没有定义section父类的团组。只在section中定义了lecture的团组。