Annotation Details

@OneToOne Specifies a one to one relationship with a corresponding object.

@OneToMany Specifies a single object that maps to many objects.

@ManyToOne Specifies a collection of objects that map to a single object.

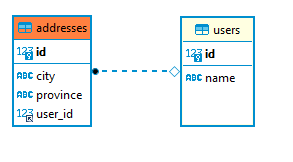
@Entity Specifies an object that maps to a database table.

@Table Specifies which database table this object maps too.

@JoinColumn Specifies which column a foregin key is stored in.

@JoinTable Specifies an intermediate table that stores foreign keys.

1. Quan hệ 1-1



1 user – 1 address

* 1. Uni-directional

@Table(name = "addresses")  
@Entity  
@Data  
public class Address {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String city;  
  
 private String province;  
  
 @OneToOne()  
 @JoinColumn(name = "user\_id")  
 private User user;  
}

@Table(name = "users")  
@Entity  
@Data  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String name;  
}

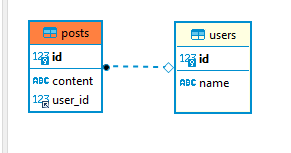
Tác động lên class User ko làm ảnh hưởng tới class Address. Lấy thông tin address sẽ chứa thông tin user.

* 1. Bi-directional

@Table(name = "users")  
@Entity  
@Getter  
@Setter  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String name;  
  
 @OneToOne(mappedBy = "user")  
 private Address address;  
}

@Table(name = "addresses")  
@Entity  
@Getter  
@Setter  
public class Address {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String city;  
  
 private String province;  
  
 @OneToOne()  
 @JoinColumn(name = "user\_id")  
 private User user;  
}

1. Quan hệ 1-n



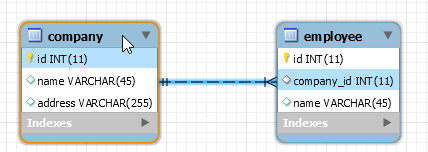
1 user – n post

@Table(name = "users")  
@Entity  
@Getter  
@Setter  
public class User {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String name;  
  
 @OneToMany(mappedBy = "user")  
 private List<Post> posts;  
}

@Table(name = "posts")  
@Entity  
@Getter  
@Setter  
public class Post {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private Long id;  
  
 private String content;  
  
 @ManyToOne  
 @JoinColumn(name = "user\_id")  
 private User user;  
}

1. Quan hệ n-n
2. ON DELETE ON UPDATE

Giả sử ta có mối quan hệ giữa bảng company và bảng employee như sau:



Mối quan hệ giữa company và employee là 1-n (một company chứa nhiều employee).

Khi xóa hoặc update id của một row trong table company sẽ có các trường hợp sau:

* **Trường hợp 1**: chưa có employee nào tham chiếu tới company đó (company đó chưa có employee nào) -> xóa / update bình thường.
* **Trường hợp 2**: đã có employee tham chiếu tới company đó (company đó đã có employee). Tùy thuộc vào giá trị của ON DELETE, ON UPDATE:
  + **NO ACTION/RESTRICT**: Không cho delete/update id của bản ghi cha (default)
  + **SET NULL**: delete/update id của bản ghi từ bảng company (cha) và set company\_id của các employee (con) đang tham chiếu tới bằng null (Tạm hiểu là công ty bị xóa, thay đổi thì các nhân viên sẽ không thuộc công ty nào cả)
  + **CASCADE**:

+ Trường hợp update id của bản ghi ở table company (cha) sẽ update tương ứng id của bản ghi ở table employee (con) đang tham chiếu tới.

+ Trường hợp delete company (cha) sẽ delete toàn bộ các bản ghi ở bảng employee (con) đang tham chiếu tới bản ghi cha đó. (Xóa company sẽ xóa toàn bộ nhân viên của company đó).